В. П. Майданюк, О. Н. Романюк, С. Є. Тужанський

Основи теорії інформації та кодування

Міністерство освіти і науки України
Вінницький національний технічний університет

В. П. Майданюк, О. Н. Романюк, С. Є. Тужанський

Основи теорії інформації та кодування

**Електронний навчальний посібник
комбінованого (локального та мережного) використання**

Вінниця
ВНТУ
2022

УДК 621.391 (075)
М18

Рекомендовано до видання Вченою радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 7 від 31.03.2022 р.)

Рецензенти:
В. А. Лужецький, доктор технічних наук, професор
О. М. Василевський, доктор технічних наук, професор
Т. І. Коробейнікова, кандидат технічних наук, доцент

Майданюк, В. П.

М18 Основи теорії інформації та кодування : електронний навчальний посібник комбінованого (локального та мережного) використання [Електронний ресурс] / Майданюк В. П., Романюк О. Н., Тужанський С. Є. – Вінниця : ВНТУ, 2022. – 133 с.

У посібнику наведено теоретичні відомості та приклади розв’язання типових задач з дисципліни «Основи теорії інформації та кодування», що сприятиме кращому засвоєнню навчальних матеріалів завдяки орієнтації на практичне застосування. Перелік та зміст тем посібника відповідає програмі дисципліни. Навчальний посібник призначений для студентів спеціальності 121 – «Інженерія програмного забезпечення».

УДК 621.391 (075)

© ВНТУ, 2022

ЗМІСТ

ВСТУП 4
1 ІНФОРМАЦІЯ ТА ЇЇ КІЛЬКІСНЕ ОЦІНЕННЯ 5
1.1 Теоретичні положення 5
1.2 Приклади розв’язування задач 18
1.3 Контрольні питання і задачі 32
2 ЕФЕКТИВНЕ КОДУВАННЯ ІНФОРМАЦІЇ 34
2.1 Теоретичні положення 34
2.2 Приклади розв’язування задач 57
2.3 Контрольні питання і задачі 69
3 ЗАВАДОСТІЙКЕ КОДУВАННЯ ІНФОРМАЦІЇ 71
3.1 Теоретичні положення 71
3.2 Приклади розв’язування задач 92
3.3 Контрольні питання і задачі 100
4 ОСНОВИ КРИПТОЛОГІЇ 102
4.1 Теоретичні положення 102
4.2 Приклади розв’язування задач 116
4.3 Контрольні питання і задачі 125
ВИСНОВКИ 127
ЛІТЕРАТУРА 128
СЛОВНИК ОСНОВНИХ ТЕРМІНІВ (GLOSSARY) 131

ВСТУП

В широкому значенні теорія інформації *(information theory)* – це і кодування *(coding, encode),* і криптографічний *(cryptographic)* захист інформації. Початком розвитку теорії інформації стали роботи К. Шеннона «Математична теорія зв'язку» та «Теорія зв'язку в секретних системах» [1], які були опубліковані в 1948 – 1949 роках і заклали наукові основи економного, завадостійкого кодування та наукової криптології (з грецької *criptologic*, kryptos – таємний, logos – слово) – науки, що займається проблемою захисту інформації шляхом її перетворення.

Навчальний посібник містить теоретичні відомості та розв'язки типових задач з чотирьох основних напрямків теорії інформації.

У першому розділі розглянуто базові питання теорії інформації, такі як кількісне оцінення інформації, моделі джерел інформації та каналів зв'язку, їх основні характеристики, наведено приклади розв'язання типових задач.

Другий розділ містить необхідні теоретичні відомості та розв'язання задач ефективного або економного кодування інформації.

У третьому розділі розглянуто основи завадостійкого кодування інформації, зокрема, побудови корегувальних кодів *(correctings codes)*, приклади розв'язання типових задач призначені для кращого засвоєння навчального матеріалу.

Теоретичні відомості та розв'язання типових задач наукової криптології наведено в четвертому розділі.

Кожний розділ завершується контрольними питаннями та вправами для самостійної роботи, що допоможе кращому засвоєнню матеріалу та набуттю практичних навичок у використанні методів теорії інформації в комп'ютерних системах.

Навчальний посібник призначений для студентів спеціальності 121 – «Інженерія програмного забезпечення» і може використовуватись під час вивчення дисципліни «Основи теорії інформації та кодування», а також для окремих розділів інших споріднених дисциплін.

1 ІНФОРМАЦІЯ ТА ЇЇ КІЛЬКІСНЕ ОЦІНЕННЯ

1.1 Теоретичні положення

Кількісне оцінювання інформації. Різні способи оцінювання кількості інформації подано в [1 – 7].

Комбінаторний (combinatory) підхід:

$$H(U) = \log N, \tag{1.1}$$

де U – дискретне джерело інформації,

N – множина можливих станів джерела,

H(U) – ентропія *(entropy)* джерела.

Ця міра інформації була запропонована Хартлі в 1928 році. Якщо основу логарифма вибрати такою, що дорівнює двом, то одиниця невизначеності називається двійковою одиницею або бітом *(binary digit – bit)*, а якщо за основу логарифма вибрати десять, то невизначеність отримаємо в десяткових одиницях на один стан – дітах *(dit)*. Комбінаторний підхід не враховує статичні властивості джерела інформації.

Імовірнісний (probabilistic) підхід. Джерело інформації характеризується сукупністю станів *(states)* з імовірностями *(probabilities)* їх появи (ансамблем).

$$U = \begin{bmatrix} u_1 & u_2 \dots & u_N \\ p_1 & p_2 \dots & p_N \end{bmatrix}, \quad \Sigma p_i = 1. \tag{1.2}$$

Якщо імовірність появи станів джерела інформації різні, то міра невизначеності згідно з К. Шенноном:

$$H(U) = -C\sum_{i=1}^{N} p_i \log p_i, \tag{1.3}$$

де C – довільне додатне число. Для основи логарифма «2» C = 1, і

$$H(U) = -\sum_{i=1}^{N} p_i \log_2 p_i. \tag{1.4}$$

Ця міра названа ентропією, оскільки збігається з ентропією фізичної системи, визначеною Больцманом.

Якщо скористатись умовними імовірностями, то отримаємо умовну ентропію, однак незалежно від способу отримання імовірностей характер залежності (1.4) залишається незмінним.

Міра Шеннона – це узагальнення міри Хартлі для джерела з нерівноімовірними станами [3 – 4].

Алгоритмічний підхід [7]. Якщо дані можуть бути описані деякими формулами або породжувальними (твірними) алгоритмами *(algorithms)*, тобто характеризуються деякими закономірностями, то ентропія буде дорівнювати мінімальній кількості інформації для передачі цих формул або алгоритмів від джерела до приймача інформації. Наприклад дані, отримані внаслідок застосування методів машинної графіки.

Властивості ентропії

1. Ентропія – дійсна невід'ємна величина.
2. Ентропія – величина обмежена.
3. Ентропія – дорівнює нулю, якщо імовірність одного із станів дорівнює 1, тобто стан джерела повністю визначений.
4. Ентропія – максимальна, коли всі стани джерела рівноімовірні.
5. Ентропія джерела з двома станами u_1 і u_2 змінюється від 0 до 1, досягаючи максимуму за рівності їх імовірностей.
6. Ентропія – об'єднання декількох статистично незалежних джерел інформації дорівнює сумі ентропій початкових джерел:

$$H(UV...Z) = H(U) + H(V) + ... + H(Z). \qquad (1.5)$$

7. Ентропія характеризує середню невизначеність вибору одного стану з ансамблю і нічого більше[2, 3].

Умовна ентропія і її властивості. Розглянемо ентропію об'єднання двох *статистично залежних* ансамблів U і V.

Об'єднання ансамблів характеризується матрицею P(UV) імовірностей $p(u_i v_j)$ всіх можливих комбінацій станів u_i ансамблю U і станів v_j ансамблю V. Нехай $1 \le i \le N$, $1 \le j \le K$, тоді матриця P(UV) буде мати такий вигляд [2, 3]:

$$P(UV) = \begin{pmatrix} p(u_1 v_1) & p(u_2 v_1) & ... & p(u_N v_1) \\ p(u_1 v_2) & p(u_2 v_2) & ... & p(u_N v_2) \\ ... & ... & & ... \\ p(u_1 v_K) & p(u_2 v_K) & ... & p(u_N v_K) \end{pmatrix}. \qquad (1.6)$$

Додавши рядки матриці, отримаємо інформацію про ансамбль U:

$$U = \begin{pmatrix} u_1 & u_2 & u_N \\ p(u_1) & p(u_2) & p(u_N) \end{pmatrix}, \qquad (1.7)$$

а додавши стовпці матриці, отримаємо інформацію про ансамбль V:

$$V = \begin{pmatrix} v_1 & v_2 & v_K \\ p(v_1) & p(v_2) & p(v_K) \end{pmatrix}. \qquad (1.8)$$

Відповідно до (1.4):

$$H(UV) = -\sum_{i=1}^{N}\sum_{j=1}^{K} p(u_i v_j)\log p(u_i v_j).$$

З урахуванням того, що $p(u_i v_j) = p(u_i)p_{u_i}(v_j) = p(v_j)p_{v_j}(u_i)$, отримаємо:

$$H(UV) = -\sum_{i=1}^{N}\sum_{j=1}^{K} p(u_i)\cdot p_{u_i}(v_j)\cdot \log\left[p(u_i)\cdot p_{u_i}(v_j)\right] =$$

$$= -\sum_{i=1}^{N}\sum_{j=1}^{K} p(u_i)\cdot p_{u_i}(v_j)\cdot \log p(u_i) - \sum_{i=1}^{N}\sum_{j=1}^{K} p(u_i)\cdot p_{u_i}(v_j)\cdot \log p_{u_i}(v_j) =$$

$$= -\sum_{i=1}^{N} p(u_i)\log p(u_i)\cdot \sum_{j=1}^{K} p_{u_i}(v_j) - \sum_{i=1}^{N} p(u_i)\sum_{j=1}^{K} p_{u_i}(v_j)\log p_{u_i}(v_j).$$

Позначимо $H_U(V) = -\sum_{i=1}^{N} p(u_i)\sum_{j=1}^{K} p_{u_i}(v_j)\log p_{u_i}(v_j)$ – умовна ентропія ансамблю V відносно ансамблю U, отримаємо:

$$H(UV) = H(U) + H_U(V). \qquad (1.9)$$

Якщо виразити $p(u_i v_j)$ через іншу умовну імовірність, то знайдемо:

$$H(UV) = H(V) + H_V(U). \qquad (1.10)$$

Оскільки $H_U(V) \le H(V)$ і $H_V(U) \le H(U)$, то

$$H(UV) \le H(U) + H(V).$$

Ентропія об’єднання декількох статистично залежних джерел така:

$$H(UVZ\ldots W) = H(U) + H_U(V) + H_{UV}(Z) + \ldots + H_{UVZ\ldots}(W). \quad (1.11)$$

Кількість інформації як міра знятої невизначеності. Внаслідок дії завад отриманий елемент повідомлення може відрізнятись від переданого. Тобто, в процесі передачі повідомлення $z_1, z_2, ..., z_i, ..., z_N$ отримаємо $\omega_1, \omega_2, ..., \omega_j, ..., \omega_N$.

Нехай стани джерела інформації статистично незалежні, тоді апріорна (*aprior* – до отримання елемента повідомлення) часткова невизначеність появи елемента повідомлення z_i така:

$$H(z_i) = -\log_2 p(z_i), \quad (1.12)$$

де $p(z_i)$ – апріорна ймовірність появи елемента повідомлення z_i.

Вважаючи, що статистичні зв'язки між елементами повідомлення і завадою відсутні, апостеріорна (*aposterior* – післядослідна) часткова невизначеність така:

$$H_{\omega_j}(z_i) = -\log_2 p_{\omega_j}(z_i), \quad (1.13)$$

де $p_{\omega_j}(z_i)$ – апостеріорна імовірність реалізації джерелом елемента повідомлення z_i у разі отримання елемента повідомлення ω_j.

Часткова умовна ентропія така:

$$H_{\omega_j}(Z) = -\sum_{i=1}^{N} p_{\omega_j}(z_i) \log p_{\omega_j}(z_i). \quad (1.14)$$

Це випадкова величина, яка залежить від того, який конкретно елемент повідомлення прийнято.

Апостеріорна ентропія джерела інформації така:

$$H_W(Z) = \sum_{j=1}^{N} p(\omega_j) * H_{\omega_j}(Z),$$

або

$$H_W(Z) = -\sum_{j=1}^{N}\sum_{i=1}^{N} p(\omega_j) p_{\omega_j}(z_i) \log p_{\omega_j}(z_i). \quad (1.15)$$

За наявності завад середня кількість інформації, яка міститься в кожному прийнятому елементі повідомлення відносно переданого, дорівнює різниці апріорної і апостеріорної ентропій джерела [2, 3, 8]:

$$I(Z) = H(Z) - H_W(Z). \quad (1.16)$$

Подавши апріорну і апостеріорну ентропії відповідно до виразів (1.4) і (1.15), отримаємо:

$$I(Z) = -\sum_{i=1}^{N} p(z_i)\log p(z_i) + \sum_{j=1}^{N}\sum_{i=1}^{N} p(\omega_j)p_{\omega_j}(z_i)\log p_{\omega_j}(z_i) = \\ = \sum_{j=1}^{N}\sum_{i=1}^{N} p(z_i\omega_j)\log\frac{p(z_i\omega_j)}{p(z_i)p(\omega_j)}. \quad (1.17)$$

Модель системи передачі. Кодування джерела інформації та кодування каналу. Кодер (рис. 1.1) призначений для перетворення повідомлення в форму, придатну для передачі по каналах зв'язку, оскільки безпосередня передача сигналу від джерела до отримувача по каналу зв'язку через завади і спотворення часто неможлива. Розділення кодування і декодування, пов'язаного з джерелом, від кодування і декодування, пов'язаного з каналом, спрощує аналіз і синтез кодера. Кодер джерела забезпечує ефективне кодування повідомлень, а кодер для каналу забезпечує завадостійку передачу по каналах зв'язку [2, 3, 8 – 10].

Розрізняють джерела неперервних і дискретних повідомлень. Джерело дискретних повідомлень формує дискретні повідомлення із обмеженої кількості елементарних повідомлень.

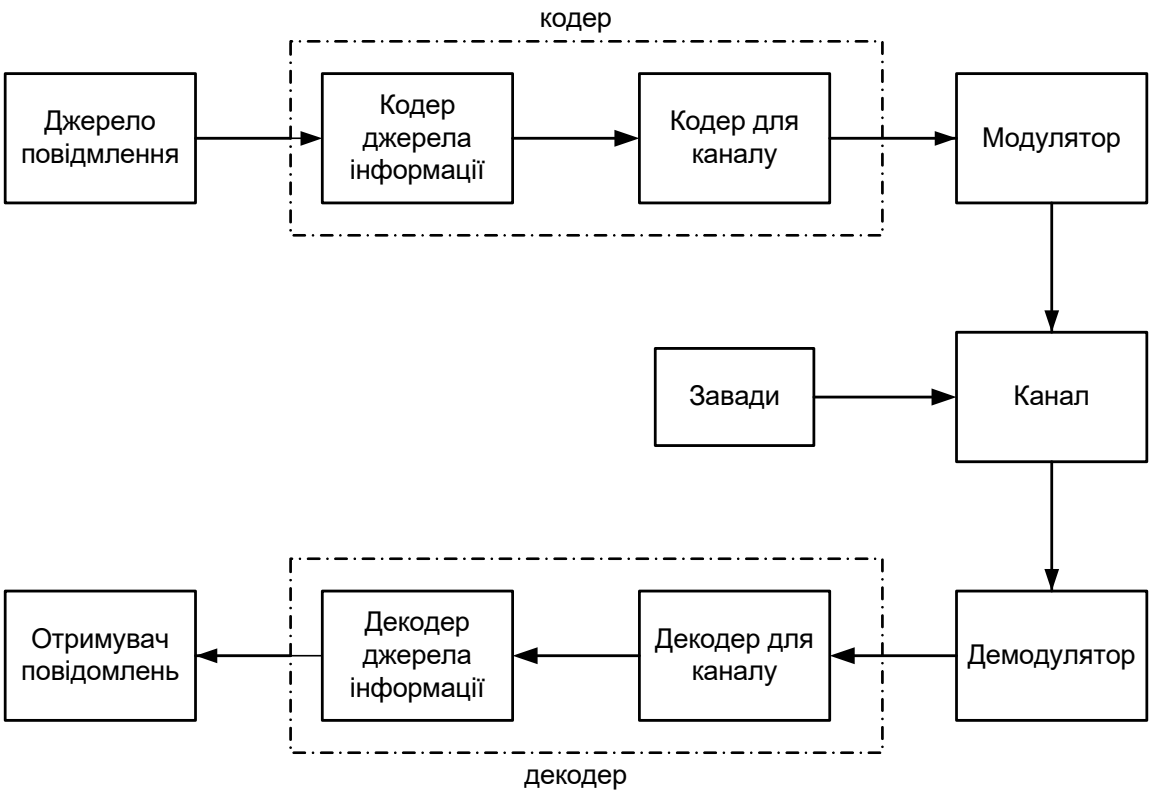

Рисунок 1.1 – Модель системи передачі

Канал зв'язку – це сукупність пристроїв і фізичних середовищ, які забезпечують передачу повідомлень з одного місця в інше, або від одного моменту часу до іншого.

Канал для передачі дискретних повідомлень називається дискретним каналом, а для передачі неперервних – неперервним.

Неперервні повідомлення z(t) шляхом дискретизації і квантування завжди можна перетворити в дискретні і таким чином перейти від неперервного каналу до дискретного.

Якщо дією завад в каналі можна знехтувати, то для аналізу використовують модель у вигляді ідеалізованого каналу, який називається каналом без завад, і в якому кожному повідомленню на вході однозначно відповідає повідомлення на виході. Коли вимоги до достовірності великі, використовують більш складну модель каналу – канал із завадами.

Якщо відомо статистичні дані про повідомлення на вході і виході каналу, а також фізичні характеристики каналу, то канал вважається заданим.

В подальшому розглядаються лише дискретні повідомлення і канали.

Модель джерел дискретних повідомлень. Для побудови моделі (model) необхідно знати обсяг алфавіту знаків z_1, z_2, …, z_N, з яких джерело формує повідомлення, та імовірності створення окремих знаків (characters) з урахуванням можливих взаємозв'язків між ними. Під час доведення основних положень теорії інформації К. Шеннон використовував модель, яка називається ергодичним (ergodic) джерелом повідомлення. Така модель задовольняє умови стаціонарності (stationary) і ергодичності. Умова стаціонарності – імовірність окремих знаків і їх поєднань не залежить від розташування останніх по довжині повідомлень, умова ергодичності - статистичні закономірності, отримані під час дослідження одного достатньо довгого повідомлення з імовірністю, близькою до «1», справедливі для всіх повідомлень, що формуються джерелом [1 – 4].

Якщо стаціонарне джерело вибирає кожен знак послідовності незалежно від інших знаків, то таке джерело завжди ергодичне і називається джерелом без пам'яті *(source without memory)* або джерелом Бернуллі. Джерело без пам'яті називають комбінаторним, якщо всі знаки повідомлення рівноімовірні. Монотонним називають джерело без пам'яті, яке відповідає таким умовам:

$$p(z_i) \geq p(z_j), \quad \text{якщо} \quad i \geq j$$
$$\text{або} \quad p(z_i) \leq p(z_j), \quad \text{якщо} \quad i \leq j. \tag{1.18}$$

Однак у більшості джерел інформації вибір одного знака повідомлення залежить від того, які знаки були вибрані джерелом раніше. Такі джерела називаються джерелами з пам'яттю *(source with memory)*. Функціонування

таких джерел описується з використанням ланцюгів Маркова *(Markov chain)*. Порядок ланцюга Маркова n визначає послідовність подій, імовірності яких залежать від того, які n подій передували заданій.

Якщо L – обсяг алфавіту джерела, тоді кількість станів джерела R не буде перевищувати L^n. Нехай $s_1, s_2,\ldots,s_q,\ldots, s_R$ – стани джерела, імовірність вибору в стані s_q знака z_i – $p_q(z_i)$. Якщо джерело знаходиться в стані s_q, то його часткова ентропія така:

$$H(s_q) = -\sum_{i=1}^{L} p_q(z_i)\log p_q(z_i). \qquad (1.19)$$

Ентропію джерела повідомлення отримаємо, усереднюючи випадкову величину $H(s_q)$ за всіма можливими станами q=1, 2, 3, …, R:

$$H(Z) = -\sum_{q=1}^{R} p(s_q)\sum_{i=1}^{L} p_q(z_i)\log p_q(z_i), \qquad (1.20)$$

де $p(s_q)$ – імовірність того, що джерело знаходиться в стані s_q.

Розглянемо окремі випадки.

1. Статистичні зв’язки між знаками відсутні - n=0, $R=L^n=L^0=1$, $p(s_1)=1$:

$$H(Z) = -\sum_{i=1}^{L} p(z_i)\log p(z_i). \qquad (1.21)$$

2. Кореляційні зв’язки спостерігаються тільки між двома знаками (простий ланцюг Маркова) – n=1, R=L:

$$p_q(z_i) = p_{z_q}(z_i),$$
$$H(Z) = -\sum_{q=1}^{L} p(z_q)\sum_{i=1}^{L} p_{z_q}(z_i)\log\ p_{z_q}(z_i)\ . \qquad (1.22)$$

На практиці обмежуються значеннями n=1, 2 через складність обчислень.

Властивості ергодичних послідовностей знаків. Розглянемо ергодичне джерело без пам’яті, яке формує знаки z_1, z_2, z_3, з імовірностями 0,1; 0,3; 0,6. В достатньо довгій послідовності знаків в середньому на один знак z_1 припадатиме три знаки z_2 і шість знаків z_3. Однак в короткій послідовності знаків існує імовірність того, що вона міститиме [1 – 4]:

- тільки знаки z_1 або z_2, або z_3;
- тільки знаки z_1 і один z_2 або z_3;
- тільки знаки z_2 і один z_1 або z_3;
- тільки знаки z_2 і два знаки z_1 або z_3;
- тільки знаки z_1 і три знаки z_2 або z_3;

і т.д.

Із збільшенням довжини послідовності, імовірність появи таких послідовностей зменшується. Фундаментальну властивість довгих послідовностей знаків, які створюються ергодичним джерелом повідомлень, відображає така теорема [1 – 4].

Означення. Які б малі не були два додатних числа $\delta>0$ і $\mu>0$, за достатньо великого N всі послідовності можуть бути розбиті на 2 групи:

- нетипові послідовності
- типові послідовності.

Сумарна ймовірність нетипових послідовностей дуже мала і менша за як завгодно мале число δ. Для типових послідовностей ймовірності їх появи практично однакові і ймовірність p будь-якої такої послідовності задовольняє нерівність:

$$\left| \frac{\log \frac{1}{p}}{N} - H(Z) \right| < \mu, \qquad (1.23)$$

де H(Z) – ентропія джерела повідомлення,

μ – як завгодно мале додатне число.

Це відношення називається властивістю асимптотичної рівномірності довгих послідовностей [3].

Оскільки за $N \to \infty$ джерело повідомлень з імовірністю, близькою до «1», видає тільки типові послідовності, кількість яких дорівнює 1/p, а невизначеність кожної послідовності з урахуванням їх рівноімовірності становить log (1/p). Тоді $(\log \frac{1}{p})/N$ – невизначеність, яка припадає на один знак повідомлення. Зрозуміло, що ця величина не має відрізнятись від ентропії джерела, що і констатує співвідношення (1.23).

Доведемо теорему для найпростішого ергодичного джерела без пам'яті. Нехай є довга послідовність з N елементів. Імовірність появи знаків алфавіту z_1, z_2, ...z_n дорівнює p_1, p_2, …p_n, і тоді згідно із законом великих чисел в послідовності міститься Np_1 елементів z_1, Np_2 елементів z_2 і т. д. Імовірність p реалізації будь-якої типової послідовності буде близькою до:

$$p = p_1^{Np_1} \cdot p_2^{Np_2} \cdot \ldots \cdot p_n^{Np_n}.$$

Прологарифмувавши цей вираз, отримаємо:

$$\log p = Np_1 \cdot \log p_1 + Np_2 \cdot \log p_2 + \ldots + Np_n \cdot \log p_n = N \sum_{i=1}^{n} P_i \log P_i.$$

Звідки $-\frac{\log p}{N} = -\sum_{i=1}^{n} p_i \log p_i$, або

$$\frac{\log \frac{1}{p}}{N} = H(Z). \qquad (1.24)$$

Що і необхідно було довести.

У випадку рівноімовірного і незалежного вибору букв джерела нетипові послідовності відсутні.

В інших випадках, за досить великого N типові послідовності становлять незначну частку від загальної кількості послідовностей. Якщо обсяг алфавіту джерела n і кількість знаків в послідовності N, то загальна кількість можливих послідовностей становить:

$$n_1 = n^N = 2^{\log_2 n^N} = 2^{N \cdot \log_2 n}. \qquad (1.25)$$

А кількість типових послідовностей становитиме:

$$n_2 = \frac{1}{p} = 2^{N \cdot H(Z)}. \qquad (1.26)$$

Відношення

$$\frac{n_1}{n_2} = 2^{N \cdot (\log_2 n - H(Z))}. \qquad (1.27)$$

Оскільки $\log_2 n > H(Z)$, то $n_1 >> n_2$, ця нерівність підсилюється із збільшенням N. К. Шеннон показав, що розглянуті вище властивості є основою ефективного кодування інформації [2, 3].

Міра надмірності. Міра надмірності *(redundancy)* визначається так [2, 3]:

$$D = \frac{H_{max}(Z) - H(Z)}{H_{max}(Z)}, \qquad (1.28)$$

де $H_{max}(Z) = \log_2 L$, L – обсяг алфавіту;

H(Z) – ентропія джерела повідомлень.

Міра надмірності показує, наскільки добре використовуються знаки цього джерела інформації: якщо D = 0, то повідомлення, які формуються джерелом повідомлень, оптимальні з погляду кількості інформації, яку вони переносять.

За відсутності завад для передачі певної кількості інформації I необхідно:

$$k_1 = \frac{I}{H_{max}(Z)} (\text{знаків}).$$

Для реального джерела для передачі такої самої кількості інформації I необхідно більше знаків:

$$k_2 = \frac{I}{H(Z)} > k_1.$$

Тоді з (1.28) отримаємо:

$$D = \frac{k_2 - k_1}{k_2} = \frac{H_{max}(Z) - H(Z)}{H_{max}(Z)}. \qquad (1.29)$$

Тому говорять також про надмірність знаків в повідомленні.

Наслідки наявності надмірності неоднозначні.

1. Надмірні повідомлення потребують додаткових витрат на передачу, обробку та зберігання.
2. Наявність надмірності підвищує завадостійкість повідомлення.

В технічних системах надмірність вилучається, а для підвищення завадостійкості вводиться «раціональна» надмірність, яка дозволяє виявити і виправити найбільш імовірні помилки простими технічними засобами.

Продуктивність джерела дискретних повідомлень. Продуктивність джерела (source productivity) повідомлення – це кількість інформації, яку виробляє джерело в одиницю часу [2].

Середня тривалість видачі джерелом одного знака становить:

$$\tau_д = \sum_{q=1}^{R} p(s_q) \sum_{i=1}^{l} p_q(z_i) \tau_{qz_i}, \qquad (1.30)$$

де τ_{qz_i} – тривалість видачі знака z_i в стані s_q;

$p_q(z_i)$ – імовірність формування знака z_i в стані s_q;

$p(s_q)$ – імовірність появи стану s_q.

Продуктивність джерела:

$$\bar{I}(Z) = \frac{H(Z)}{\tau_д}. \qquad (1.31)$$

Тривалість знаків бажано вибирати пропорційною їх імовірностям. Якщо $\tau_0 = \tau$, тобто не залежить від стану джерела, то

$$\bar{I}(Z) = \frac{H(Z)}{\tau}. \qquad (1.32)$$

Моделі дискретних каналів. Дискретний канал – сукупність засобів, призначених для передачі дискретних сигналів [2, 3]. Пристрій кодування перетворює дискретні повідомлення, які складаються з послідовності знаків алфавіту z_1, z_2, …, z_n, на послідовність символів u_1, u_2, u_m. У більшості випадків m<n, але вони можуть і збігатися.

Матеріальним втіленням символу є елементарний сигнал, який формується в процесі дискретної зміни певного параметра носія інформації (маніпуляції), а кожній послідовності символів ставиться у відповідність складний сигнал. Множина сигналів скінченна. Поняття «елементарний сигнал» і «символ» або «складний сигнал» і «послідовність символів» – синоніми. Множина символів на вході і виході каналу разом з описом імовірних властивостей передачі окремих сигналів задають інформаційну модель каналу із завадами. Канал характеризується множиною станів і може переходити з одного стану в інший як з часом, так і залежно від послідовності символів, що передаються. Кожний стан каналу характеризується матрицею умовних імовірностей $p(v_j/u_i)$ (імовірність того, що переданий символ u_i буде сприйнятий на виході каналу як v_j). У нестаціонарному каналі зв'язку перехідні ймовірності $p(v_j/u_i)$ залежать від часу, що характерно для реальних каналів. Якщо перехідні імовірності не залежать від часу або ця залежність несуттєва, то використовується модель стаціонарного каналу. Нестаціонарний канал для різних проміжків часу може бути поданий рядом стаціонарних каналів.

Якщо перехідні імовірності в цьому стані залежать від його попередніх станів, то такий канал називають каналом з пам'яттю *(channels with memory)*, інакше *каналом без пам'яті (memoryless channel)*. У *стаціонарному каналі без пам'яті* перехідні імовірності постійні для всіх станів, тобто фактично канал має один стан.

Якщо кількість символів на вході і виході каналу однакова і дорівнює k, то канал називають k-їчним каналом канал зв'язку.

Стаціонарний дискретний двійковий канал без пам'яті однозначно визначається 4-ма умовними імовірностями $p_0(0)$, $p_1(1)$, $p_0(1)$, $p_1(0)$. Модель такого каналу зображають у вигляді графа (рис. 1.2).

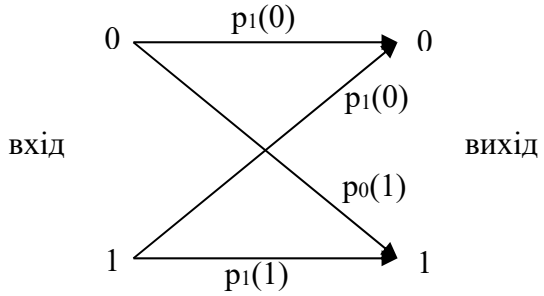

Рисунок 1.2 – Модель стаціонарного дискретного двійкового каналу без пам'яті

У двійковому симетричному каналі *(binary symmetric channel)* $p_1(0) = p_0(1) = q$, а $p_0(0) = p_1(1) = p$. Символи на його вході правильно приймаються з імовірністю p і неправильно – з імовірністю q=1-p, що спрощує математичну модель каналу.

Модель дискретного каналу зі стиранням характеризується тим, що алфавіт вихідних символів відрізняється від алфавіту вхідних символів. На вході символи 0 та 1, а на виході каналу фіксується стан, який позначається символом стирання S і сигнал в цьому стані може бути віднесений як до одиниці, так і до нуля. Під час декодування символи стирання S виправити легше, чим помилково визначені. Відомо дві моделі каналу зі стиранням: за відсутності (рис. 1.3, а) і за наявності (рис. 1.3, б) трансформації символів.

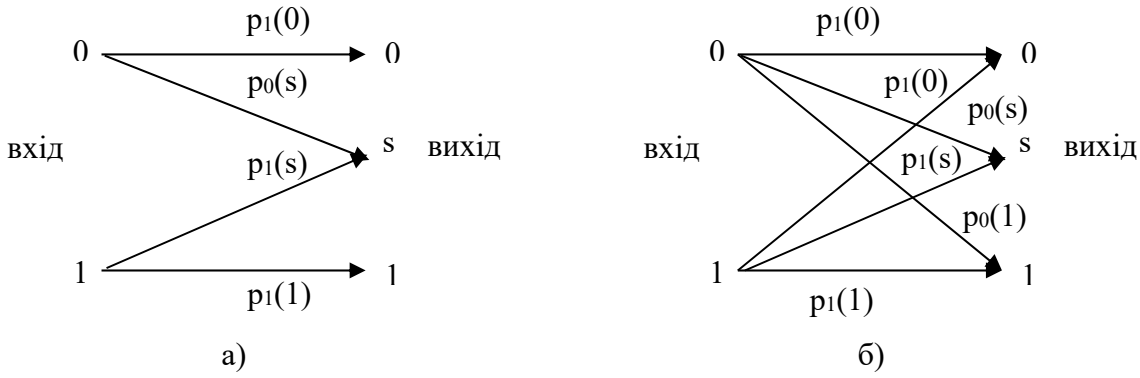

Рисунок 1.3 – Моделі каналів зі стиранням

Швидкість передачі інформації по дискретному каналу. Розрізняють два поняття швидкості передачі дискретного каналу [2, 4].

1. Технічна швидкість передачі – швидкість маніпуляції *(manipulation speed)*:

$$V_\tau = \frac{1}{\tau_{ср}}, \tag{1.33}$$

де $\tau_{ср}$ - середнє значення тривалості символу. Одиницею вимірювання технічної швидкості є [бод] – швидкість, за якої в секунду передається один символ.

2. Інформаційна швидкість або швидкість передачі інформації *(information rate)*. Визначається середньою кількістю інформації, яка передається по каналу в одиницю часу:

$$\bar{I}(V,U) = V_\tau \cdot I(V,U). \tag{1.34}$$

де $I(V,U)$ – середня кількість інформації, яка переноситься одним символом.

Пропускна здатність дискретного каналу без завад. Для дискретного каналу без завад пропускна здатність *(capacity)* визначається так [2, 3]:

$$C_д = \max \bar{I}(V,U) = \max(V_\tau \cdot I(V,U)). \quad (1.35)$$

Пропускна здатність – це максимальна швидкість передачі інформації по каналу за найдосконаліших способів передачі і прийому. За відсутності завад між множиною можливих V символів на виході і U на вході каналу має місце однозначна відповідність, тобто:

$$I(U,V) = I(V,U) = H(U),$$

де $H_{max}(U)=\log_2 m$, а m – обсяг алфавіту символів.

Таким чином:

$$C_д = V_\tau \cdot \log_2 m. \quad (1.36)$$

Це означає, що для збільшення швидкості передачі і наближенні її до пропускної здатності дискретному каналу без завад повідомлення мають так перетворюватись в кодері, щоб символи у вихідній послідовності з'являлись якомога рівноімовірніше, а статистичні зв'язки між ними були відсутні.

Для практичних розрахунків пропускна здатність дискретного каналу без завад визначається згідно з формулою Найквіста для каналу без завад [11]:

$$C_д = 2W \cdot \log_2 V,$$

де W – смуга пропускання каналу (Гц);

V – кількість дискретних рівнів сигналу.

Пропускна здатність дискретного каналу із завадами. У випадку дії завад відповідність між символами на вході і на виході каналу зв'язку неоднозначна. Середня кількість інформації, що передається одним символом, така [2]:

$$I(V,U)=H(V)-H_U(V)=H(U)-H_V(U)=\sum_{i=1}^{m_1}\sum_{j=1}^{m_2} p(v_j u_i)\log\frac{p(v_j u_i)}{p(u_i)p(v_j)}, \quad (1.37)$$

де $H_v(U)$ – умовна ентропія ансамблю U відносно V;

$H_u(V)$ – умовна ентропія ансамблю V відносно U;

m_1 – обсяг алфавіту вхідних повідомлень;

m_2 – обсяг алфавіту на виході каналу.

А швидкість передачі інформації по каналу із завадами:

$$\bar{I}(V,U) = V_\tau I(V,U) = V_\tau \sum_{i=1}^{m_1}\sum_{j=1}^{m_2} p(v_j u_i)\log\frac{p(v_j u_i)}{p(u_i)p(v_j)}, \qquad (1.38)$$

де V_τ – швидкість маніпуляції (технічна швидкість).

Змінюючи за рахунок перетворення (кодера каналу) статистичні властивості послідовностей символів на вході каналу, I(V,U) можна максимізувати, тобто пропускна здатність дискретного каналу із завадами така:

$$C_д = \max_{p\{U\}} V_\tau \cdot I(V,U), \qquad (1.39)$$

де $p\{U\}$ – множина можливих розподілів імовірностей вхідних сигналів.

Пропускна здатність – це найбільша кількість інформації, яка може бути передана по каналу в одиницю часу з як завгодно малою імовірністю помилки, але граничні можливості каналу ніколи не використовуються. Для характеристики ступеня завантаження каналу вводиться коефіцієнт його використання:

$$\lambda = \frac{\bar{I}(Z)}{C_д}, \qquad (1.40)$$

де $\bar{I}(Z)$ – продуктивність джерела інформації.

Оскільки, як буде показано нижче, $0 \le \bar{I}(Z) \le C_д$, то $0 \le \lambda \le 1$ [2].

Часто в практичних розрахунках для визначення пропускної здатності каналу із завадами використовують формулу Найквіста для каналу із завадами [11]:

$$C_д = W \cdot \log_2(1 + S/N).$$

$$\sigma = 10 \cdot \lg\frac{S}{N}.(дБ),$$

де S/N – відношення сигнал/шум в каналі, яке звичайно задається децибелах (дБ).

1.2 Приклади розв'язування задач

Приклад 1.1. Яка мінімальна кількість зважувань на рівноплечих вагах необхідна для визначення однієї фальшивої монети серед 27 однакових монет. Відомо, що фальшива монета більш легка.

Розв'язання. Невизначеність ансамблю U така:

$$H(U) = \log_2 27 = 3 \ \log_2 3 \ \text{біт}.$$

Ансамблю U′, який утворює одне зважування, відповідають три можливих стани – ваги знаходяться в рівновазі, права чаша ваг легша, ліва чаша ваг легша. Тому невизначеність цього ансамблю така:

$$H(U') = \log_2 3 \text{ біт.}$$

Тоді,

$$H(U) = 3 \ H(U').$$

Таким чином, для визначення фальшивої монети достатньо три зважування. Якщо на кожну чашу ваг покласти по дев'ять монет, то фальшива монета буде або на легшій чаші ваг, або серед тих дев'яти, що не зважувались у випадку рівноваги чаш. На другому зважуванні з дев'яти монет, серед яких одна фальшива, беруть по три монети і аналогічно визначають групу з фальшивою монетою. На третьому зважуванні з цієї групи на чаші ваг кладуть по одній монеті і точно визначають фальшиву монету.

Приклад 1.2. Знайти невизначеність, яка припадає на букву джерела інформації (російський алфавіт) і порівняти її з невизначеністю, яка б була у того ж джерела за рівномірного використання букв.

Розв'язання. Відомо, що кількість букв в російському алфавіті N=32. Тоді за рівномірного використання букв невизначеність, яка припадає на букву джерела інформації, така:

$$H(U) = \log_2 N = \log_2 32 = 5 \text{ біт.}$$

Але відомо, що букви російського алфавіту в тексті зустрічаються не з однаковою імовірністю [2]. Розподіл імовірностей букв в текстах російською мовою має приблизно такий вигляд:

Таблиця 1.1 - Розподіл імовірностей букв в текстах російською мовою

Буква	Імовірність	Буква	Імовірність	Буква	Імовірність	Буква	Імовірність
А	0,064	Й	0,010	Т	0,056	Ъ, Ь	0,015
Б	0,015	К	0,029	У	0,021	Ы	0,016
В	0,039	Л	0,036	Ф	0,02	Э	0,003
Г	0,014	М	0,026	Х	0,09	Ю	0,007
Д	0,026	Н	0,056	Ц	0,04	Я	0,019
Е	0,074	О	0,096	Ч	0,013	-	0,0143
Ж	0,008	П	0,024	Ш	0,006		
З	0,015	Р	0,041	Щ	0,003		
И	0,064	С	0,047				

Для джерела інформації із заданим ансамблем згідно з мірою Шеннона (1.7) ентропія джерела така:

$$H(U) = -0{,}064 \log_2 0{,}064 - 0{,}015 \log_2 0{,}015 - \ldots - 0{,}143 \log_2 0{,}143 = 4{,}42 \text{ біт}.$$

Таким чином, нерівномірність імовірностей використання букв зменшує ентропію джерела з 5 до 4,42 біт.

Приклад 1.3. Визначити ентропію H(U), H(V), $H_V(U)$, H(UV), якщо задана матриця імовірностей станів системи, яка об'єднує джерела U і V:

$$P(VU) = \begin{pmatrix} 0,4 & 0,1 & 0 \\ 0 & 0,2 & 0,1 \\ 0 & 0 & 0,2 \end{pmatrix}.$$

Розв'язання. Знайдемо безумовні імовірності станів джерел U і V як суму імовірностей по рядках і стовпцях заданої матриці:

$$p(v_j) = [0,4 \;\; 0,3 \;\; 0,3];$$

$$p(u_i) = \begin{pmatrix} 0,5 \\ 0,3 \\ 0,2 \end{pmatrix}.$$

Тоді

$$H(U) = -\sum_{i=1}^{N} p(u_i) \log_2 p(u_i) = -0,5 \log_2 0,5 - 0,3 \log_2 0,3 - 0,2 \log_2 0,2 = 1,485 \text{ біт};$$

$$H(V) = -\sum_{j=1}^{K} p(v_j) \log_2 p(v_j) = -0,4 \log_2 0,4 - 0,3 \log_2 0,3 - 0,3 \log_2 0,3 = 1,57 \text{ біт}.$$

Умовні імовірності такі:

$$p_{v_j}(u_i) = \frac{p(u_i v_j)}{p(v_j)};$$

$$p_{v_1}(u_1) = = \frac{0,4}{0,4} = 1;$$

$$p_{v_2}(u_1) = p_{v_3}(u_2) = \frac{0,1}{0,3} = 0,33;$$

$$p_{v_2}(u_2) = p_{v_3}(u_3) = \frac{0,2}{0,3} = 0,67;$$

$$p_{v_3}(u_1) = p_{v_1}(u_2) = p_{v_1}(u_3) = p_{v_2}(u_3) = 0.$$

Тоді умовна ентропія ансамблю U відносно ансамблю V така:

$$H_V(U) = -\sum_j p(v_j) \sum_i p_{v_j}(u_i) \log p_{v_j}(u_i) = -[0,4(1 \cdot \log_2 1) + 0,3(0,33 \cdot \log_2 0,33 +$$

$$+ 0,67 \cdot \log_2 0,67) + 0,3(0,33 \log_2 0,33 + 0,67 \log_2 0,67)] \approx 0,55 \text{ біт}.$$

Ентропія об’єднання двох статистично залежних ансамблів (UV) дорівнює:

$$H(UV) = -\sum_i \sum_j p(u_i v_j) \log_2 p(u_i v_j) =$$

$$= -[0{,}4 \cdot \log_2 0{,}4 + 0{,}1 \cdot \log_2 0{,}1 + 0{,}1 \cdot \log_2 0{,}1 + 0{,}2 \cdot \log_2 0{,}2 + 0{,}2 \cdot \log_2 0{,}2] = 2{,}12 \text{ біт.}$$

Перевіримо результат за формулою:

$$H(UV) = H(V) + H_V(U) = 1{,}57 + 0{,}55 = 2{,}12 \text{ біт.}$$

Приклад 1.4. Алфавіт стаціонарного дискретного джерела повідомлень складається з 4-х знаків z_1, z_2, z_3, z_4. Безумовні імовірності вибору знаків такі:

$$p(z_1) = p(z_2) = p(z_3) = p(z_4) = \frac{1}{4}.$$

А умовні імовірності задано у таблиці 1.2.

Таблиця 1.2 – Умовні імовірності вибору знаків

$z_q \backslash z_i$	z_1	z_2	z_3	z_4
z_1	1/3	1/3	1/3	0
z_2	1/3	1/3	1/3	0
z_3	1/3	1/3	1/3	0
z_4	0	0	0	1

Визначити, чи є це джерело ергодичним.

Розв’язання. З таблиці 1.2 видно, що якщо першим вибрати знак z_4, то формується послідовність, яка містить тільки знаки z_4, а якщо першим вибрати один із знаків z_1, z_2, z_3, то джерело почне формувати послідовність з рівноімовірною появою цих знаків. З урахуванням (1.26) ентропія такого джерела становитиме:

$$H(Z) = -\frac{3}{4}\left(\frac{1}{3}\log_2\frac{1}{3} + \frac{1}{3}\log_2\frac{1}{3} + \frac{1}{3}\log_2\frac{1}{3}\right) - \frac{1}{4}\log_2 1 = 1{,}19 \text{ біт.}$$

Розглянемо два випадки.

1. Першим вибрано один із знаків z_1, z_2, z_3. Тоді ентропія послідовності така:

$$H_1(Z) = \left(-\frac{1}{3}\log_2\frac{1}{3}\right) \cdot 3 = 1.585 \text{ біт.}$$

2. Першим вибрано Z_4. Тоді ентропія послідовності така:

$$H_2(Z) = 1 \cdot \log_2 1 = 0.$$

Оскільки, ентропії послідовностей не збігаються з ентропією джерела, то воно не є ергодичним.

Приклад 1.5: Оцінити, яку частку загальної кількості можливих послідовностей необхідно врахувати в практичних розрахунках, якщо ергодичне джерело характеризується такими параметрами: n =16, H(Z) = 3,5 дв.од., N = 50.

Розв'язання. Знайдемо загальну кількість послідовностей:

$$n_1 = n^N = 16^{50} = 2^{200}.$$

Кількість типових послідовностей становитиме:

$$n_2 = 2^{N \cdot H(Z)} = 2^{50 \cdot 3,5} = 2^{175}.$$

Частка типових послідовностей така:

$$\frac{n_2}{n_1} = \frac{2^{175}}{2^{200}} = \frac{1}{2^{25}} \approx \frac{1}{30 \cdot 10^6}.$$

Приклад 1.6. Визначити можливий ефект в процесі передачі тексту українською мовою від вилучення надмірності.

Розв'язання. Відомо, що для тексту українською мовою [2]:

$$H_{max}(Z) = \log_2 32 = 5 \text{ біт.}$$
$$H(Z) = 4,42 \text{ біт.}$$

З урахуванням перехідних імовірностей встановлено, що ентропія тексту становить:

$$H(Z) = 1,5 \text{ біт.}$$

Тоді надмірність тексту становитиме:

$$D = \frac{5 - 1.5}{5} = \frac{3.5}{5} = 0,7.$$

Тобто, повне вилучення надмірності дозволило б підвищити продуктивність каналу зв'язку майже в 3 рази.

Приклад 1.7. Розподіл імовірностей появи символів на виході немарковського джерела з алфавітом X потужності $M = 5$ є таким:

$$p(x_1) = p(x_2) = 0,1;\ p(x_3) = 0,15;\ p(x_4) = 0,2;\ p(x_5) = 0,45.$$

Тривалості символів $\tau_1 = \tau_2 = \tau_3 = 2\,\text{мс}$; $\tau_4 = 1\,\text{мс}$; $\tau_5 = 3\,\text{мс}$.

Розрахувати ентропію, продуктивність та надмірність джерела [12].

Розв'язання. Користуючись виразами (1.1), (1.4), (1.28), (1.30) знаходимо:

1) ентропія

$$H = (-0{,}1 \cdot \log_2 0{,}1) \cdot 2 - 0{,}15 \cdot \log_2 0{,}15 -$$
$$- 0{,}2 \cdot \log_2 0{,}2 - 0{,}45 \cdot \log_2 0{,}45 \approx 2{,}058\,\text{біт};$$

2) середня тривалість символу

$$\tau = 2 \cdot (0{,}1 + 0{,}1 + 0{,}15) + 1 \cdot 0{,}2 + 3 \cdot 0{,}45 = 2{,}25\,\text{мс};$$

3) продуктивність

$$\bar{I} = 2{,}058 / (2.25 \cdot 10^{-3}) \approx 914{,}67\,\text{біт/с};$$

4) надмірність

$$D = 1 - (2{,}058 / \log_2 5) = 0{,}114\,.$$

Приклад 1.8. Маємо два дискретних немарковських джерела інформації з алфавітами $X = \{x_1, x_2, x_3\}$ та $Y = \{y_1, y_2, y_3\}$ з такими розподілами ймовірностей появи символів:

$$p(x_1) = 0{,}7;\ p(x_2) = 0{,}2;\quad p(x_3) = 0{,}1;$$
$$p(y_1) = 0{,}4;\ p(y_2) = 0{,}35;\ p(y_3) = 0{,}25.$$

Не розраховуючи ентропії джерел, дати відповідь, яке з них має більшу ентропію.

Розв'язання. Оскільки розподіл імовірностей появи символів на виході джерела з алфавітом Y є більш близьким до рівноімовірного, ентропія цього джерела буде більшою, ніж джерела з алфавітом X.

Розрахунки підтверджують цей висновок:

$$H(Y) = 1{,}56 > H(X) = 1{,}16.$$

Приклад 1.9. Матриця імовірностей сумісної появи символів на виходах двох немарковських джерел з алфавітами $X = \{x_1, x_2, x_3\}$ та $Y = \{y_1, y_2, y_3\}$ має вигляд:

$$\begin{bmatrix} p(x_1, y_1) & p(x_2, y_1) & p(x_3, y_1) \\ p(x_1, y_2) & p(x_2, y_2) & p(x_3, y_2) \\ p(x_1, y_3) & p(x_2, y_3) & p(x_3, y_3) \end{bmatrix} = \begin{bmatrix} 0{,}0336 & 0{,}0264 & 0{,}0200 \\ 0{,}3150 & 0{,}2475 & 0{,}1875 \\ 0{,}0714 & 0{,}0561 & 0{,}0425 \end{bmatrix}.$$

Визначити, яке з джерел має більшу ентропію та чи є джерела статистично незалежними.

Розв'язання. Розрахуємо безумовні імовірності появи символів на виходах першого та другого джерел:

$$p(x_1) = 0{,}0336 + 0{,}3150 + 0{,}0714 = 0{,}42;$$
$$p(x_2) = 0{,}0264 + 0{,}2475 + 0{,}0561 = 0{,}33;$$
$$p(x_3) = 0{,}0200 + 0{,}1875 + 0{,}0425 = 0{,}25;$$
$$p(y_1) = 0{,}0336 + 0{,}0264 + 0{,}0200 = 0{,}08;$$
$$p(y_2) = 0{,}3150 + 0{,}2475 + 0{,}1875 = 0{,}75;$$
$$p(y_3) = 0{,}0714 + 0{,}0561 + 0{,}0425 = 0{,}17.$$

Тоді ентропії джерел за виразом (1.4):

$$H(X) = 1{,}553 \text{ біт}; \qquad H(Y) = 1{,}037 \text{ біт}.$$

Таким чином, джерело X має більшу ентропію, ніж джерело з Y.

Статистична залежність джерел перевіряється декількома способами.

Перший спосіб ґрунтується на перевірці рівності (1.5). Сумісна ентропія $H(X,Y)$ згідно з виразом (1.4) така:

$$H(X,Y) = 2{,}59 \text{ біт}.$$

Оскільки $H(X) + H(Y) = 1{,}533 + 1{,}037 = 2{,}59 = H(X,Y)$, то джерела є статистично незалежними.

Другий спосіб ґрунтується на перевірці виконання співвідношень $p(x_i, y_k) = p(x_i) \cdot p(y_k)$ для всіх пар символів:

$$p(x_1) \cdot p(y_1) = 0{,}42 \cdot 0{,}08 = 0{,}0336;$$
$$p(x_2) \cdot p(y_1) = 0{,}33 \cdot 0{,}08 = 0{,}0264;$$
$$p(x_3) \cdot p(y_1) = 0{,}25 \cdot 0{,}08 = 0{,}0200;$$
$$p(x_1) \cdot p(y_2) = 0{,}42 \cdot 0{,}75 = 0{,}3150;$$
$$p(x_2) \cdot p(y_2) = 0{,}33 \cdot 0{,}75 = 0{,}2475;$$
$$p(x_3) \cdot p(y_2) = 0{,}25 \cdot 0{,}75 = 0{,}1875;$$
$$p(x_1) \cdot p(y_3) = 0{,}42 \cdot 0{,}17 = 0{,}0714;$$
$$p(x_2) \cdot p(y_3) = 0{,}33 \cdot 0{,}17 = 0{,}0561;$$
$$p(x_3) \cdot p(y_3) = 0{,}25 \cdot 0{,}17 = 0{,}0561.$$

Розраховані імовірності цілком збігаються із відповідними значеннями ймовірностей $p(x_i, y_k)$ сумісної появи символів, наведеними в умові задачі, тому джерела є статистично незалежними.

Третій спосіб оцінення статистичної залежності джерел – обчислення повної взаємної інформації $I(X,Y)$. Аналізуючи вираз (1.36), легко зрозуміти, що для джерел цієї задачі $I(X,Y) = 0$, оскільки для усіх пар x_i, y_k

$$\log_2 \frac{p(x_i, y_k)}{p(x_i)p(y_k)} = \log_2 1 = 0 \ .$$

Четвертий спосіб розв'язання задачі ґрунтується на аналізі матриці умовних імовірностей. Розрахуємо, наприклад, умовні ймовірності $p(x_i / y_k)$, користуючись виразом $p(x_i / y_k) = p(x_i, y_k) / p(y_k)$:

$$\begin{bmatrix} p(x_1/y_1) & p(x_2/y_1) & p(x_3/y_1) \\ p(x_1/y_2) & p(x_2/y_2) & p(x_3/y_2) \\ p(x_1/y_3) & p(x_2/y_3) & p(x_3/y_3) \end{bmatrix} = \begin{bmatrix} 0{,}42 & 0{,}33 & 0{,}25 \\ 0{,}42 & 0{,}33 & 0{,}25 \\ 0{,}42 & 0{,}33 & 0{,}25 \end{bmatrix}.$$

Всі елементи кожного стовпця однакові і дорівнюють безумовній ймовірності $p(x_i)$ появи відповідного символу x_i. Це означає, що ймовірність появи символу на виході першого джерела не залежить від символу на виході другого джерела. Можна переконатись, що і в матриці умовних ймовірностей $p(y_k / x_i)$ всі елементи кожного стовпця будуть однаковими і дорівнювати $p(y_k)$.

Приклад 1.10. Маємо три дискретних немарковських джерела інформації з алфавітами $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2\}$, $Z = \{z_1, z_2\}$. Матриці імовірностей сумісної появи пар символів є такими:

$$\begin{bmatrix} p(x_1,y_1) & p(x_2,y_1) & p(x_3,y_1) \\ p(x_1,y_2) & p(x_2,y_2) & p(x_3,y_2) \end{bmatrix} = \begin{bmatrix} 0{,}28 & 0{,}08 & 0{,}04 \\ 0{,}42 & 0{,}12 & 0{,}06 \end{bmatrix},$$

$$\begin{bmatrix} p(x_1,z_1) & p(x_2,z_1) & p(x_3,z_1) \\ p(x_1,z_2) & p(x_2,z_2) & p(x_3,z_2) \end{bmatrix} = \begin{bmatrix} 0{,}50 & 0{,}05 & 0 \\ 0{,}20 & 0{,}15 & 0{,}10 \end{bmatrix},$$

$$\begin{bmatrix} p(y_1,z_1) & p(y_2,z_1) \\ p(y_1,z_2) & p(y_2,z_2) \end{bmatrix} = \begin{bmatrix} 0{,}35 & 0{,}15 \\ 0{,}05 & 0{,}45 \end{bmatrix}.$$

Визначити, між якими джерелами статистичний зв'язок найбільший, а між якими найменший.

Розв'язання. Для відповіді на поставлене запитання потрібно знайти значення повної взаємної інформації для всіх пар джерел та порівняти їх. Найпростіше в такому разі користуватись виразом:

$$I(X, Y) = H(X) + H(Y) - H(X, Y).$$

Щоб обчислити безумовні ентропії кожного з джерел, знайдемо безумовні ймовірності появи символів на виході джерел за виразом (1.13):

$$p(x_1) = 0{,}7;\ p(x_2) = 0{,}2;\ p(x_3) = 0{,}1;$$
$$p(y_1) = 0{,}4;\ p(y_2) = 0{,}6;$$
$$p(z_1) = 0{,}5;\ p(z_2) = 0{,}5.$$

Потрібно зазначити, що значення кожної з ймовірностей можна отримати двома шляхами. Так $p(y_i)$, $i = 1,2$ є сумою елементів відповідних рядків першої матриці або елементів стовпців третьої матриці. Це означає, що матриці, наведені в завданні, відповідним чином узгоджені.

Розрахуємо ентропії джерел, користуючись (1.4):

$$H(X) = 1{,}157 \text{ біт} ; \; H(Y) = 0{,}971 \text{ біт} ; \; H(Z) = 1{,}0 \text{ біт} .$$

Далі відповідно до означення ентропії знаходимо сумісні ентропії:

$$H(UV) = -\sum_{i=1}^{N}\sum_{j=1}^{K} p(u_i v_j)\log p(u_i v_j)$$

$$H(X,Y) = 2{,}218 \text{ біт} ; \; H(X,Z) = 1{,}923 \text{ біт} ;$$

$$H(Y,Z) = 1{,}675 \text{ біт} .$$

Нарешті отримаємо:

$$I(X,Y) = 1{,}157 + 0{,}971 - 2{,}128 = 0 \text{ біт} ;$$

$$I(X,Z) = 1{,}157 + 1{,}0 - 1{,}923 = 0{,}234 \text{ біт} ;$$

$$I(Y,Z) = 0{,}971 + 1{,}0 - 1{,}675 = 0{,}296 \text{ біт} .$$

Те, що $I(X,Y)$ дорівнює нулю, означає, що джерела з алфавітами X та Y статистично незалежні. Найбільший статистичний зв'язок має місце між джерелами з алфавітами Y та Z, оскільки $I(Y,Z)$ має найбільше значення.

Приклад 1.11. Маємо три дискретних немарковських джерела з алфавітами $X = \{x_1, x_2\}$, $Y = \{y_1, y_2\}$, $Z = \{z_1, z_2\}$. Імовірності сумісної появи символів мають такі значення:

$$p(x1, y1, z1) = 0{,}048;$$
$$p(x1, y1, z2) = 0{,}012;$$
$$p(x1, y2, z1) = 0{,}072;$$
$$p(x1, y2, z2) = 0{,}162;$$
$$p(x2, y1, z1) = 0{,}272;$$
$$p(x2, y1, z2) = 0{,}068;$$
$$p(x2, y2, z1) = 0{,}300;$$
$$p(x2, y2, z2) = 0{,}060.$$

Знайти ентропії кожного з джерел, системи трьох джерел, а також повну взаємну інформацію для кожної пари джерел.

Розв'язання. Щоб знайти ентропію кожного з джерел, розрахуємо імовірності появи символів на виході джерел згідно з виразами:

$$p(x_1) = \sum_{k=1}^{2}\sum_{j=1}^{2} p(x_1, y_k, z_j) \; ; \quad p(x_2) = 1 - p(x_1) \; ;$$

$$p(y_1) = \sum_{i=1}^{2}\sum_{j=1}^{2} p(x_i, y_1, z_j) \ ; \qquad p(y_2) = 1 - p(y_1) \ ;$$

$$p(z_1) = \sum_{i=1}^{2}\sum_{k=1}^{2} p(x_i, y_k, z_1) \ ; \qquad p(z_2) = 1 - p(z_1) \ .$$

Маємо такі значення:

$$p(x1) = 0{,}3; \quad p(x2) = 0{,}7; \quad p(y1) = 0{,}4;$$
$$p(y2) = 0{,}6; \quad p(z1) = 0{,}692; \quad p(z2) = 0{,}308.$$

Знаходимо ентропії джерел:

$$H(X) = 1{,}8813 \text{ біт} \ ; \ H(Y) = 0{,}971 \text{ біт} \ ; \ H(Z) = 0{,}8909 \text{ біт} \ .$$

Щоб обчислити повну взаємну інформацію, знайдемо ентропії $H(X,Y), H(X,Z), H(Y,Z),$ для чого розрахуємо ймовірності сумісної появи пар символів, користуючись виразами типу:

$$p(x_i, y_k) = \sum_{j=1}^{2} p(x_i, y_k, z_j) \ .$$

Маємо

$$\begin{bmatrix} p(x_1, y_1) & p(x_2, y_1) \\ p(x_1, y_2) & p(x_2, y_2) \end{bmatrix} = \begin{bmatrix} 0{,}06 & 0{,}34 \\ 0{,}24 & 0{,}36 \end{bmatrix},$$

$$\begin{bmatrix} p(y_1, z_1) & p(y_2, z_1) \\ p(y_1, z_2) & p(y_2, z_2) \end{bmatrix} = \begin{bmatrix} 0{,}32 & 0{,}372 \\ 0{,}08 & 0{,}228 \end{bmatrix},$$

$$\begin{bmatrix} p(x_1, z_1) & p(x_2, z_1) \\ p(x_1, z_2) & p(x_2, z_2) \end{bmatrix} = \begin{bmatrix} 0{,}12 & 0{,}572 \\ 0{,}18 & 0{,}128 \end{bmatrix} \ .$$

Відповідно до означення обчислюємо ентропії кожної із систем двох джерел:

$$H(UV) = -\sum_{i=1}^{N}\sum_{j=1}^{K} p(u_i v_j) \log p(u_i v_j).$$

$$H(X,Y) = 1{,}7975 \text{ біт} \ ; \ H(X,Z) = 1{,}653 \text{ біт} \ ; \ H(Y,Z) = 1{,}8345 \text{ біт} \ .$$

Тепер знаходимо значення повної взаємної інформації для всіх пар джерел:

$$I(X,Y) = H(X) + H(Y) - H(X,Y) = 0{,}0548 \text{ біт} \ ,$$

Аналогічно

$$I(X,Z) = 0{,}1192 \text{ біт} \ ; \ I(Y,Z) = 0{,}0274 \text{ біт} \ .$$

Для обчислення ентропії системи трьох джерел скористаємось виразом (1.11):

$$H(X,Y,Z) = H(X) + H(Y/X) + H(Z/X,Y) \ .$$

Із виразу (1.9) отримаємо:

$$H(Y/X) = H(X,Y) - H(X) = 0{,}9162 \text{ біт} \ .$$

Щоб знайти H(Z/Y,X), необхідно розрахувати умовні ймовірності типу $p(z_i / x_j, y_k)$. Для цього скористаємось виразами:

$$p(z_1 / x_j, y_k) = \frac{p(x_j, y_k, z_1)}{p(x_j, y_k)} \ ; \ \ p(z_2 / x_j, y_k) = 1 - p(z_1 / x_j, y_k) \ .$$

Отримаємо

$$p(z_1 / x_1, y_1) = 0{,}8 \ ; \qquad p(z_2 / x_1, y_1) = 0{,}2 \ ;$$

$$p(z_1 / x_2, y_1) = 0{,}8 \ ; \qquad p(z_2 / x_2, y_1) = 0{,}2 \ ;$$

$$p(z_1 / x_1, y_2) = 0{,}3 \ ; \qquad p(z_2 / x_1, y_2) = 0{,}7 \ ;$$

$$p(z_1 / x_2, y_2) = 0{,}8333 \ ; \ \ p(z_2 / x_2, y_2) = 0{,}1667 \ .$$

Далі знаходимо частинні умовні ентропії:

$$H(Z / x_1, y_1) = -\sum_{k=1}^{2} p(z_k / x_1, y_1) \cdot \log_2 p(z_k / x_1, y_1) = 0{,}7219 \text{ біт},$$

аналогічно

$$H(Z / x_2, y_1) = 0{,}7219 \text{ біт} \ ,$$

$$H(Z / x_1, y_2) = 0{,}8813 \text{ біт} \ ,$$

$$H(Z / x_2, y_2) = 0{,}6501 \text{ біт} \ .$$

Тепер обчислимо повну умовну ентропію:

$$H(Z / X, Y) = -\sum_{k=1}^{2} \sum_{i=1}^{2} H(Z / x_i, y_k) \cdot p(x_i, y_k) = 0{,}7343 \text{ біт} \ .$$

Нарешті

$$H(X,Y,Z) = 0{,}8813 + 0{,}9162 + 0{,}7343 = 2{,}5316 \text{ біт} \ .$$

Приклад 1.12. Марковське дискретне джерело інформації з алфавітом $X = \{x_1, x_2, x_3\}$ та глибиною пам'яті $h = 1$ описується такою матрицею умовних ймовірностей $p(x_i / x_k)$ виникнення символу x_i за умови, що йому передував символ x_k:

$$\begin{bmatrix} p(x_1/x_1) & p(x_2/x_1) & p(x_3/x_1) \\ p(x_1/x_2) & p(x_2/x_2) & p(x_3/x_2) \\ p(x_1/x_3) & p(x_2/x_3) & p(x_3/x_3) \end{bmatrix} = \begin{bmatrix} 0{,}8 & 0{,}1 & 0{,}1 \\ 0{,}7 & 0{,}2 & 0{,}1 \\ 0{,}6 & 0{,}2 & 0{,}2 \end{bmatrix}.$$

Обчислити ентропію такого джерела.

Розв'язання. Щоб розрахувати ентропію марковського джерела, необхідно знати безумовні ймовірності $p(x_1), p(x_2), p(x_3)$ появи відповідних символів на виході джерела. Їх можна отримати, скориставшись рівняннями:

$$p(x_1) = p(x_1)\cdot p(x_1/x_1) + p(x_2)\cdot p(x_1/x_2) + p(x_3)\cdot p(x_1/x_3) \ ;$$
$$p(x_2) = p(x_1)\cdot p(x_2/x_1) + p(x_2)\cdot p(x_2/x_2) + p(x_3)\cdot p(x_2/x_3) \ ;$$
$$p(x_1) + p(x_2) + p(x_3) = 1 \ .$$

Підставивши сюди значення умовних імовірностей та дещо спростивши, будемо мати систему лінійних рівнянь:

$$-0{,}2p(x_1) + 0{,}7p(x_2) + 0{,}6p(x_3) = 0 \ ,$$
$$0{,}1p(x_1) - 0{,}8p(x_2) + 0{,}2p(x_3) = 0 \ ,$$
$$p(x_1) + p(x_2) + p(x_3) = 1 \ .$$

Розв'язання системи дає:

$$p(x_1) = 0{,}76543 \ ; p(x_2) = 0{,}12346 \ ; p(x_3) = 0{,}11111 \ .$$

Тепер можна обчислити частинні умовні ентропії для кожного стану джерела, а потім знайти ентропію марковського джерела як математичне сподівання вищезгаданих частинних умовних ентропій (див. вираз (1.22)). Кожна частинна умовна ентропія $H(X/x_i)$ – це ентропія розподілу умовних імовірностей, розташованих в одному з рядків матриці:

$$H(X/x_1) = -\sum_{k=1}^{3} p(x_k/x_1)\cdot \log_2 p(x_k/x_1) \approx 0{,}922 \text{ біт} \ ;$$
$$H(X/x_2) \approx 1{,}157 \text{ біт}; \qquad H(X/x_3) \approx 1{,}371 \text{ біт} \ .$$

Ентропія джерела така:

$$H(X) = \sum_{i=1}^{3} H(X/x_i)\cdot p(x_i) \approx 1{,}001 \text{ біт} \ .$$

Приклад 1.13. Маємо два дискретних джерела з алфавітами $X = \{x_1, x_2, ..., x_M\}$ та $Y = \{y_1, y_2, ..., y_N\}$. Перше джерело – марковське з глибиною пам'яті *h* = 1. Воно описується матрицею умовних імовірностей:

$$[p(x_n/x_i)],\ n = 1, 2, 3, ..., M;\ i = 1, 2, 3, ..., M$$

виникнення символу x_n за умови, що попереднім був символ x_i. Імовірність виникнення символу y_k на виході другого джерела залежить від того, який символ з'явився на виході першого джерела, тобто ці ймовірності задаються матрицею

$$[p(y_k/x_i)],\ k = 1, 2, 3, ..., N;\ i = 1, 2, 3, ..., M.$$

Визначити ентропію другого джерела та повну взаємну інформацію.

Зауважимо, що ця задача моделює ситуацію, коли вихід марковського джерела з глибиною пам'яті $h = 1$ подано на вхід стаціонарного дискретного каналу без пам'яті (див. рис. 1.2).

Розв'язання. Друге джерело буде в загальному випадку марковським, більш того, глибина пам'яті цього джерела може перевищувати одиницю. Отримати вираз для ентропії через імовірності появи символів y_k на виході другого джерела за цих умов досить важко, оскільки для цього необхідно знати не тільки безумовні ймовірності $p(y_k)$, але й умовні – $p(y_k / y_j), p(y_k / y_n y_j)$ тощо. Для вирішення цієї задачі краще скористатись виразом:

$$H(Y) = H(X) + H(Y / X) - H(X / Y).$$

Ентропія $H(X)$ для марковського джерела з глибиною пам'яті $h = 1$ знаходиться так, як в прикладі 12.

Умовна ентропія $H(Y / X)$ знаходиться аналогічно значенню другого члена у виразі (1.9) – значення імовірностей $p(x_i)$ та $p(y_k / x_i)$ є відомими. Щоб визначити умовну ентропію $H(X / Y)$, необхідно знати ймовірності $p(y_k)$ та $p(x_i / y_k)$. Перші можна отримати таким чином:

$$p(y_k) = \sum_{i=1}^{M} p(x_i) p(y_k / x_i)\ .$$

Для ймовірностей $p(y_k / x_i)$ маємо:

$$p(y_k / x_i) = \frac{p(y_k) \cdot p(x_i / y_k)}{p(x_i)}\ .$$

Таким чином, отримали всі компоненти, щоб розрахувати $H(Y)$.

Припустимо, що перше джерело має характеристики із задачі 12, потужність алфавіту другого джерела $M = 3$, а матриця умовних імовірностей така:

$$\begin{bmatrix} p(y_1/x_1) & p(y_2/x_1) & p(y_3/x_1) \\ p(y_1/x_2) & p(y_2/x_2) & p(y_3/x_2) \\ p(y_1/x_3) & p(y_2/x_3) & p(y_3/x_3) \end{bmatrix} = \begin{bmatrix} 0,7 & 0,15 & 0,15 \\ 0,2 & 0,6 & 0,2 \\ 0 & 0 & 1 \end{bmatrix}.$$

Послідовно виконуючи вищезгадані дії, отримаємо:

$$H(Y/X) = 1,073 \text{ біт};$$
$$p(y_1) = 0,56049;\ p(y_2) = 0,18889;\ p(y_3) = 0,25062;$$
$$\begin{bmatrix} p(x_1/y_1) & p(x_2/y_1) & p(x_3/y_1) \\ p(x_1/y_2) & p(x_2/y_2) & p(x_3/y_2) \\ p(x_1/y_3) & p(x_2/y_3) & p(x_3/y_3) \end{bmatrix} = \begin{bmatrix} 0,95595 & 0,04405 & 0 \\ 0,60784 & 0,39216 & 0 \\ 0,45813 & 0,09852 & 0,44335 \end{bmatrix};$$
$$H(X/Y) = 0,671 \text{ біт}.$$

Враховуючи, що $H(X) = 1,001$ біт, маємо $H(Y) = 1,403$ біт. Нарешті, повна взаємна інформація

$$I(X,Y) = H(X) - H(X/Y) = H(Y) - H(Y/X) = 0,33 \text{ біт}.$$

Приклад 1.14. Маємо два немарковських дискретних джерела інформації з алфавітами $X = \{x_1, x_2, x_3\}$, $Y = \{y_1, y_2, y_3\}$. Імовірності появи символів на виході першого джерела:

$$p(x_1) = 0,65;\ p(x_2) = 0,2;\ p(x_3) = 0,15.$$

Значення умовних ймовірностей $p(y_k/x_i)$ появи символу y_k на виході другого джерела за умови, що на виході першого з'явився символ x_i, є такими:

$$\begin{bmatrix} p(y_1/x_1) & p(y_2/x_1) & p(y_3/x_1) \\ p(y_1/x_2) & p(y_2/x_2) & p(y_3/x_2) \\ p(y_1/x_3) & p(y_2/x_3) & p(y_3/x_3) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

Розрахувати ентропії кожного з джерел, системи двох джерел та повну взаємну інформацію [12].

Розв'язання. Скористаємось формулою повної імовірності:

$$p(y_k) = \sum_{i=1}^{3} p(x_i) p(y_k/x_i)$$

для визначення імовірностей появи символів на виході джерела з алфавітом Y:

$$p(y_1) = 0,2;\ p(y_2) = 0,65;\ p(y_3) = 0,15.$$

Отримані значення збігаються із значеннями імовірностей $p(x_i)$. Висновок про це можна було зробити, аналізуючи матрицю в умові задачі.

Кожний рядок і кожний стовпець має у своєму складі по одній одиниці. Це свідчить, що між символами існує взаємно-однозначний зв'язок. Тому

$$H(X) = H(Y) = 1{,}28 \text{ біт.}$$

Для визначення H(X,Y) та I(X,Y) розрахуємо імовірності $p(x_i, y_k)$ сумісної появи символів x_i та y_k за виразом $p(x_i, y_k) = p(x_i) \cdot p(y_k / x_i) = p(y_k) \cdot p(x_i / y_k)$:

$$\begin{bmatrix} p(x_1,y_1) & p(x_2,y_1) & p(x_3,y_1) \\ p(x_1,y_2) & p(x_2,y_2) & p(x_3,y_2) \\ p(x_1,y_3) & p(x_2,y_3) & p(x_3,y_3) \end{bmatrix} = \begin{bmatrix} 0 & 0{,}2 & 0 \\ 0{,}65 & 0 & 0 \\ 0 & 0 & 0{,}15 \end{bmatrix}.$$

Тепер неважко пересвідчитись (зробіть це самостійно), що

$$H(X,Y) = I(X,Y) = H(X) = H(Y) = 1{,}28 \text{ біт}.$$

1.3 Контрольні питання і задачі

1. Дайте означення інформації.
2. Які відмінності понять інформація, повідомлення, сигнал?
3. Наведіть класифікацію повідомлень.
4. Охарактеризуйте кількісні оцінки інформації.
5. Визначте взаємозв’язок міри Шеннона і Хартлі.
6. Наведіть основні властивості ентропії.
7. Умовна ентропія і її властивості.
8. Кількість інформації як міра знятої невизначеності.
9. Охарактеризуйте модель системи передачі.
10. Наведіть відмінності кодування джерела інформації та кодування каналу.
11. Охарактеризуйте моделі джерел дискретних повідомлень, властивості ергодичних послідовностей знаків.
12. Що таке міра надлишковості?
13. Як визначається продуктивність джерела дискретних повідомлень?
14. Охарактеризуйте моделі дискретних каналів.
15. Як визначається швидкість передачі інформації по дискретному каналу та пропускна здатність дискретного каналу без завад і з завадами.

Задачі

1. Визначити ентропію повідомлення, ймовірності появи символів якого такі: $p(z_1)= p(z_2)= p(z_3)= p(z_4)=0{,}01$, $p(z_5)=0{,}96$.
   Відповідь: 0,322 біт.
2. Визначити ентропію повідомлення «aaaabbcddd». Відповідь: 1,84 біт.

3. Визначити ентропію двійкового джерела інформації, якщо імовірність появи «0» – p(0)=0,1, а ймовірність появи «1» – p(1)=0,9. Відповідь: 0,47 біт.
4. Визначити ентропію джерела інформації, якщо імовірність появи на виході комбінації «00_2» – p(00)=0,01, «11_2» – p(11)=0,81, «01_2» – p(01)=0,09, «10_2» – p(10)=0,09. Пояснити результат та його взаємозв'язок з вправою 3. Відповідь: 0,94 біт.
5. Для двійкового джерела визначити ймовірності появи «0» та «1», за яких ентропія максимальна. Відповідь: 0,5.
6. Визначити ентропію H(U), H(V), $H_V(U)$, H(UV), якщо задана матриця ймовірностей станів системи, яка об'єднує джерела U і V:

$$P(VU) = \begin{pmatrix} 0,3 & 0,1 & 0 \\ 0 & 0,2 & 0,2 \\ 0 & 0 & 0,2 \end{pmatrix}.$$

Відповідь: H(U) = 1,52 біт, H(V) = 1,57 біт, $H_V(U)$ = 0,67 біт, H(UV) = 2,24 біт.

7. Визначити, чи є ергодичним стаціонарне дискретне джерело повідомлень, алфавіт якого складається з 4-х знаків z_1, z_2, z_3, z_4 Причому, безумовні імовірності вибору знаків такі:

$$p(z_1) = p(z_2) = p(z_3) = p(z_4) = \frac{1}{4}.$$

А умовні ймовірності задані такою таблицею:

Таблиця 1.3 – Умовні імовірності вибору знаків

$z_q \backslash z_i$	z_1	z_2	z_3	z_4
z_1	1/2	1/4	1/4	0
z_2	1/4	1/4	1/2	0
z_3	1/4	1/2	1/4	0
z_4	0	0	0	1

Відповідь: неергодичне.

8. Оцінити, яку частку загальної кількості можливих послідовностей необхідно врахувати в практичних розрахунках, якщо ергодичне джерело характеризується такими параметрами: n =8 (обсяг алфавіту джерела), H(Z) = 2 біти (ентропія джерела), N = 100 (кількість знаків в послідовності). Відповідь: $1/2^{100}$.
9. Визначити надмірність повідомлення «aaaabbcddd» (вправа 2). Відповідь: 0,54.
10. Розробити програму для визначення частот символів у файлі. Мова програмування C++.

2 ЕФЕКТИВНЕ КОДУВАННЯ ІНФОРМАЦІЇ

2.1 Теоретичні положення

Нерівномірні коди. У нерівномірних кодах, на відміну від рівномірних, знаки повідомлення подаються кодами не обов'язково однакової довжини. Якщо деякі знаки повідомлення більш імовірні, порівняно з іншими, то знаки, які частіше зустрічаються, можна подати більш короткими кодовими комбінаціями. Тоді нерівномірні коди можуть бути суттєво ефективнішими порівняно з блоковими [2, 5].

Основною проблемою нерівномірних кодів є однозначність декодування, тобто як в двійковій системі відрізнити, де закінчується одне кодове слово і починається наступне.

Нехай маємо код: $z_1 = 0$, $z_2=10$, $z_3 =110$, $z_4 =111$. Для декодування такого коду приймач використовує дерево розв'язків (рис. 2.1).

Після прийому першої двійкової цифри автомат переходить з початкового стану в кінцевий стан z_1, якщо прийнято «0», або в другу точку розгалуження, якщо прийнято «1». Наступна прийнята двійкова цифра або переводить автомат в наступний кінцевий стан, або в наступну точку розгалуження. В кінцевому стані видається відповідний символ і автомат переходить в початковий стан, тобто кожний біт прийнятої послідовності переглядається лише один раз. В цьому прикладі декодування є миттєвим [5], оскільки у разі отримання всього кодового слова приймач зразу про це знає і йому немає необхідності досліджувати наступні біти, щоб прийняти рішення, який символ джерела прийнято.

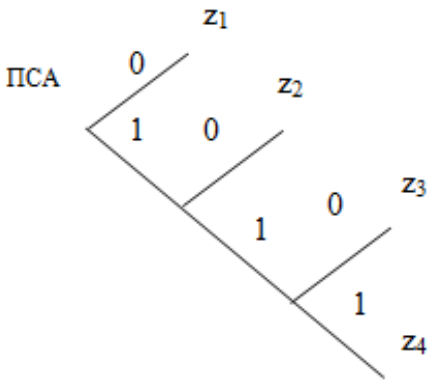

Рисунок 2.1 – Дерево декодування. ПСА – початковий стан автомата

Ніяке кодове слово цього коду не збігається з початковою частиною іншого кодового слова, тобто не є префіксом. Коди такого типу називаються миттєвими *(instantaneous code)* або префіксними *(prefix code)*. Крім того, оскільки на кінці кожного кодового слова є двійкова цифра «0», то розглянутий код називають ще «кодом з комою».

Розглянемо ще один код: $z_1 = 0$, $z_2=01$, $z_3 =011$, $z_4 =111$. Цей код не є миттєвим, оскільки деякі кодові слова є префіксами інших. Його можна декодувати, але декодують цей код з кінця, тобто приймач має очікувати останній знак повідомлення, що потребує великого обсягу пам'яті.

Тобто, для отримання миттєвого коду необхідно і достатньо, щоб ніяке кодове слово коду не було префіксом іншого кодового слова, а для немиттєвих кодів декодування деколи потребує необмеженого обсягу пам'яті [5].

Нерівність Крафта. Нерівність Крафта дає умову існування миттєвих кодів, але не вказує як їх будувати [5].

Теорема. Необхідною і достатньою умовою існування миттєвого коду для джерела з алфавітом Z із k символів z_i, де i = 1, 2, ... k, кодові слова якого мають довжини $l_1 \le l_2 \le l_3 \le \ldots \le l_k$, є виконання нерівності:

$$\sum_{i=1}^{k} \frac{1}{r^{l_i}} \le 1, \qquad (2.1)$$

де r – основа системи числення (коду).

Укорочені блокові коди. Розглянемо приклад, коли кількість символів джерела не дорівнює точному степені основи коду.

Якщо кількість символів джерела дорівнює 5, то для подання цих символів потрібно три розряди. Оскільки 3 розряди дозволяють передавати 8 різних комбінацій, то необхідно відкинути 3 комбінації. Для вирішення питання, які комбінації відкинути, побудуємо скінченний автомат для декодування цього коду (рис. 2.2).

Якщо відкинути 001, 011, 101, то можливо укоротити 3 гілки дерева, зберігши миттєве декодування (рис. 2.3, а).

Таким чином отримуємо $S_1=00$, $S_2=01$, $S_3=10$, $S_4=110$, $S_5=111$. Перевіряємо за нерівністю Крафта:

$$\frac{1}{2^2} + \frac{1}{2^2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^3} = 1.$$

Нерівність Крафта виконується, тому код є миттєвим. За умови, що всі символи рівноімовірні, середня довжина кодових комбінацій така:

$$l_{ср} = \sum_{i=1}^{5} p_i n_i = 0{,}2 \cdot 2 + 0{,}2 \cdot 2 + 0{,}2 \cdot 2 + 0{,}2 \cdot 3 + 0{,}2 \cdot 3 = 2{,}4 \text{ біт}.$$

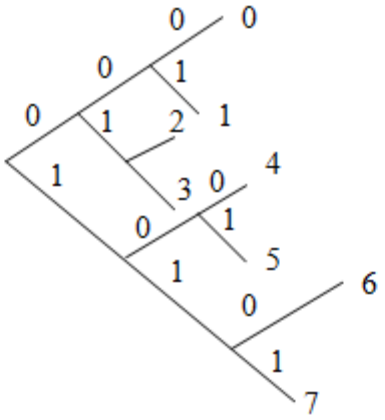

Рисунок 2.2 – Дерево декодування для трибітового рівномірного коду

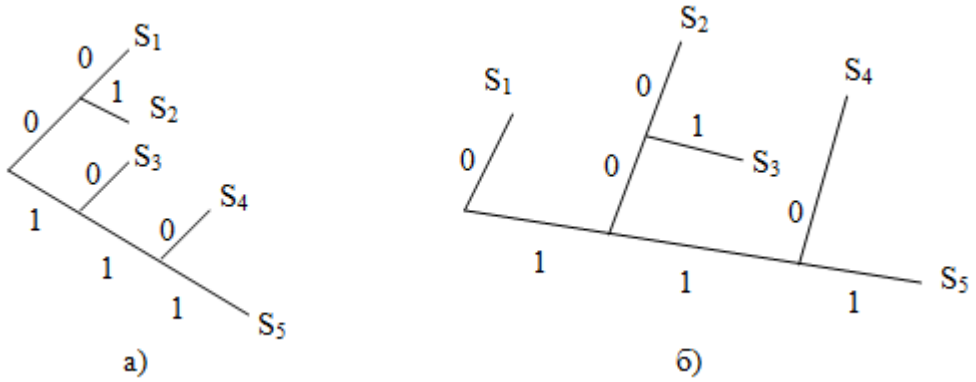

Рисунок 2.3 – Дерево декодування блокового укороченого коду

Розглянутий вище код називають блоковим укороченим кодом [5].

Можна відкинути 001, 010, 011 (рис. 2.3, б). Тоді S_1=0, S_2=100, S_3=101, S_4=110, S_5=111. Перевіряємо за нерівністю Крафта:

$$\frac{1}{2}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^3}=1.$$

Цей код миттєвий, його можна однозначно декодувати. За умови, що всі символи рівноімовірні, середня довжина кодових комбінацій така:

$$l_{ср}=\sum_{i=1}^{5}p_i n_i=0{,}2\cdot 1+0{,}2\cdot 3+0{,}2\cdot 3+0{,}2\cdot 3+0{,}2\cdot 3=2{,}6\ \text{біт}.$$

Вибір укороченого блокового коду залежить від статистики джерела інформації, оскільки розглядався тільки випадок рівноімовірного джерела інформації.

Основна теорема Шеннона для кодування каналу без завад. Теорему Шеннона для кодування каналу без завад можна сформулювати так [1 – 3]:

- повідомлення джерела з ентропією H(z) завжди можна закодувати послідовностями символів з обсягом алфавіту m так, що середнє число символів на знак повідомлення $l_{ср}$ буде як завгодно близьким до величини $\frac{H(z)}{\log m}$, але не менше за неї. За m=2, $l_{ср} \geq H(z)$.

З теореми видно, що кожний символ кодової комбінації має нести максимальну інформацію, тобто набувати значень або 0, або 1 якомога з однаковими ймовірностями і кожний вибір має бути незалежним. Хоча теорема не вказує конкретного способу побудови таких кодів.

Конструктивні методи побудови ефективних кодів для випадку відсутності взаємозв'язків між знаками розроблено американськими вченими Шенноном і Фано [2, 3].

Ефективне кодування некорельованої послідовності знаків методом Шеннона-Фано. Код будують так [3].

1. Знаки алфавіту повідомлення вписують в таблицю в порядку убування їх імовірностей.
2. Таблицю розділяють на 2 групи так, щоб суми імовірностей в обох групах були якомога однаковішими.
3. Всім знакам однієї групи приписують як перший символ 0(1), а знакам другої групи приписують символ 1(0).
4. Кожну з отриманих груп знову аналогічно розбивають на 2 підгрупи.
5. Процес повторюється доти, доки в кожній підгрупі не залишиться по одному знаку.

Методика Шеннона-Фано не завжди приводить до однозначної побудови коду, оскільки під час розбиття на підгрупи можна зробити більшою за імовірностями як одну, так і іншу підгрупу. Цих недоліків не має методика Гаффмана.

Типова класифікація методів оптимального кодування. Теорія економного або оптимального кодування (optimal coding) об'єднує в собі декілька різних напрямів: методи економного кодування інформації без втрат (lossless) і методи економного кодування інформації з втратами (lossy) [10, 15 – 17]. Методи першої групи не передбачають інформаційних втрат, тоді як методи другої групи пов'язані з такими втратами.

Основу багатьох систем ущільнення без втрат [15] формують дві схеми ущільнення – кодування Гаффмана (Huffman) і LZW-кодування (за початковими літерами прізвищ Лемпел (Lempel), Зів (Ziv) і Уелч (Welch)). Окремий клас утворюють методи арифметичного кодування. Таким чином, алгоритми ущільнення даних без втрат можна розділити на такі групи:

- статистичні методи ущільнення;
- словникові (евристичні) методи ущільнення;
- арифметичне кодування.

Статистичні алгоритми (Шеннона-Фано, Гаффмана, кодування за ступенем новизни, імовірнісне ущільнення та ін.) ґрунтуються на знанні імовірностей або частот появи символів в повідомленні [15]. Оскільки апріорі ці імовірності невідомі, то статистичні алгоритми можна поділити на такі класи.

1. Неадаптивні – використовують фіксовані, завчасно задані імовірності появи символів у повідомленні, тому таблиця імовірностей символів не передається разом з файлом. Недолік: невеликий перелік файлів, для яких досягається прийнятний коефіцієнт стиснення.
2. Напівадаптивні – для кожного файлу будується таблиця частот (імовірностей) символів, яку і використовують для ущільнення файлів. Напівадаптивні алгоритми непогано ущільнюють більшість файлів, але додатково передається таблиця частот символів, а також необхідно два проходи початкового файлу для кодування.
3. Адаптивні – починають працювати з фіксованою початковою таблицею частот (імовірностей) символів (як правило, на початку роботи всі символи рівноімовірні), в процесі роботи ця таблиця змінюється залежно від символів, що зустрічаються у файлі. Переваги: однопрохідність алгоритму, таблиця частот (імовірностей) символів не передається, ефективно ущільнюють широкий клас файлів.

Евристичні (словникові) алгоритми ущільнення (LZ77, LZ78 та ін.) шукають в файлі рядки символів, що повторюються, і будують словник фраз, що вже зустрічались. Ці алгоритми мають цілий ряд специфічних параметрів (розмір буфера, максимальна довжина фрази і т. п.), підбір яких залежить від досвіду автора роботи. Такі параметри добираються так, щоб досягти оптимального співвідношення часу роботи алгоритму, коефіцієнта ущільнення та переліку файлів, що добре ущільнюються [15].

Арифметичне кодування, як і статистичні методи, використовує імовірність появи символів у повідомленні, однак процес арифметичного кодування має принципові відмінності. Внаслідок арифметичного кодування символьна послідовність (рядок) замінюється дійсним числом більше нуля і менше одиниці [15].

Статистичні алгоритми ущільнення даних без втрат

Алгоритм RLE. Алгоритм RLE *(Run Length Encoding)* або групове кодування, один з найбільш давніх алгоритмів кодування графіки, дуже простий в реалізації. Символи, що повторюються, замінюються на один

цей символ та лічильник повторень. Для того, щоб архіватор в процесі відновлення міг відрізнити в результувальному потоці кодовану серію від інших символів потрібно помістити у всі ланцюжки деякі заголовки (наприклад, використовувати перший біт як ознаку кодування серії). Метод достатньо ефективний для графічних зображень в форматі «байт *(byte)* на піксел» (формат PCX використовує кодування RLE) .

Недоліки алгоритму: мала пристосованість до багатьох типів файлів, тому його можна ефективно використовувати тільки в поєднанні зі вторинним кодуванням.

Наприклад, в алгоритмі кодування факсів спочатку зображення розбивається на чорні та білі точки, які перетворюються алгоритмом RLE в потік довжин серій, а потім ці довжини серій кодуються методом Гаффмана зі спеціально підібраним (експериментально) деревом [15].

Кодування Гаффмана

Девід Гаффман запропонував будувати кодове дерево не коренем вниз, як це неявно робиться в алгоритмі Шеннона-Фано, а коренем верх – від листкових вузлів до кореневого вузла. Такий підхід забезпечує однозначність побудови коду.

На початку роботи алгоритму кожному символу повідомлення ставиться у відповідність вага, що дорівнює імовірності (частоті) появи цього символу у повідомленні, список символів сортується за спаданням ваг (рис. 2.4).

На кожному кроці (ітерації) два останні елементи списку об'єднуються в новий елемент з вагою, що дорівнює сумі ваг цих елементів. Новий елемент поміщається в список замість двох об'єднуваних елементів і новий список, який завжди містить на один елемент менше, сортується за спаданням ваг.

Паралельно здійснюється побудова кодового дерева або дерева Гаффмана. На кожному кроці алгоритму кореневий вузол бінарного дерева відповідає новому елементу, що поміщається в список, а елементам списку, що заміщаються, відповідають дочірні вузли цього кореневого вузла. Коли в списку залишається один елемент, відповідний кореневому вузлу побудованого бінарного дерева, алгоритм завершує роботу.

Префіксні коди отримують шляхом присвоєння конкретних двійкових значень ребрам дерева Гаффмана.

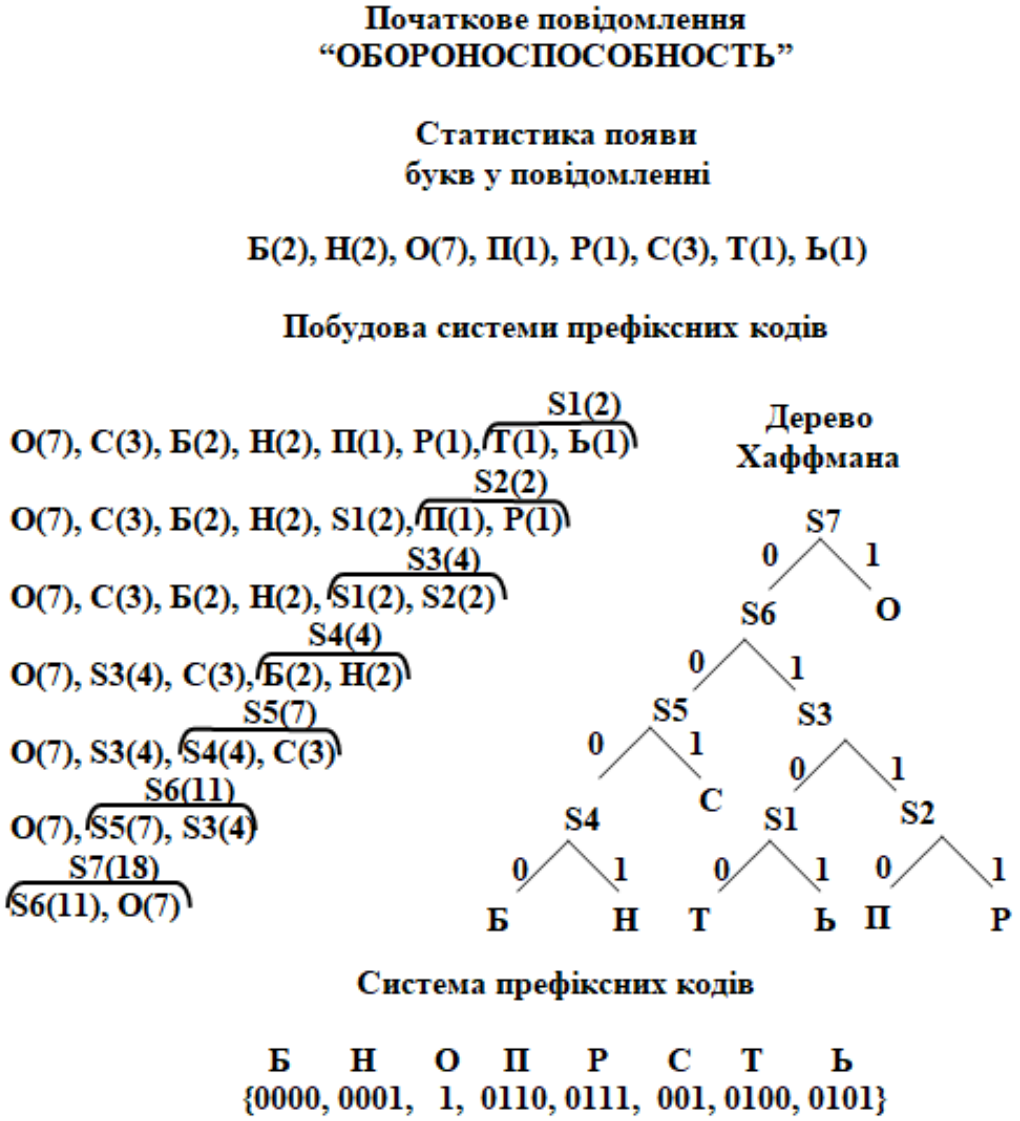

Рисунок 2.4 – Ілюстрація роботи алгоритму Хаффмана

Префіксні коди, отримані внаслідок побудови дерева Гаффмана, є оптимальними [16 – 17]. Однак це не означає, що префіксні коди, отримані з використанням інших алгоритмів, є менш ефективними.

Недоліком класичного (напівадаптивного) кодування Гаффмана є необхідність додаткової передачі або зберігання таблиці частот символів, а також необхідно два проходи початкового файлу для кодування. Однак, існують адаптивні версії побудови дерева Гаффмана. Коди, які при цьому отримують є квазіоптимальними, тому коефіцієнт ущільнення трохи гірший. Зростає також складність алгоритму і час роботи (хоча він і стає однопрохідним) [17].

Кодування за ступенем новизни

Відомий також як MTF *(Move To Front – рухай нагору)* або кодування за допомогою купки книжок [2, 18] є одним із варіантів адаптивного алгоритму Гаффмана. Кодер зберігає список символів алфавіту джерела.

Коли на вхід надходить черговий символ С повідомлення, кодер передає його номер «і» в списку символів алфавіту і переставляє символ С в цьому списку на початок, збільшуючи на одиницю номери всіх символів, які стоять перед С. Таким чином, символи, що часто зустрічаються, будуть розміщуватись ближче до початку списку і їх можна буде закодувати більш короткими кодами.

Як код для кодування за ступенем новизни можна використовувати різні нерівномірні миттєві коди. Якщо необхідно передати слово w=221312233 в алфавіті А={1,2,3}. Тоді кодом позиції 1 є 0, позиції 2 – 10, позиції 3 – 11. Купка книжок за послідовної появи літер слова w змінюється таким чином:

(1,2,3) – (2,1,3) – (2,1,3) – (1,2,3) – (3,1,2) і т. д.

Кодом слова буде

100101110110…

Кодування MTF поступається кодуванню Гаффмана під час ущільнення текстових файлів, однак для файлів з малим початковим алфавітом можна досягти гарних результатів. Кодування за ступенем новизни легко пристосовується до можливих коливань частот, тому він названий ще локально-адаптивним, ідеально підходить до сумісного використання разом з BWT-перетворенням, яке розміщує однакові символи блока, що перетворюється, один за одним.

Імовірнісне кодування

Працює алгоритм так: є досить велика таблиця передбачень, що досить швидко оновлюється, в якій для кожної можливої пари послідовних вхідних символів вказується передбачений наступний (третій) символ. Якщо символ передбачений правильно – генерується код у вигляді однобітного префікса, що дорівнює 1. Якщо ж символ не вгадано – видається код у вигляді префікса, що дорівнює 0 і невгаданого символу, який замінює в таблиці символ, що був там раніше, забезпечуючи оновлення статистичної інформації [19].

Алгоритм є адаптивним, однопрохідним, тому не потребує збереження таблиці разом із закодованим текстом.

Декодувальник працює за аналогічним алгоритмом, підтримуючи й оновлюючи таблицю синхронно з кодувальником. Якщо надійшов префікс 1, черговий вихідний символ береться з таблиці за індексом, отриманим з двох попередніх символів, інакше копіюється код із вхідного потоку у вихідний. Для підвищення ступеня ущільнення рекомендується ініціалізувати таблицю перед початком роботи будь-якими сполученнями, які часто зустрічаються.

Подібний алгоритм використовується в архіваторі PKZIP Ver.1.5 під ім'ям Reducing. Хоча він за всіма параметрами гірший порівняно з імовірнісним (двопрохідний, потребує передачі таблиці разом з текстом), проте заслуговує на згадку.

Словникові методи оптимального кодування

Ланцюжки подій поміщаються в словник. Якщо у вхідному потоці зустрічається ланцюжок подій, який присутній в словнику, то замість цього ланцюжка у вихідний потік поміщається посилання на елемент словника. Ідея словникових алгоритмів була запропонована Я. Зівом і А. Лемпелем в 1977 р. [16] і була втілена в алгоритмі LZ77, що став основою цілого сімейства алгоритмів. Інший підхід, який вони запропонували у 1978 р., породив сімейство алгоритмів LZ78.

Словникові алгоритми забезпечують не найвищі коефіцієнти ущільнення, але швидкість роботи достатньо висока, особливо під час декодування даних, що і обумовлює їх широке використання.

Словник в алгоритмах сімейства LZ77 – це деяка кількість попередніх подій. Посилання на елементи словника – це відстань до ланцюжка подій, що зустрічалися раніше, і довжина цього ланцюжка [2]. Для забезпечення відмінності між ланцюжками подій і літеральними подіями (подіями, що не увійшли до ланцюжків із словника) додатково вводиться прапорець-біт (алгоритм LZSS).

Алгоритм LZH аналогічний LZSS, але використовує для покажчиків кодування Гаффмана.

Алгоритми сімейства LZ77 широко використовуються в програмах універсальних архіваторів, як правило, це модифікації LZSS з використанням кодів Гаффмана (LHARC) Шеннона-Фано (PKZIP) або арифметичного кодування (HA).

У алгоритмах сімейства LZ78 ланцюжки у словнику збільшуються на один символ кожного разу, коли ланцюжок із словника зустрічається у вхідному потоці.

У 1984 р. Т. Велч запропонував алгоритм LZW, у якому всі літеральні події є елементами словника [15]. У вихідний потік поміщаються тільки посилання на елементи словника, що спрощує алгоритм. Велч показав, що алгоритм може бути легко реалізований апаратно. Програмна реалізація, як правило, виконується на основі вектора двійкових дерев. Корінням дерев слугують елементи вхідного алфавіту.

LZC – модифікація алгоритму LZW, направлена на підвищення коефіцієнта ущільнення. Для цього використовується змінна довжина кодів, із заповненням словника довжина кодів збільшується. Крім того, відстежується ступінь ущільнення: якщо ступінь ущільнення спадає нижче деякої межі, словник очищається, що забезпечує швидку адаптацію алгоритму до різкої зміни характеру даних.

LZC використовується в утиліті COMPRESS операційної системи

UNIX і графічному форматі GIF.

Алгоритм LZFG є гібридом LZ77 і LZ78. Як і LZ77 він допускає кодування ланцюжків довільної довжини (у алгоритмах LZ78 довжина ланцюжків, як правило, обмежена кількістю попередніх поява ланцюжка у вхідному потоці). Однак адресація елементів словника аналогічна техніці LZ78, хоча замість двійкового дерева використовується дерево Patricia. Відмінність дерева Patricia полягає в тому, що якщо дерево не містить розгалужень, то послідовність вершин об'єднується в одну.

Алгоритм RFGD є модифікацією LZFG з вищим ступенем ущільнення [13]. У вихідний потік поміщуються дані чотирьох типів: термінальні вершини дерева, нетермінальні вершини дерева, літерали, що зустрічалися раніше у вхідному потоці, і літерали, що раніше не зустрічалися. Кожний тип кодується за власною схемою. Для визначення типу даних, що поміщаються у вихідний потік, перед ними поміщається спеціальний префікс. Алгоритм реалізований апаратно.

Алгоритм LZW

LZW-стиснення заміняє рядки символів кодами без будь-якого аналізу вхідного тексту. За додання кожного нового рядка проглядається таблиця рядків, ущільнення відбувається, коли код заміняє рядок символів. Коди, що генеруються LZW-алгоритмом, можуть бути будь-якої довжини, але вони містять більше бітів, ніж одиничний символ. За замовчуванням перші 256 кодів (якщо використовуються 8-бітові символи) відповідають стандартному набору символів. Наступні коди відповідають рядкам, що обробляються алгоритмом.

LZW маніпулює трьома об`єктами: потоком символів, потоком кодів і таблицею рядків. Під час кодування потік символів є вхідним і потік кодів – вихідним. Під час декодування вхідним є потік кодів, а потік символів – вихідним. Таблиця рядків породжується і під час кодування, і під час декодування, але вона не передається від кодування до декодування.

Доцільно використовувати коди різної довжини, оскільки розмір файлу невідомий наперед.

Ущільнення інформації

Коли генерується новий код, новий рядок додається в таблицю рядків. Процедуру LZW-кодування наведено на рис. 2.5, а демонстрацію роботи алгоритму – на рис. 2.6.

Робота починається з того, що на першому кроці циклу виконується перевірка на наявність рядка «/W» в таблиці. Якщо цей рядок відсутній в таблиці, то генерується код для «/» , а в таблицю додається рядок «/W» і цьому рядку ставиться у відповідність код 256. Після цього система читає наступну букву «E» і з другим рядком «WE» виконує аналогічні дії. Коли знову з’явиться рядок «/W», який уже є в таблиці, то система виведе код 256 і додасть трисимвольний рядок «/WE» в таблицю рядків. Цей процес

продовжується до тих пір, поки не закінчиться вхідний потік і всі коди не будуть виведені [2].

Процедура LZW-стиснення:
РЯДОК = черговий символ з вхідного потоку
WHILE вхідний потік не пустий DO
СИМВОЛ = черговий символ з вхідного потоку
IF РЯДОК + СИМВОЛ в таблиці рядків THEN
РЯДОК = РЯДОК + СИМВОЛ
ELSE
вивести в вихідний потік код для РЯДОК
додати в таблицю рядків РЯДОК + СИМВОЛ
РЯДОК = СИМВОЛ
END of IF
END of WHILE
вивести в вихідний потік код для РЯДОК

Рисунок 2.5 – Алгоритм ущільнення методом LZW

Вхідний рядок: /WED/WE/ WEE/ WEB/WET

Вхід (символи)	Вихід (коди)	Нові коди і відповідні рядки
/W	/	256 = /W
E	W	257 = WE
D	E	258 = ED
/	D	259 = D/
WE	256	260 = /WE
/	E	261 = E/
WEE	260	262 = /WEE
/W	261	263 = E/W
EB	257	264 = WEB
/	B	265 = B/
WET	260	266 = /WET
<EOF>	T	

Рисунок 2.6 – Процес ущільнення

Відновлення інформації

Таблиця рядків під час декодування може бути точно відновлена на основі вихідного потоку алгоритму кодування, тому не потрібно її зберігати або передавати. Аналогічно до алгоритму кодування під час декодування новий рядок додається в таблицю рядків кожний раз, коли з вхідного потоку надходить новий код (рис. 2.7), необхідно лише додатково перевести кожний вхідний код в рядок і переслати його у вихідний потік.

Процедура LZW - декодування:
читати СТАРИЙ_КОД
вивести СТАРИЙ_КОД
WHILE вхідний потік не пустий DO
читати НОВИЙ_КОД
РЯДОК = перевести НОВИЙ_КОД
вивести РЯДОК
СИМВОЛ = перший символ РЯДКА
добавити в таблицю рядків СТАРИЙ_КОД + СИМВОЛ
СТАРИЙ_КОД = НОВИЙ_КОД
END of WHILE

Рисунок 2.7 – Алгоритм декодування

Роботу цього алгоритму на основі ущільнених даних, отриманих раніше, демонструє рис. 2.8.

Вихідний потік точно відповідає вхідному потоку алгоритму ущільнення. Перші 256 кодів виділені для переведення одиничних символів.

Вхідні коди: / W E D 256 E 260 261 257 B 260 T

Вхід Новий код	Старий код	Рядок Вихід	Символ	Новий вхід Таблиці
/	/	/		
W	/	W	W	256 = /W
E	W	E	E	257 = WE
D	E	D	D	258 = ED
256	D	/W	/	259 = D/
E	256	E	E	260 = /WE
260	E	/WE	/	261 = E/
261	260	E/	E	262 = /WEE
257	261	WE	W	263 = E/W
B	257	B	B	264 = WEB
260	B	/WE	/	265 = B/
T	260	T	T	266 = /WET

Рисунок 2.8 – Процес декодування

Арифметичне кодування. Арифметичне кодування дозволяє кодувати події кількістю біт, як завгодно близькою до величини $-\log_2 p$ [17], де p – імовірністю появи подій.

Нехай в кожен момент може настати подія A, B або C. Перші 3 кроки роботи алгоритму під час надходження на вхід потоку подій «B», «A», «B» наведено на 4 діаграмах рисунка 2.9. Перша діаграма відображає поділ інтервалу (0,1) на початкові підінтервали, пропорційні імовірностям появи цих подій до початку роботи алгоритму. Після надходження на вхід кодера події «B» робочий інтервал (X,Y) знову ділиться пропорційно імовірностям появи цих подій (діаграма 2) і т. д. Діаграма 4 на рис. 2.9 відповідає заштрихованій ділянці діаграми 1. У міру кодування подій вхідного потоку робочий інтервал (X,Y), (X’,Y’), (X’’,Y’’) ... скорочується. Вихідними даними алгоритму може слугувати будь-яке число, що входить в робочий інтервал після обробки останньої події вхідного потоку, однак на практиці під час кожного скорочення інтервалу у вихідний потік поміщаються всі збіжні старші біти верхньої і нижньої межі робочого інтервалу. У подій з більшою імовірністю початковий інтервал ширший, тому робочий інтервал скоротиться менше і менше бітів потрапить у вихідний потік. Тобто, події з більшою імовірністю появи кодуються меншою кількістю бітів.

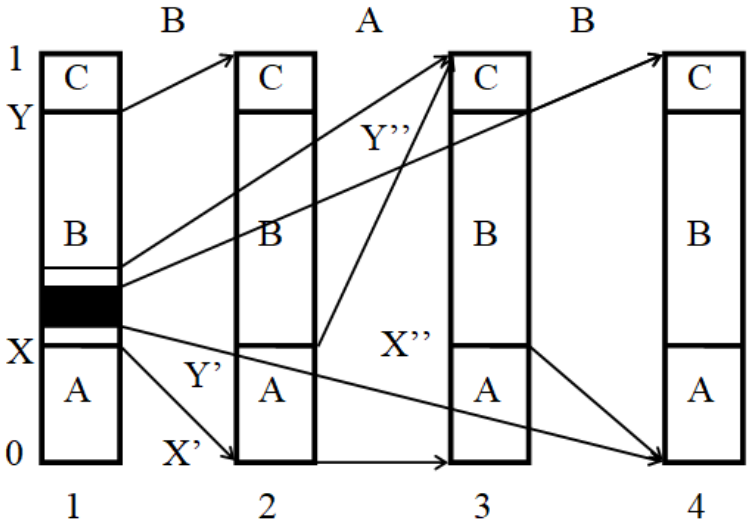

Рисунок 2.9 – Схема роботи алгоритму арифметичного кодування

Недоліки: висока обчислювальна складність [2]. З кожнии бітом вихідного потоку необхідно виконувати ряд операцій порівняння і зсуву, а для кожної кодованої події необхідно виконати ряд операцій множення і ділення.

Швидкою альтернативою арифметичному кодуванню є квазіарифметичне [17] кодування. Під час квазіарифметичного кодування операції множення і ділення замінені операціями вибору значення з таблиці. Алгоритм реалізовано у вигляді кінцевого цифрового автомата, який емулює роботу арифметичного кодера. За надходження на вхід

кодованої події автомат, за необхідності, поміщає у вихідний потік деякі біти і переходить в деякий новий стан. До недоліків квазіарифметичного кодера можна віднести низьку точність або надмірно великий обсяг таблиць.

Ще однією альтернативою арифметичному кодеру є Range-кодер [17]. Відмінністю його від арифметичного кодера є не побітне, а побайтне порівняння верхньої і нижньої границь і побайтне виведення закодованого результату у вихідний потік, що дозволяє підвищити швидкодію кодера. Але точність (якість кодування) у окремих реалізацій Range-кодерів є навіть вищою порівняно з класичним арифметичним кодером (за використання 32-бітової цілочислової арифметики).

Приклад роботи Range-кодера

Нехай необхідно закодувати слово «REDUNDANCE» (надмірність) [15]. Символи E, D і N зустрічаються двічі і імовірність їхньої появи дорівнює 0,2; імовірність появи інших символів в цьому слові дорівнює 0,1. Виконується поділ інтервалу (0,1) на початкові підінтервали, довжини яких пропорційні імовірностям появи цих символів (табл. 2.1).

Перша буква слова – «R» одержує інтервал з нижньою границею 0,8 і з верхньою – 0,9 (табл. 2.1). Нижня границя інтервалу і стає першою значущою цифрою коду. Далі підінтервал (0,8; 0,9) знову поділяється на підінтервали пропорційно імовірностям символів, тобто виконується класичне арифметичне кодування, але вхідними даними є символи (байти). Якщо цей процес формалізувати, то далі можна виконувати розрахунок границь підінтервалів для кожного наступного символу з вихідного потоку за такими виразами:

$$\left.\begin{aligned} \beta_n^l &= \beta_{n-1}^l + (\beta_{n-1}^h - \beta_{n-1}^l)P_n^l \\ \beta_n^h &= \beta_{n-1}^l + (\beta_{n-1}^h - \beta_{n-1}^l)P_n^h \end{aligned}\right\}, \qquad (2.2)$$

де β^l, β^h – нижня і верхня границі кодового інтервалу;

P^l і P^h – нижня і верхня границі інтервалу імовірності для символу.

Процес кодування наведено в табл. 2.2. Послідовність символів «REDUNDANCE» заміняється числом 0,8478570048, тобто замість 10 байт, необхідних для збереження символьного рядка, буде потрібно всього 4 байти для запису числа.

Арифметичне кодування дозволяє забезпечити високий ступінь ущільнення даних, особливо у випадках, коли зустрічаються дані, де частота появи різних символів дуже відрізняється одна від одної.

Таблиця 2.1 – Інтервали імовірності для символів в слові REDUNDANCE

Символ	Імовірність (p)	Інтервал
A	0,1	0,0-0,1
C	0,1	0,1-0,2
D	0,2	0,2-0,4
E	0.2	0,4-0,6
N	0,2	0,6-0,8
R	0,1	0,8-0,9
U	0,1	0,9-1,0

Таблиця 2.2 – Покрокове подання рядка REDUNDANCE методом арифметичного кодування

Символ	Нижня границя (β^l)	Верхня границя (β^h)
R	0,8	0,9
E	0,84	0,86
D	0,844	0,848
U	0,8476	0,848
N	0,84784	0,84792
D	0,847856	0,847872
A	0,8478560	0,8478576
N	0,84785696	0,84785728
C	0,847856992	0,847857024
E	0,8478570048	0,8478570432

Сучасні тенденції розвитку ущільнення даних

З появою методу арифметичного кодування основна увага приділяється питанню моделювання. Нова схема ущільнення за допомогою універсального моделювання і кодування *(universal modelling and coding)* запропонована Ріссаненом і Ленгдоном (Langdon) в 1981 р [13]. Процес ущільнення складається з двох самостійних частин: моделювання; кодування. Задача моделювання полягає у побудові моделі інформаційного джерела, а задача кодування – відобразити дані у стислу форму подання на підставі результатів моделювання (рис. 2.10).

Необхідно розрізняти поняття кодування в широкому сенсі (весь процес, включно й моделювання) і у вузькому (генерація потоку кодів на підставі моделі). Втрачає сенс і поняття «статистичне кодування», тому щоб уникнути плутанини ряд авторів застосовує термін «ентропійне кодування» для кодування у вузькому сенсі, хоча цей термін також обґрунтовано критикують. Тому далі в цьому розділі для процесу

кодування в широкому сенсі використовується термін «кодування», а у вузькому сенсі – «статистичне кодування» або «власне кодування».

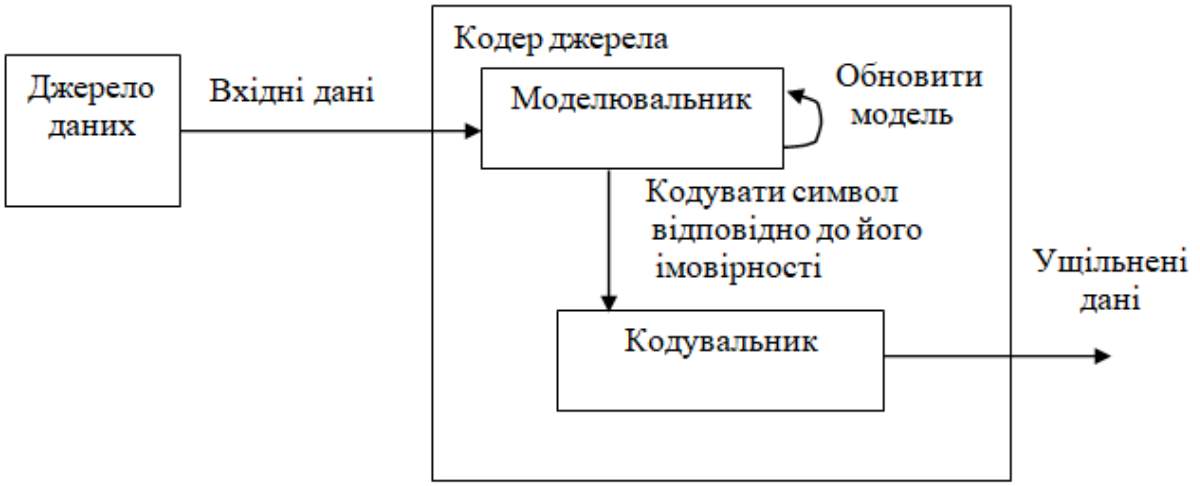

Рисунок 2.10 – Нова схема ущільнення даних

Моделювальник передбачає імовірність появи кожного символу в кожній позиції вхідного рядка, звідси ще одне найменування цього компонента – «передбачувач» або «предиктор» (від predictor). Чим точніша оцінка імовірності появи символів, тим більше коди відповідають оптимальним, тим краще ущільнення. На етапі статистичного кодування або власне кодування виконується заміщення символу s_i, з оцінкою імовірності появи $p(s_i)$ кодом завдовжки $\log_2 p(s_i)$ бітів.

Виділяють 4 варіанти моделювання:

- статичне;
- напівадаптивне;
- адаптивне (динамічне);
- блоково-адаптивне.

При статичному моделюванні використовується одна і та сама модель для всіх типів даних. Опис моделі міститься в структурах кодера і декодера.

Недолік: погане ущільнення або навіть збільшення даних, що не відповідають заданій моделі.

Напівадаптивне моделювання забезпечує простий спосіб оцінювання імовірностей за рахунок попереднього підрахунку частот появ символів в ущільнюваному блоці даних.

Недолік: необхідні два проходи та передача або зберігання таблиці частот або імовірностей символів.

В процесі адаптивного моделювання модель змінюється за заданим алгоритмом після надходження кожного символу. Однозначність декодування досягається тим, що кодер і декодер мають ідентичну початкову модель, модифікація якої під час кодування і декодування

виконується однаково. Коефіцієнти ущільнення за адаптивного моделювання можуть досягати значень близьких до напівадаптивного.

Під час блоково-адаптивного моделювання модель обновлюється не після надходження кожного символу, а після обробки блока символів. Довжина блока в процесі кодування може змінюватись.

Аналіз поширених типів даних повів, що більшість джерел повідомлень є джерелами з пам'яттю [13]. Моделювання, яке дає оцінку імовірності появи символу залежно від попередніх або контексту, називається контекстним моделюванням.

В широкому значенні контекст – це сукупність символів, що оточують поточний символ. Лівосторонній контекст – послідовність символів, що примикають до поточного символу зліва, правосторонній контекст – послідовность символів, що примикають до поточного символу справа. За контекстного моделювання терміном «контекст» позначають саме лівосторонній контекст, оскільки контекстне моделювання практично завжди застосовується як адаптивне.

На практиці довжина контексту завжди обмежена, тому говорять про контекстне моделювання обмеженого порядку (finite context modeling), порядок контексту – максимальна довжина N контекстів, що використовуються.

Серед методів контекстного моделювання найбільш відома техніка PPM (Prediction by Partial Matching) – передбачення за частковим збігом та її модифікації, яка за прийнятних обчислювальних затрат забезпечує ущільнення в 3-4 рази для текстів і в 2-3 рази – для об'єктних файлів.

Деякі інші відомі методи контекстного моделювання такі:

- моделі станів (використовується за динамічного марковського ущільнення – Dinamic Markov Compression або DMC);
- граматичні моделі (використовуються в алгоритмі SEQUITUR);
- моделі з використанням штучних нейронних мереж для побудови передбачувача [13].

Детальний розгляд методів контекстного моделювання виходить за межі цього навчального посібника.

Концепція універсального моделювання і кодування дала поштовх розвитку методів ущільнення без втрат на основі перетворень, які призначені для перетворення одних видів надмірності в інші, простіше модельовані. Тут наявні фактично два етапи моделювання: перший етап – це робота перетворення, направлена на отримання простішої інформаційної моделі, а другий – побудова допоміжної моделі для кодувальника.

Найбільш відомими є перетворення MTF (див. вище) та перетворення BWT (Burrows-Wheeler Transform) [13]. MTF давно використовується під час ущільнення і як перетворення, і як самостійний метод ущільнення.

BWT-перетворення може використовуватись тільки як перетворення. Вихідний потік перетворення BWT кодується за методом RLE або додатково перетворюється за методом MTF і далі кодується або префіксним, або арифметичним кодером.

BWT-перетворення

Розглянемо перетворення рядка символів «абракадабра». Створимо матрицю всіх можливих його циклічних перестановок (рис. 2.11). Для створення матриці циклічних перестановок запишемо символьний рядок «абракадабра» в «0» рядок матриці (рис. 2.11) і кожний символ продублюємо в інші рядки матриці в позиції, які визначаються так:

$$Pos=(rbwt - ja + ia) \bmod rbwt,$$

де rbwt – розмір блока BWT-перетворення;

ia – номер поточного стовпця (позиція символу в рядку);

ja – номер рядка, в який записується черговий символ (байт).

Наприклад, для рис. 2.11 – rbwt=11, нехай ja=0 і ia=6, і нехай ja=7 і ia=6, тоді:

0-рядок: Pos=(11 –0 +6) mod 11=6, тобто буква «д»;
7-рядок: Pos=(11 –7 +6) mod 11=10, тобто буква «д».

0 абракадабра
1 бракадабраа
2 ракадабрааб
3 акадабраабр
4 кадабраабра
5 адабраабрак
6 дабраабрака
7 абраабракад
8 браабракада
9 раабракадаб
10 аабракадабр

Рисунок 2.11 – Матриця циклічних перестановок рядка «абракадабра»

Відсортуємо всі рядки цієї матриці у лексикографічному порядку символів (рис. 2.12).

Тепер потрібно виписати символи останнього стовпця і запам’ятати номер початкового рядка серед відсортованих, тобто, «рдакрааааабб», 2 – це результат BWT-перетворення.

0 аабракадабр
1 абраабракад
2 абракадабра - початковий рядок
3 адабраабрак
4 акадабраабр
5 браабракада
6 бракадабраа
7 дабраабрака
8 кадабраабра
9 раабракадаб
10 ракадабрааб

Рисунок 2.12 – Матриця циклічних перестановок рядка «абракадабра», відсортована зліва направо відповідно до лексикографічного порядку символів

Для виконання зворотного перетворення рядок «рдакрааааабб» запишемо як стовпець і відсортуємо його (рис. 2.13) у лексикографічному порядку та внаслідок сортування отримаємо перший стовпець початкової матриці відсортованої матриці (рис. 2.12). Додамо рядок «рдакрааааабб» як останній стовпець в отриману матрицю (рис. 2.14). Оскільки рядки матриці отримано внаслідок циклічного зсуву початкового рядка, то символи останнього і першого стовпців утворюють один з одним пари. Відсортуємо ці пари і допишемо в матрицю відомий нам останній стовпець (рис. 2.15).

0 а
1 а
2 а
3 а
4 а
5 б
6 б
7 д
8 к
9 р
10 р

Рисунок 2.13 – Відсортовані символи початкового рядка

0 а...р
1 а...д
2 а...а
3 а...к
4 а...р
5 б...а
6 б...а
7 д...а
8 к...а
9 р...б
10 р...б

Рисунок 2.14 – Перший і останній стовпці матриці циклічних перестановок

0	аа…р
1	аб…д
2	аб…а
3	ад…к
4	ак…р
5	бр…а
6	бр…а
7	да…а
8	ка…а
9	ра…б
10	ра…б

Рисунок 2.15 – Перший, другий і останній стовпці матриці

Тобто, два перших стовпці і останній нам вже відомі. Легко помітити, що символи останнього стовпця і відсортовані пари становлять трійки. Далі сортуються трійки і додається останній стовпець, потом четвірки і т. д. поки не буде відновлено всю матрицю. А на підставі записаного наперед номера початкового рядка в матриці – і сам початковий рядок (рис. 2.16).

Такий підхід потребує великих затрат часу і пам’яті. Для швидкого оберненого перетворення потрібно спочатку знайти вектор оберненого перетворення, який є масивом чисел з розміром, що дорівнює кількості символів в блоці. Для отримання вектора оберненого перетворення рядок «рдакрааааабб» записується в три стовпці масиву, як показано на рис. 2.17.

Другий стовпець (нумерація з «0») матриці сортується у лексикографічному порядку і дописується четвертий стовпець, який містить номер цього рядка (рис. 2.18). Нульовий і перший стовпці не змінюються.

Далі матриця сортується за першим і другим стовпцями в лексикографічному порядку, нульовий стовпець матриці не змінюється (рис. 2.19).

Останній стовпець чисел є вектором оберненого перетворення.

Для отримання початкового рядка візьмемо елемент вектора оберненого перетворення, що відповідає номеру початкового рядка в матриці циклічних перестановок, T[2]=6.

Тобто, першим символом початкового рядка буде шостий символ з нульового стовпця «рдакрааааабб» – символ «а». Далі T[6]=10, десятий символ з нульового стовпця «рдакрааааабб» – «б». T[10]= 4 – «р», T[4]=8 – «а», T[8]=3 – «к», T[3]= 7 – «а», T[7]= 1 – «д», T[1]= 5 – «а», T[5]= 9 – «б», T[9]= 0 – «р», T[0]= 2 – «а». Отримаємо слово «абракадабра», що і потрібно.

0	ааб… р	аабр…р		аабракада.р	аабракадабр
1	абр…..л	абра…д		абраабрак.д	абраабракад
2	абр… а	абра…а		абракадаб.а	абракадабра
3	ада…к	адаб…к		адабраабр.к	адабраабрак
4	ака… р	акад…р		акадабраа.р	акадабраабр
5	бра… а	браа…а	…	браабрака.а	браабракада
6	бра… а	брак…а		бракадабр.а	бракадабраа
7	даб… а	дабр…а		дабраабра.а	дабраабрака
8	кад… а	када…а		кадабрааб.а	кадабраабра
9	раа… б	рааб…б		раабракад.б	раабракадаб
10	рак… б	рака…б		ракадабра.б	ракадабрааб

Рисунок 2.16 – Процес визначення всіх стовпців матриці

0	ррр
1	ддд
2	ааа
3	ккк
4	ррр
5	ааа
6	ааа
7	ааа
8	ааа
9	ббб
10	ббб

Рисунок2.17 – Три стовпці початкової матриці

0	рра	0
1	дда	1
2	ааа	2
3	кка	3
4	рра	4
5	ааб	5
6	ааб	6
7	аад	7
8	аак	8
9	ббр	9
10	ббр	10

Рисунок 2.18 – Чотири стовпці початкової матриці

0	раа	2
1	даб	5
2	ааб	6
3	кад	7
4	рак	8
5	абр	9
6	абр	10
7	ада	1
8	ака	3
9	бра	0
10	бра	4

Рисунок 2.19 – Відсортовані рядки матриці

Ущільнення зображень і звуку з втратами

Під час використання методів ущільнення без втрат для ущільнення зображень і звуку (аудіо) коефіцієнт ущільнення, який досягається за використання цих методів, незначний – приблизно 1,5 – 2 рази.

Оскільки, зображення і звук орієнтовані переважно на сприйняття людиною, то це дає можливість створити спеціальні алгоритми ущільнення з втратами несуттєвої для людини інформації [15, 20 – 21].

Більшість методів ущільнення звуку з втратами ґрунтується на врахуванні особливостей сприйняття звуку людиною «кодування для сприйняття» (perceptual coding), за якого із звукового сигналу видаляється інформація, малопомітна для слуху. Хоча форма і спектр сигналу змінились, слухове сприйняття практично не змінюється, а незначне зменшення якості компенсується великим коефіцієнтом ущільнення. Таке кодування відноситься до методів кодування з втратами (lossy compression), після якого із ущільненого сигналу вже неможливо точно відновити початкову хвильову форму. Деякі особливості людського слуху, які можуть бути використані в процесі ущільнення звуку, такі [21]:

- діапазон сприйняття звуку людиною – від 20 Гц до 20 кГц (приблизно), найбільша чутливість в діапазоні від 2 до 4 кГц;
- динамічний діапазон (від найтихіших звуків до найгучніших) близько 96 дБ (більше ніж 1 до 30000 за лінійною шкалою);
- людина може розрізнити зміну частоти на 0,3% на частоті близько 1кГц;
- два сигнали, що розрізняються менше ніж на 1 дБ за амплітудою, важко розрізнити. Роздільна здатність за амплітудою залежить від частоти, найбільша чутливість в діапазоні від 2 до 4 кГц;
- просторова роздільна здатність (здібність до локалізації джерела звуку) – до 1 градуса;
- звуки різної частоти поширюються в повітрі з різною швидкістю, високочастотна складова спектру поширюється повільніше;
- людина не помічає раптове зникнення високих частот, якщо воно не перевищує порядку 2 мс;
- деякі люди можуть відчувати частоти, вищі 20 кГц;
- роздільна здатність людської системи сприйняття звуку обмежена і залежить від частоти. В межах ширини критичних смуг людина не сприймає зміни частоти. Їх ширина менше 100 Гц для нижніх частот, і більше 4 кГц для найбільш високих. Весь частотний діапазон може бути розділений на 25 критичних смуг.

В зображеннях природного походження також міститься надмірність, два основних її види такі [15]:

- статистична надмірність;
- фізіологічна надмірність.

Статистична надмірність пов'язана з тим, що сусідні відліки часто мають подібні значення яскравості, тобто вони просторово корельовані. Використання цієї властивості дозволяє значно зменшити кількість біт для подання зображення у цифровій формі.

Фізіологічна надмірність – це та частина інформації, яка не сприймається оком людини. Скорочення фізіологічної надмірності значною мірою скорочує і статистичну надмірність та навпаки.

Детальний огляд особливостей зорового сприйняття людини наведено в [20], де виділено основні особливості зорового сприйняття зображень, корисні під час розробки пристроїв кодування. Особливості зорового сприйняття нерухомих зображень наведено в табл. 2.13, де також вказано спосіб застосування в процесі ущільнення.

Великі можливості для ущільнення даних є під час ущільнення відео (рухомі зображення). Велика частина зображення передається від кадру до кадру без змін, що дозволяє створювати алгоритми ущільнення на основі вибіркового відстежування частини зображення [15].

Таблиця 2.3 – Основні особливості зорового сприйняття

Вид зорового сприйняття	Як застосовувати	Спотворення
Сприйняття змін яскравості	Малі прирости, області чорного кодуються точніше; застосування логарифмічних шкал	Непомітні
Сприйняття просторових частот	Високочастотна складова кодується меншим числом розрядів	Непомітні
Сприйняття кольору	Перехід від моделі RGB до моделі YUV; сигнали кольоровості UV піддаються субдискретизації	Непомітні
Сприйняття підкреслення контурів	У місцях наявності контурів динамічний діапазон високочастотної компоненти збільшується	Непомітні, поліпшення візуальної якості зображення

Таким чином, загальна схема ущільнення зображення або звуку з втратами містить таке:

1. На першому етапі знаходиться подання зображення або звуку у вигляді набору коефіцієнтів деякого математичного перетворення (наприклад, перетворення Фур’є, дискретне косинусне перетворення, Wavelet-перетворення або інше). Ця операція, як правило, обернена або умовно обернена.
2. На другому етапі зменшується точність подання компонент зображення або звуку (коефіцієнтів перетворення), але так, щоб виконувались задані вимоги до якості зображення або звуку. Така операція призводить до втрат інформації, тому не є оберненою.

3. На третьому етапі усувається статистична надмірність в даних, отриманих після виконання перших двох етапів. Для виконання цього етапу може застосовуватись кодування Гаффмана, арифметичне кодування та інші. Ця операція обернена [15].

Найбільше обчислювальних ресурсів витрачається для виконання першого і другого етапів; дослідження пов'язані не тільки з пошуком математичного перетворення, але і з дослідженням особливостей зорового сприйняття зображення та особливостей завадостійкої передачі цього зображення по каналах зв'язку.

Найбільш відомі стандарти ущільнення зображень, відео і звуку – стандарти JPEG та MPEG [13, 15].

Кодування зображень і звуку вивчається в курсах, присвячених обробці сигналів, детально висвітлюється в багатьох публікаціях, наприклад в [20 − 21].

2.2 Приклади розв'язування задач

Приклад 2.1. Нехай r=2 і задано довжини кодових слів (ДКС) – 1, 3, 3, 3. Перевірити, чи можна побудувати миттєвий код для таких довжин кодових слів.

Розв'язання. З нерівності Крафта (2.1) отримаємо:

$$\frac{1}{2^1}+\frac{1}{2^3}+\frac{1}{2^3}+\frac{1}{2^3}=\frac{7}{8}.$$

Оскільки $\frac{7}{8}<1$, тобто нерівність Крафта виконується, то миттєвий код з такими довжинами існує.

Приклад2.2. Нехай r=2 і задано довжини кодових слів – 1, 2, 2, 3. Перевірити, чи можна побудувати миттєвий код для таких довжин кодових слів.

Розв'язання. З нерівності Крафта (2.1) отримаємо:

$$\frac{1}{2^1}+\frac{1}{2^2}+\frac{1}{2^2}+\frac{1}{2^3}=\frac{9}{8}>1.$$

Нерівність Крафта не виконується, тому миттєвий код з такими довжинами не існує.

Приклад 2.3. Виконати ефективне кодування ансамблю із 8 знаків, якщо відомі їх імовірності.

Розв'язання. Використовуючи методику Шеннона-Фано, отримаємо сукупність кодових комбінацій, наведених в табл. 2.4.

Таблиця 2.4 – Кодування Шеннона-Фано

Знак	p	Кодові комбінації	Ступінь
z_1	1/2	1	I
z_2	1/4	01	II
z_3	1/8	001	III
z_4	1/16	0001	IV
z_5	1/32	00001	V
z_6	1/64	000001	VI
z_7	1/128	0000001	VII
z_8	1/128	0000000	VII

Обчислимо ентропію повідомлення:

$$H(z) = -\sum_{i=1}^{8} p(z_i)\log p(z_i) = -\frac{1}{2}\log\frac{1}{2} - \frac{1}{4}\log\frac{1}{4} - \ldots - \frac{1}{128}\log\frac{1}{128} = \frac{127}{64} \text{ біт.}$$

А середня кількість символів на знак після кодування визначається так:

$$l_{ср} = \sum_{i=1}^{8} p(z_i)n(z_i),$$

де $n(z_i)$ – кількість символів (розрядів) в кодовій комбінації, що відповідає знаку z_i.

$$l_{ср} = 1/2 \cdot 1 + 1/4 \cdot 2 + \ldots + 1/128 \cdot 7 = 127/64 \text{ біт.}$$

Оскільки імовірності знаків є цілочислові від'ємні значення двійки, то надлишковість під час кодування вилучається повністю, що і підтверджує те, що ентропія дорівнює середній кількості символів на знак після кодування.

Приклад 2.4. Розглянемо процедуру ефективного кодування повідомлень, утворених за допомогою алфавіту, що складається з двох знаків z_1 і z_2 з імовірностями появи $p(z_1)=0,9$, $p(z_2)=0,1$.

Розв'язання. Визначимо ентропію джерела інформації:

$$H(Z)= -0,9* \log_2 0,9 – 0,1*\log_2 0,1 = 0,47 \text{ біт.}$$

Однак, на передачу однієї букви потрібен 1 біт. Щоб видалити цю надмірність, перейдемо до кодування блоками по 2 символи (табл. 2.5). З

урахуванням того, що знаки статистично незалежні, імовірність появи блоку визначається так:

$$p(z_m\, z_n) = p(z_m) \cdot p(z_n).$$

Розрахуємо середню кількість бітів на блок повідомлення після кодування (табл. 2.5):

$$l_{срб} = \sum_{j=1}^{2} \sum_{i=1}^{2} p(z_i z_j) n(z_i z_j) == 0{,}81 \cdot 1 + 0{,}09 \cdot 2 + 0{,}09 \cdot 3 + 0{,}01 \cdot 3 = 1{,}29 \text{ біт}.$$

Оскільки символів в блоці 2, то середня кількість бітів на символ повідомлення становитиме:

$$l_{ср} = l_{срб}/2 = 0{,}645 \text{ біт}.$$

Таблиця 2.5 – Кодування Шеннона-Фано блоками по 2 символи

Блоки	p	Кодові комбінації	Ступінь розбиття
z_1z_1	0,81	1	I
z_1z_2	0,09	01	II
$z_2\, z_1$	0,09	001	III
$z_2\, z_2$	0,01	000	III

Кодування блоків, що містять 3 знаки, дасть ще кращий результат (табл. 2.6).

Таблиця 2.6 – Кодування Шеннона-Фано блоками по 3 символи

Блоки	p	Кодові комбінації	Ступінь розбиття
$z_1\, z_1\, z_1$	0,729	1	I
$z_2\, z_1\, z_1$	0,081	011	III
$z_1\, z_2\, z_1$	0,081	010	III
$z_1\, z_1\, z_2$	0,081	001	III
$z_2\, z_2\, z_1$	0,009	00011	V
$z_2\, z_1\, z_2$	0,009	00010	V
$z_1\, z_2\, z_2$	0,009	00001	V
$z_2\, z_2\, z_2$	0,001	00000	V

Приклад 2.5. Чи можна побудувати нерівномірний код, що однозначно декодується, до складу якого входять кодові комбінації з такими довжинами : $l_1 = 1;\ l_2 = l_3 = 2;\ l_4 = 3; l_5 = l_6 = l_7 = l_8 = 4;\ l_9 = l_{10} = 5$?

Розв'язання. Необхідною умовою побудови нерівномірного коду, що однозначно декодується, є виконання нерівності Крафта. Підставивши значення довжин кодових комбінацій у (2.1), отримаємо:

$$\sum_{i=1}^{10} 2^{-l_i} = 2^{-1} + 2 \cdot 2^{-2} + 2^{-3} + 4 \cdot 2^{-4} + 2 \cdot 2^{-5} = 23/16 > 1.$$

Нерівність Крафта не виконується. Таким чином, на поставлене в умові задачі запитання відповідь є негативною.

Приклад 2.6. Яку мінімальну кількість кодових комбінацій необхідно вилучити із множини кодових комбінацій 00, 01, 10, 1110, 1111, 0, 101, 1000, 1001, 110, щоб ті, які залишаться, утворювали префіксний код? Що це за комбінації [12]?

Розв'язання. Задачу можна розв'язати методом перебору. Починаючи з більш коротких кодових комбінацій, з'ясовуємо, чи не збігається вона з початковою частиною кожної більш довгої. Якщо збіг має місце, коротку кодову комбінацію вилучаємо.

Інший метод полягає в побудові кодового дерева, в якому є вузли, що відповідають усім кодовим комбінаціям, наведеним в умовах задачі, та вилученні тих кодових комбінацій, які відповідають некінцевим вузлам. Таке дерево зображено на рис. 2.20. Некінцевим вузлам дерева відповідають кодові комбінації 0 та 10 (вони закреслені). Тобто, якщо із переліку кодових комбінацій, наведених в умовах задачі, вилучити всього дві комбінації, а саме: 0 та 10, то комбінації, що залишаться, будуть утворювати префіксний код.

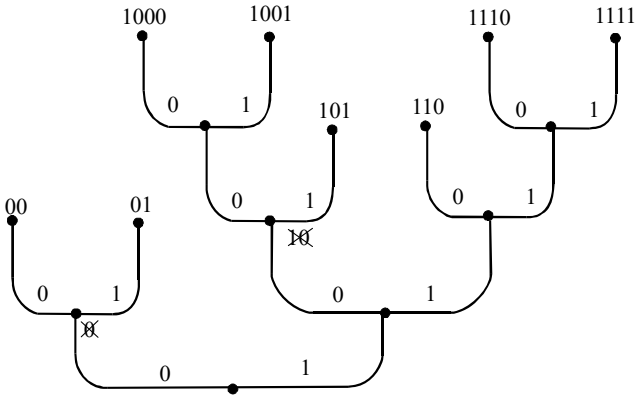

Рисунок 2.20 – Кодове дерево для перевірки коду

Приклад 2.7. Побудувати ефективні нерівномірні коди Гаффмена та Шеннона-Фано для кодування символів немарковського джерела з алфавітом $X = \{x_1, x_2, x_3, x_4, x_5\}$; імовірності виникнення символів мають такі значення: $p(x_1) = p(x_2) = 0{,}15$; $p(x_3) = 0{,}1$; $p(x_4) = 0{,}2$; $p(x_5) = 0{,}4$.

Розв'язання. Побудуємо нерівномірний код за методикою Шеннона-Фано. Таблиці 2.7 та 2.8 відображають побудову кодів Шеннона-Фано для двох варіантів.

Таблиця 2.7 – Перший варіант кодування Шеннона-Фано

$p(x_i)$	x_i	Кодова комбінація	l_i	Номер поділу
0,4	x_5	1	1	
0,2	x_4	011	3	1
0,15	x_1	010	3	3
0,15	x_2	001	3	2
0,1	x_3	000	3	4

Таблиця 2.8 – Другий варіант кодування Шеннона-Фано

$p(x_i)$	x_i	Кодова комбінація	l_i	Номер поділу
0,4	x_5	11	2	
0,2	x_4	10	2	2
0,15	x_1	01	2	1
0,15	x_2	001	3	3
0,1	x_3	000	3	4

Середня довжина кодової комбінації для першого варіанта

$$l_{сер1} = 1 \cdot 0{,}4 + 3 \cdot (0{,}2 + 0{,}15 + 0{,}15 + 0{,}1) = 2{,}2;$$

для другого

$$l_{сер2} = 2 \cdot (0{,}4 + 0{,}2 + 0{,}15) + 3 \cdot (0{,}15 + 0{,}1) = 2{,}25.$$

Код, що має більшу середню довжину кодової комбінації не є оптимальним. Тобто, методика Шеннона-Фано не гарантує оптимальності коду.

Для отримання коду Гаффмена будуємо кодове дерево коренем вниз (рис. 2.21), приписуємо гілкам символи алфавіту двійкового коду та записуємо кодові комбінації: $x_1 - 010$; $x_2 - 001$; $x_3 - 000$; $x_4 - 011$; $x_5 - 1$. Цей код є оптимальним з середньою довжиною кодової комбінації $l_{сер} = 2{,}2$.

Ентропія джерела H(X) = 2,146 біт. Відносна різниця середньої довжини оптимального ефективного коду та ентропії джерела дорівнює

$$[\ (2{,}2 - 2{,}146)\ /\ 2{,}146\ ] \times 100\% = 2{,}52\%.$$

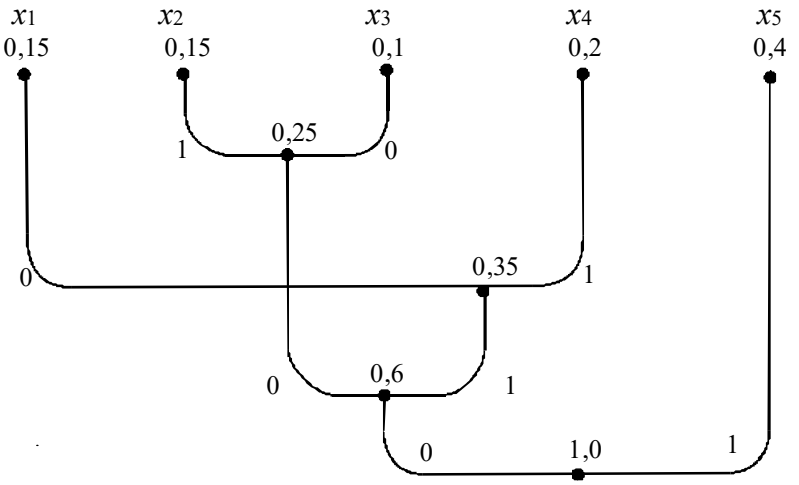

Рисунок 2.21 – Кодове дерево коренем вниз

Приклад 2.8. Маємо дискретне марковське джерело інформації, алфавіт якого містить два символи {A,B}, а глибина пам'яті h = 1. Умовні ймовірності появи символів такі:

$$\begin{bmatrix} p(A/A) & p(B/A) \\ p(A/B) & p(B/B) \end{bmatrix} = \begin{bmatrix} 0{,}9 & 0{,}1 \\ 0{,}1 & 0{,}9 \end{bmatrix}.$$

Побудувати нерівномірні ефективні коди для кодування слів джерела довжиною від 1 до 4. Прослідкувати наближення середньої довжини кодової комбінації (в розрахунку на один символ джерела) до ентропії джерела. Закодувати розробленими кодами послідовність із 24 символів *BBBBBAAAAAAAAAAAAAAAAA*, яку згенерувало джерело.

Розв'язання. Для розрахунку імовірностей появи слів на виході джерела та його ентропії необхідно знати безумовні ймовірності p(A) та p(B) виникнення символів. Оскільки:

$$p(AB) = p(BA),$$

то

$$p(A) * p(B/A) = p(B) * p(A/B);$$
$$0{,}1 * p(A) = 0{,}1 * p(B);$$
$$p(A) + p(B) = 1.$$

Звідки p(A) = p(B) = 0,5. А ентропія джерела така:

$$H(Z) = -\sum_{q=1}^{L} p(z_q) \sum_{i=1}^{L} p_{z_q}(z_i) \log\ p_{z_q}(z_i) = -0{,}5(0{,}9 * \log_2 0{,}9 + 0{,}1 * \log_2 0{,}1) -$$

$$-0{,}5(0{,}9 * \log_2 0{,}9 + 0{,}1 * \log_2 0{,}1) = 0{,}47 \text{ біт.}$$

Побудова коду для кодування окремих символів, тобто слів довжиною у один символ, дасть середню довжину кодової комбінації $l_{сер\,(1)} = 1$, оскільки одному із символів, наприклад *A*, зіставляється 0, а іншому – 1.

Для побудови ефективних кодів необхідно брати слова більшої довжини, тому необхідно знайти імовірності появи таких слів на виході джерела. Всі дані для розрахунків маємо. Так, ймовірність p(AB) появи слова AB буде дорівнювати p(AB) = = p(A)·p(B/A) = 0,5·0,1 = 0,05, а слова BBAA:

$$p(BBAA) = p(B) \cdot p(B/B) \cdot p(A/B) \cdot p(A/A) = 0{,}5 \cdot 0{,}9 \cdot 0{,}1 \cdot 0{,}9 = 0{,}04505.$$

Таблиці 2.9, 2.10, 2.11 відображають процес побудови кодів Шеннона - Фано для кодування слів джерела довжиною у 2, 3 та 4 символи.

Таблиця 2.9 – Кодування слів довжиною 2 символи

Слово джерела (СД)	Ймовірність появи СД	Кодова комбінація (КК)	Довжина КК	Номер поділу
AA	0,45	1	1	1
BB	0,45	01	2	2
AB	0,05	001	3	3
BA	0,05	000	3	

Таблиця 2.10 – Кодування слів довжиною 3 символи

Слово джерела (СД)	Ймовірність появи СД	Кодова комбінація (КК)	Довжина КК	Номер поділу
AAA	0,405	1	1	1
BBB	0,405	01	2	2
AAB	0,045	0011	4	4
ABB	0,045	0010	4	3
BBA	0,045	0001	4	5
BAA	0,045	00001	5	6
ABA	0,005	000001	6	7
BAB	0,005	000000	6	

Таблиця 2.11 – Кодування слів довжиною 4 символи

Слово джерела (СД)	Ймовірність появи СД	Кодова комбінація (КК)	Довжина КК	Номер поділу
AAAA	0,3645	1	1	
BBBB	0,3645	01	2	1
AAAB	0,0405	0011	4	2
AABB	0,0405	00101	5	4
ABBB	0,0405	00100	5	5
BAAA	0,0405	00011	5	3
BBAA	0,0405	00010	5	7
BBBA	0,0405	00001	5	6
AABA	0,0045	0000011	7	8
ABAA	0,0045	00000101	8	10
ABBA	0,0045	00000100	8	11
BAAB	0,0045	00000011	8	9
BBAB	0,0045	00000010	8	13
BABB	0,0045	00000001	8	12
ABAB	0,0005	000000001	9	14
BABA	0,0005	000000000	9	15

Отримаємо значення середньої довжини $l_{сер(N)}$ кодової комбінації (в розрахунку на один символ джерела) для коду, розробленого для кодування слів джерела довжиною N:

$l_{сер(2)} = [(1+2)\times 0{,}45+3\times 2\times 0{,}05]/2 = 0{,}825;$

$l_{сер(3)} = [(1+2)\times 0{,}405+(4\times 3+5)\times 0{,}045+6\times 2\times 0{,}05]/3 = 0{,}68;$

$l_{сер(4)} = [(1+2)\times 0{,}3645+(4+5\times 5)\times 0{,}0405+(7+8\times 5)\times 0{,}0045+$

$+9\times 2\times 0{,}0005]/4 = 0{,}622.$

Відносна різниця між середньою довжиною кодової комбінації та ентропією джерела для побудованих кодів становить 75%, 44%, 32%. Під час збільшення довжини N слів, що підлягають кодуванню, середня довжина $l_{сер(N)}$ кодової комбінації буде асимптотично наближатися до ентропії джерела.

Можна переконатися, що коди, побудовані за методикою Гаффмена, в цьому випадку будуть мати такі самі середні довжини кодових комбінацій. Це означає, що всі отримані коди є оптимальними, але кожен для свого джерела.

Щоб закодувати згенеровану джерелом послідовність символів побудованими кодами, поділяємо її на підпослідовності (слова) відповідної довжини та застосовуємо коди таблиць 2.9, 2.10, 2.11. Отримуємо:

– для слів по два символи 0101000111111111 довжина закодованої послідовності $L_2 = 16$;

– для слів по три символи 010001111111 довжина закодованої послідовності $L_3 = 12$;

– для слів по чотири символи 01000111111 довжина закодованої послідовності $L_4 = 11$.

Приклад 2.9. Розробити ефективний код для кодування послідовності символів на виході немарковського дискретного джерела інформації з алфавітом {A,B,C} та імовірностями появи символів p(A) = 0,15; p(B) = 0,1; p(C) = 0,75. Відносна різниця між середньою довжиною кодової комбінації та ентропією джерела не має перевищувати 2%. Закодувати таким кодом послідовність довжиною у 30 символів:

CACCCCCCCCCCBCCACCCCBCCBCAACCC.

Розв'язання. Зменшення середньої довжини кодової комбінації на один символ джерела можна досягти, якщо будувати ефективний код для укрупненого алфавіту. Хоча ефективний нерівномірний код розробляється на основі статистичних характеристик джерела, знання цих характеристик не дає можливості розрахувати середню довжину кодової комбінації без побудови коду.

Тому шлях розв'язання задачі має бути таким. Послідовно укрупнюємо алфавіт; для кожного укрупнення будуємо нерівномірний ефективний код, розраховуємо середню довжину кодової комбінації на один символ джерела та перевіряємо, чи задовольняються вимоги до неї. Процес укрупнення припиняємо, як тільки ці вимоги задовольняються.

На рис. 2.22 зображено кодове дерево, що відповідає коду Гаффмена для кодування за одним символом джерела. Для символу *A* довжина кодової комбінації дорівнює 2, для символу *B* – 2, для символу *C* – 1. Середня довжина кодової комбінації:

$l_{сер\,(1)} = 2\cdot p(A) + 2\cdot p(B) + 1\cdot p(C) = 2(0{,}15+0{,}1)+1\cdot 0{,}75 = 1{,}25$ біт.

Ентропія джерела:

$H = -p(A)\cdot\log_2 p(A) - p(B)\cdot\log_2 p(B) - p(C)\cdot\log_2 p(C) = 1{,}054$ біт.

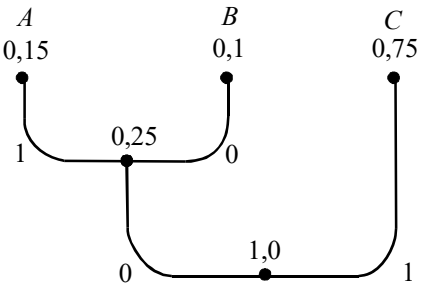

Рисунок 2.22 – Кодове дерево для символів джерела

Відносна різниця між середньою довжиною кодової комбінації та ентропією джерела становить:

$$[(1{,}25 - 1{,}054) / 1{,}054] \times 100\% = 18{,}6\%,$$

що не задовольняє умови задачі.

Об'єднаємо по два символи джерела. Оскільки джерело не має пам'яті, імовірності появи слів $p(AB) = p(A)\cdot p(B)$. Кодове дерево для цього варіанта зображено на рис. 2.23. Результати побудови коду Гаффмена згідно з цим деревом наведено у табл. 2.12.

Середня довжина кодової комбінації в розрахунку на один символ вихідного джерела має значення:

$$l_{сер\,(2)} = [1\times 0{,}5625+3\times(2\times 0{,}1125+0{,}075)+4\times 0{,}075+6\times(0{,}0225+ +2\times 0{,}015+0{,}01)]/2 = 1{,}069.$$

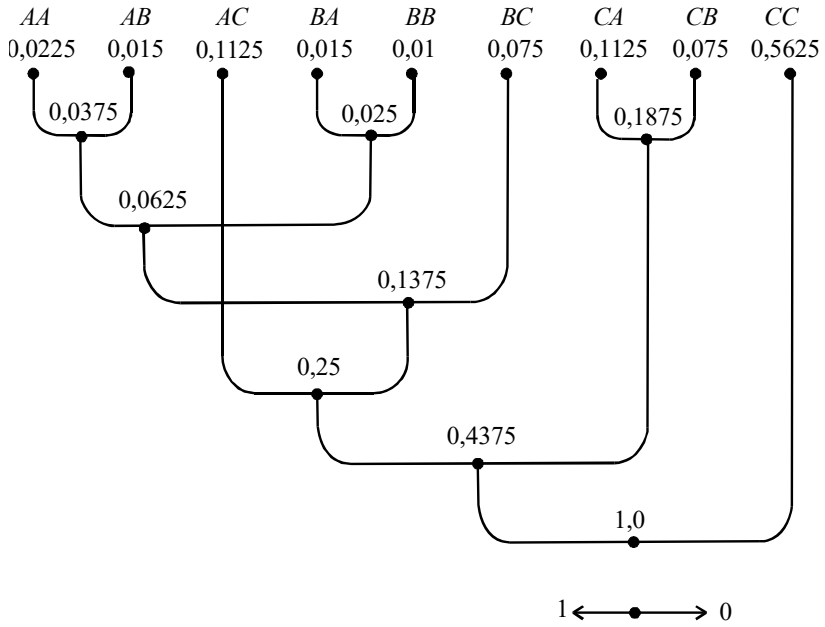

Рисунок 2.23 – Кодове дерево для укрупненого по два символи алфавіту джерела

Таблиця 2.12 – Кодові комбінації для укрупненого по два символи

Слово джерела (СД)	Ймовірність появи СД	Кодова комбінація (КК)	Довжина КК
AA	0,0225	110111	6
AB	0,015	110110	6
AC	0,1125	111	3
BA	0,015	110101	6
BB	0,01	110100	6
BC	0,075	1100	4
CA	0,1125	101	3
CB	0,075	100	3
CC	0,5625	0	1

Відносна різниця між $l_{сер(2)}$ та ентропією джерела

$$[(1{,}069 - 1{,}054) / 1{,}054] \times 100\% = 1{,}4\%,$$

що менше, ніж 2%, тобто такий код задовольняє вимоги.

Щоб закодувати послідовність, яку формує джерело, поділяємо її на слова довжиною по 2 символи та застосовуємо для цих слів код табл. 2.12. Маємо

$$10100000110010100110010010111 10.$$

Довжина двійкової послідовності дорівнює 31.

Приклад 2.10. Розробити алгоритм кодування для марковського джерела з алфавітом {A,B,C} та глибиною пам'яті h = 1. Умовні імовірності появи символів задаються матрицею:

$$\begin{bmatrix} p(A/A) & p(B/A) & p(C/A) \\ p(A/B) & p(B/B) & p(C/B) \\ p(A/C) & p(B/C) & p(C/C) \end{bmatrix} = \begin{bmatrix} 0{,}15 & 0{,}10 & 0{,}75 \\ 0{,}85 & 0{,}10 & 0{,}05 \\ 0{,}15 & 0{,}70 & 0{,}15 \end{bmatrix}.$$

Знайти відносну різницю між середньою довжиною кодової комбінації ефективного коду та ентропією джерела. Закодувати послідовність довжиною у 30 символів

ACACBACBACBABABBACBBACCACBACBA.

Розв'язання. Оскільки глибина пам'яті h = 1, кількість Q станів джерела дорівнює потужності його алфавіту, тобто Q = 3. Для кожного стану, який визначається попереднім символом на виході джерела, будуємо нерівномірний код. Застосовуючи методику Гаффмена або Шеннона-Фано, легко отримати коди, наведені в таблицях.

Таблиця 2.13 – Після символу A

Символ, що очікується	Умовна ймовірність	Кодова комбінація (КК)	Довжина КК
C	p(C/A) = 0,75	0	1
A	p(A/A) = 0,15	11	2
B	p(B/A) = 0,10	10	2

Таблиця 2.14 – Після символу B

Символ, що очікується	Умовна ймовірність	Кодова комбінація (КК)	Довжина КК
A	p(A/B) = 0,85	1	1
B	p(B/B) = 0,10	01	2
C	p(C/B) = 0,05	00	2

Таблиця 2.15 – Після символу С

Символ, що очікується	Умовна ймовірність	Кодова комбінація (КК)	Довжина КК
A	p(A/C) = 0,15	01	2
B	p(B/C) = 0,70	1	1
C	p(C/C) = 0,15	00	2

Знайдемо значення $l_{сер/A}$ середньої довжини кодової комбінації для коду, який застосовується після символу А :

$$l_{сер/A} = l_{A/A} \cdot \text{p(A/A)} + l_{B/A} \cdot \text{p(B/A)} + l_{C/A} \cdot \text{p(C/A)} =$$
$$= 2 \times 0{,}15 + 2 \times 0{,}10 + 1 \times 0{,}75 = 1{,}25;$$

тут $l_{A/A}$, $l_{B/A}$, $l_{C/A}$ – довжини кодових комбінацій для кодування відповідно символів А, В, С коду, який застосовується після символу А. Аналогічно отримаємо значення $l_{сер/B}$, $l_{сер/C}$ середніх довжин кодових комбінацій для кодів, що застосовуються після символів В та С, відповідно:

$$l_{сер/B} = 1{,}15; \qquad l_{сер/C} = 1{,}3.$$

Середня довжина $l_{сер}$ кодової комбінації у разі застосування марковського алгоритму

$$l_{сер} = l_{сер/A} \cdot \text{p(A)} + l_{сер/B} \cdot \text{p(B)} + l_{сер/C} \cdot \text{p(C)},$$

де p(A), p(B), p(C) – безумовні ймовірності появи символів А, В, С на виході джерела.

Аналогічно прикладу 2.8 знаходимо

$$\text{p(A)} = 0{,}361; \quad \text{p(B)} = 0{,}302; \quad \text{p(C)} = 0{,}337.$$

Тоді

$$l_{сер} = 1{,}237.$$

Ентропія джерела (див. задачу 2.8)

$$H = 1{,}004 \text{ біт.}$$

Відносна різниця між $l_{сер}$ та H становить:

$$[(1{,}237 - 1{,}004) / 1{,}004] \times 100\% = 23{,}2\% .$$

Закодуємо послідовність символів, наведену в умовах задачі. Таблиці 2.13 – 2.15 дають змогу виконати таке кодування. Для першого символу послідовності можна застосувати рівномірний код, наприклад: А − 00; В – 01; С – 10. Більш раціональним є код, побудований з урахуванням безумовних ймовірностей появи символів, наприклад А – 0; В – 11; С – 10. Обираючи цей варіант для кодування першого символу та застосовуючи коди таблиць 2.13 – 2.15 для подальших символів, отримаємо

$$0001011011011101100110101100001011011.$$

Кодова послідовність має 37 двійкових символів, в той час як за кодування символів джерела рівномірним двійковим кодом (мінімальна довжина такого коду в цьому випадку становить 2) довжина кодової послідовності буде дорівнювати 60.

2.3 Контрольні питання і задачі

1. Наведіть переваги застосування нерівномірних кодів.
2. Яка нерівність задає умову однозначного декодування нерівномірного коду?
3. Поясніть поняття миттєвого коду.
4. Коли застосовують блокові укорочені коди?
5. Поясніть принцип побудови блокового укороченого коду.
6. Які ви знаєте нерівномірні коди?
7. Сформулюйте теорему Шеннона для кодування каналу без завад.
8. Наведіть класифікацію алгоритмів ущільнення без втрат.
9. Наведіть та охарактеризуйте метод ефективного кодування Шеннона-Фано.
10. Охарактеризуйте словникові методи ущільнення даних.
11. Охарактеризуйте статистичні методи ущільнення даних.
12. Дайте характеристику стиснення зображень методом RLE.
13. Які недоліки кодування Гаффмена?
14. Наведіть порядок побудови кодового дерева.
15. Що таке префіксні коди?
16. Наведіть алгоритм кодування методом LZW.
17. Наведіть алгоритм декодування методом LZW.
18. Поясніть основи арифметичного кодування.
19. Які основні недоліки арифметичного кодування?
20. Як працює Range – кодер?
21. Які недоліки алгоритму LZW?
22. Чи забезпечують алгоритми ущільнення без втрат високий коефіцієнт ущільнення зображень?
23. Який основний недолік арифметичних методів ущільнення даних?
24. На якому етапі кодування зображень застосовуються алгоритми ущільнення без втрат?

Задачі

1. Виконати ефективне кодування ансамблю із 4 знаків згідно з методом Шеннона-Фано, обчислити ентропію ансамблю та середню кількість бітів на знак повідомлення після кодування. Імовірності знаків такі: $p(z_1)=1/2$, $p(z_2)=1/3$, $p(z_3)=1/12$, $p(z_4)=1/12$. Відповідь: $H(Z)=1{,}63$ біт, $l_{ср}=1{,}67$ біт/знак.
2. Виконати ефективне кодування Шеннона-Фано блоками по 2 символи повідомлень, утворених за допомогою алфавіту, що складається з двох знаків z_1 і z_2 з імовірностями появи $p(z_1)=0{,}7$, $p(z_2)=0{,}3$. Обчислити ентропію повідомлення та середню кількість бітів на знак повідомлення після кодування.
   Відповідь: $H(Z)=0{,}88$ біт, $l_{ср}=0{,}9$ біт/знак.
3. Виконати ефективне кодування Шеннона-Фано блоками по 3 символи повідомлень, утворених за допомогою алфавіту, що складається з двох знаків z_1 і z_2 з імовірностями появи $p(z_1)=0{,}7$,

$p(z_2)$=0,3. Обчислити ентропію повідомлення та середню кількість бітів на знак повідомлення після кодування.
Відповідь: H(Z)=0,88 біт, $l_{ср}$=0,9 біт/знак.

4. Подати знаки повідомлення «абракадабра» префіксним кодом згідно з методом Гаффмана. Застосувати дерево коренем вверх, ліва гілка «0», права «1». Відповідь: а=1, б=01, р=001, к=0001, д=0000.
5. Виконати ефективне кодування повідомлення «ddaaaaaccc» методом кодування за ступенем новизни. Обчислити середню кількість бітів на знак повідомлення після кодування. Відповідь: вихід – 11010100001100, $l_{ср}$=1,3 біт/знак.
6. Виконати ефективне кодування повідомлення «dbaaaaaccc» методом MTF. Обчислити ентропію повідомлення та середню кількість бітів на знак повідомлення після кодування. Відповідь: вихід – 111110110000011100, H(Z)=1,68 біт, $l_{ср}$=1,8 біт/знак.
7. Ущільнити методом LZW таке повідомлення: «dedecdedec».
Відповідь: d e 256 c 256 258.
8. Ущільнити методом LZW таке повідомлення: «/we/we/wet».
Відповідь: / w e 256 258 257 t.
9. Виконати арифметичне кодування такого повідомлення «аббат». Відповідь: 0.24448.
10. Виконати арифметичне кодування такого повідомлення «bedkmdomce». Відповідь: 0.0478570048.
11. Виконати BWT-перетворення такої послідовності: «dedecdedec».
Відповідь: eeeeccdddd, 4.
12. Виконати BWT-перетворення такої послідовності: «abaka».
Відповідь: kabaa, 1.
13. Виконати обернене BWT-перетворення такої послідовності: «тббаа», 0. Відповідь: аббат.
14. Розробити програму ущільнення файлів імовірнісним методом. Імена файлів вводяться з командного рядка. Мова програмування C++.
15. Розробити програму ущільнення файлів методом Шеннона-Фано. Мова програмування C++ .
16. Розробити програму ущільнення файлів напівадаптивним методом Гаффмана. Мова програмування C++ .
17. Розробити програму ущільнення файлів методом MTF. Мова програмування C++ .
18. Розробити програму ущільнення файлів методом LZW. Мова програмування C++ .
19. Розробити програму ущільнення файлів з використанням арифметичного range-кодера. Мова програмування C++ .
20. Розробити програму виконання BWT-перетворення над файлами. Розмір блоків має задаватися користувачем в межах 8-256. Мова програмування C++.

3 ЗАВАДОСТІЙКЕ КОДУВАННЯ ІНФОРМАЦІЇ

3.1 Теоретичні положення

Кодування як метод боротьби із завадами. Застосування кодів, що виявляють і виправляють помилки, є одним із способів боротьби з випадковими завадами в системах передачі та обробки інформації. Для кожного конкретного каналу теорія завадостійкого кодування дозволяє вибрати найбільш ефективний метод завадостійкого кодування. Розрізняють два напрямки завадостійкого кодування [22]:

- кодування з виправленням помилок (корегувальні коди) – приймач виявляє і виправляє помилки;
- кодування з виявленням помилок – приймач виявляє помилку і виконує запит на повторну передачу помилкового блока.

Використання простих завадостійких кодів з невеликою надмірністю дозволяє знизити імовірність помилки на біт з 10^{-6} до 10^{-9} і нижче.

Теорія завадостійкого кодування ґрунтується на теоремі Шеннона про кодування для каналу із завадами [1].

Теорема Шеннона про кодування для каналу із завадами. Теорема задає умови передачі інформації по каналу зв'язку із завадами з як завгодно малою імовірністю помилки.

Формулювання теореми таке [1].

1. За будь-якої продуктивності джерела інформації, меншої ніж пропускна здатність каналу, існує спосіб кодування, який забезпечує передачу інформації, що створюється джерелом повідомлень, з як завгодно малою імовірністю помилки.
2. Не існує способу кодування, який дозволив би передавати інформацію з як завгодно малою імовірністю помилки, якщо продуктивність джерела повідомлення більша за пропускну здатність каналу.

Теорема задає теоретичну межу ефективності системи за достовірної передачі інформації. Теорема неконструктивна, оскільки не указує шляхи побудови ефективних завадостійких кодів, але обґрунтувавши можливість такого кодування, вона мобілізувала зусилля вчених на розробку конкретних кодів.

Основні принципи завадостійкого кодування. Виявляти помилки можна, коли використовуються не всі $N_0=m^n$ комбінацій коду (m – основа коду, а n – довжина (значність)), а тільки N комбінацій і $N<N_0$ [2, 6]. Комбінації, що використовуються, називаються *дозволеними*, а

невикористовувані *забороненими*. Помилки не можуть бути виявлені, якщо множина помилок в одній дозволеній кодовій комбінації перетворює її на іншу дозволену кодову комбінацію.

Для збільшення кількості помилок, які виправляє код, потрібно збільшувати відстань між дозволеними кодовими комбінаціями. Відстань між двома кодовими комбінаціями визначається з використанням метрики Гемінга [5 – 6]:

$$d_{ij} = \sum_{k=1}^{n} \left| x_{ik} - x_{jk} \right|. \qquad (3.1)$$

Для двійкового коду відстань дорівнює кількості знаків, в яких одна кодова комбінація відрізняється від іншої, її визначають шляхом додавання за модулем 2 двох кодових комбінацій і підрахунку кількості одиниць в цій сумі. Найменшу відстань для цього коду позначають символом «d_{min}» і називають *кодовою відстанню коду*.

Для виявлення всіх помилок кратності q_d (кількість неправильно переданих символів в межах однієї кодової комбінації) в кодовій комбінації необхідно і достатньо, щоб кодова відстань задовольняла таку умову [6]:

$$d_{min} \geq q_d + 1. \qquad (3.2)$$

Для коду, що виявляє одну помилку, $d_{min}=2$.

Необхідна і достатня умова виправлення помилок кратності q_c така:

$$d_{min} \geq 2q_c + 1. \qquad (3.3)$$

Для коду, що виправляє одну помилку, $d_{min}=3$.

Необхідна і достатня умова виправлення помилок кратності q_c і виявлення помилок кратності q_d така [5]:

$$d_{min} \geq q_c + q_d + 1, \qquad (3.4)$$

де $q_d \geq q_c$.

Наведені вирази дозволяють будувати прості завадостійкі коди.

Відомі схеми кодування ґрунтуються на різних математичних теоріях, дуже несхожі одна на одну, але можна виділити дві загальні властивості характерні для них.

Перша – використання надмірності. Закодовані послідовності містять додаткові (надмірні) символи.

Друга – властивість усереднювання. Інформація, що міститься в кодовій послідовності, перерозподіляється на надмірні символи, які залежать від декількох інформаційних символів [22].

Класифікація завадостійких кодів та їх основні параметри. Коди, які виявляють і виправляють помилки, можна розділити на імовірнісні та алгебраїчні [2, 3]. Імовірнісні коди не знайшли широкого застосування, тому не розглядаються в посібнику.

В алгебраїчних завадостійких кодах [23] здатність виявляти і виправляти помилки є властивістю їхньої алгебраїчної структури. Їх можна розділити на *блокові* та *деревоподібні (*рис. 3.1*)*.

До блокових кодів відносять коди, в яких операції кодування і декодування виконуються над блоками фіксованої довжини. Кодер блокового коду відображає послідовність з k символів в послідовність із n символів і є пристроєм «без пам'яті», тобто кожний вихідний блок з n символів залежить тільки від відповідного вхідного блоку з k символів і не залежить від інших блоків. Параметри блокового коду: довжина коду n, довжина інформаційної послідовності k, швидкість коду r=k/n, мінімальна кодова відстань d_{min}.

До деревоподібних кодів відносять коди, в яких обробляється неперервна послідовність символів без розподілу її на блоки. Найпростішими з погляду реалізації і тому найпоширенішими є лінійні деревоподібні коди, які називають згортними кодами. У згортному коді кожен набір n_0 вихідних символів залежить від поточного вхідного набору k_0 і від K-1 попередніх вхідних наборів, тому кодер для згортного коду є пристроєм з пам'яттю. Параметри згортного коду: K – *конструктивна довжина* згортного коду, $n_A=Kn_0$ – *довжиною кодового обмеження,* швидкістю коду $r=k/n_0$ і *вільною відстанню* d_{free} (аналог кодової відстані).

За основою коду m *завадостійкі* коди можна розділити на *двійкові* (за m=2) і *недвійкові* (за m>2*)*.

Розрізняють також *роздільні* і *нероздільні коди*. *Роздільні* коди містять інформаційну частину і перевірні символи. В *нероздільних* кодах виділити ці складові неможливо.

В лінійних кодах перевірні символи отримують шляхом лінійної комбінації інших символів коду. Кожний лінійний код має еквівалентний систематичний код. Якщо в коді перших k символів – інформаційні, а наступні – перевірні, то такий код називають систематичним.

Лінійні коди, які характеризуються циклічними властивостями, називають *циклічними* кодами. У циклічному коді кодова комбінація є результатом циклічного зсуву іншої кодової комбінації, тобто, якщо $\vec{C} = (c_0, c_1, ..., c_{n-1})$ – кодове слово циклічного коду, то і кодове слово $\vec{C}' = (c_1, ..., c_{n-1}, c_0)$ також є кодовим словом цього коду. Циклічні коди характеризуються рядом структурних особливостей, які спрощують реалізацію операцій кодування та декодування.

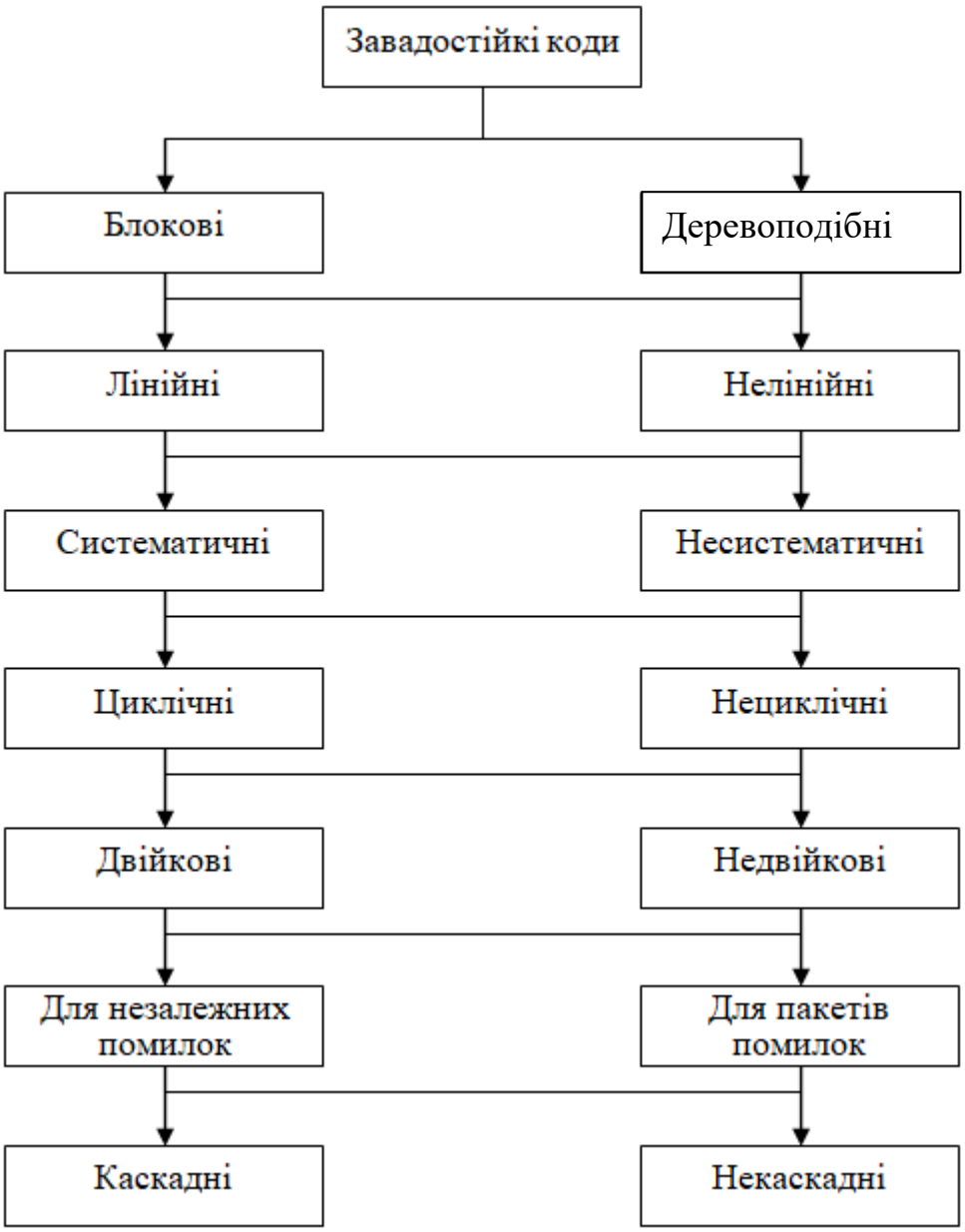

Рисунок 3.1 – Класифікація завадостійких кодів

Розрізняють коди для виправлення *випадкових* або *незалежних* помилок і коди для виправлення *пакетів* помилок. На практиці пакети помилок перед декодуванням розбивають на випадкові помилки за рахунок використання перемішування для зміни порядку символів в закодованій послідовності і відновлення початкового порядку після прийому, що спрощує виправлення пакетів помилок з використанням кодів для виправлення *випадкових* помилок.

В основі побудови каскадних кодів лежить ідея сумісного використання декількох кодів – спочатку дані кодуються зовнішнім кодом, а потім закодовані символи зовнішнього коду кодуються кодером внутрішнього коду. Це дозволяє значно підвищити ефективність застосування кодування порівняно з некаскадними методами. Якщо d_1 і d_2

– мінімальні відстані складових кодів, то мінімальна відстань сформованого каскадного коду дорівнює $D=d_1 \cdot d_2$.

Отже, завадостійкі коди характеризуються за такими параметрами:

1. Довжина коду – n;
2. Довжина інформаційної послідовності – k;
3. Довжина перевірної послідовності – m=n-k;
4. Кодова відстань коду – d_{min};
5. Швидкість коду – r=k/n;
6. Надмірність коду –1/r;
7. Імовірність виявлення помилки – $p_{вп}$;
8. Імовірність невиявлення помилки (спотворення) – $p_{нв}$.

Граничні співвідношення між параметрами завадостійких кодів. Встановлюють співвідношення між здатністю виявляти або виправляти помилки і надмірністю. Деякі з них такі [5, 9].

1. Межа Гемінга визначається такими співвідношеннями:

$$n - k \geq \log_q \sum_{i=0}^{t_u} C_n^i \cdot (q-1)^i, \tag{3.5}$$

де q – основа коду;

C_n^i – кількість сполучень з n елементів за i:

$$C_n^i = \frac{n!}{i! \cdot (n-i)!}. \tag{3.6}$$

Для двійкових кодів (q=2):

$$n - k \geq \log_2 \sum_{i=0}^{t_u} C_n^i. \tag{3.7}$$

Доцільно використовувати для високошвидкісних кодів.

2. Межа Плоткіна. Визначається таким співвідношеннями:

$$d_0 \leq n \cdot (q-1) \cdot q^{k-1} / (q^k - 1). \tag{3.8}$$

Для двійкового коду:

$$d_0 \leq n \cdot 2^{k-1} / (2^k - 1). \tag{3.9}$$

Доцільно використовувати для низькошвидкісних кодів.

Границі Гемінга і Плоткіна задають мінімальну надмірність, за якої існує завадостійкий код, що має мінімальну кодову відстань і гарантовано виправляє t_u-кратні помилки; є верхніми межами для кодової відстані за заданих n і k.

3. Межа Варшамова-Гільберта (нижня межа), визначається такими співвідношеннями:

$$q^{n-k} > \sum_{i=0}^{d_{min}-2} C_{n-1}^{i}(q-1)^{i}; \qquad (3.10)$$

для двійкових кодів:

$$2^{n-k} > \sum_{i=0}^{d_{min}-2} C_{n-1}^{i}. \qquad (3.11)$$

Показує, за якої кількості перевірних символів (n-k) існує код, що гарантовано виправляє помилки кратності t_u.

Способи подання лінійних блокових кодів. Більшість практичних схем кодування ґрунтуються на лінійних кодах [2 – 5, 9, 22 – 30]. Блоковий код довжиною n з 2^k словами, які утворюють k-вимірний підпростір V_k, що породжується базисом з k лінійно незалежних векторів, векторного n-вимірного простору V_n, називається лінійним (n, k) кодом. Ці k лінійно незалежних векторів утворюють рядки породжувальної матриці (n, k) коду:

$$G = \begin{bmatrix} g_0 \\ g_1 \\ \cdot \\ \cdot \\ \cdot \\ g_{k-1} \end{bmatrix} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots g_{0,n-1} \\ g_{1,0} & g_{1,1} & \cdots g_{1,n-1} \\ \cdots & \cdots & \cdots \\ g_{k,0} & g_{k,1} & \cdots g_{k-1,n-1} \end{bmatrix}. \qquad (3.12)$$

Інформаційна послідовність розбивається на повідомлення $\vec{U} = (u_0, u_1, ..., u_{k-1})$ довжини k, а процес кодування полягає у відображенні цих повідомлень в кодові слова $\vec{C} = (c_0, c_1, ..., c_{n-1})$ довжини n:

$$\vec{C} = \vec{U}G. \qquad (3.13)$$

Деколи перевагу має подання кодових слів в систематичній формі $\vec{C} = (\vec{U}, \vec{V})$, де $\vec{U}$ – інформаційна частина із k символів, а $\vec{V}$ – перевірна частина із m=n-к символів. Породжувальна матриця систематичного коду така:

$$G = [I_k P] = \begin{bmatrix} 1 & 0 & \ldots 0 & g_{0,0} & g_{0,1} & \ldots g_{0,n-1} \\ 0 & 1 & \ldots 0 & g_{1,0} & g_{1,1} & \ldots g_{1,n-1} \\ \ldots & \ldots & & \ldots & \ldots & \ldots \\ 0 & 0 & \ldots 1 & g_{k,0} & g_{k,1} & \ldots g_{k-1,n-1} \end{bmatrix}. \quad (3.14)$$

Тобто, матриця *G* містить одиничну матрицю I_k розміром k×k, яка формує інформаційну частину слова, і матрицю P розміром k×(n-k) для визначення перевірних символі.

Матриця, простір рядків якої ортогональний простору рядків породжувальної матриці, називається перевірною матрицею H:

$$GH^T = 0. \quad (3.15)$$

Кожне кодове слово лінійного коду $\vec{C}$ також задовольнятиме умову ортогональності:

$$\vec{C}H^T = \vec{U}GH^T = 0. \quad (3.16)$$

Для породжувальної матриці у формі (3.14) для виконання умови ортогональності перевірна матриця H має виглядати так:

$$H = \left[P^T I_{n-k}\right]. \quad (3.17)$$

Розглянемо принцип синдромного декодування. Нехай прийняті слова мають вигляд $\vec{Y} = \vec{C} + \vec{E}$, де $\vec{Y} = (y_0, y_1, ..., y_{n-1})$ – прийнятий вектор, а $\vec{E} = (e_0, e_1, ..., e_{n-1})$ – вектор помилок. На приймальній стороні спочатку обчислюють синдром:

$$\vec{S} = \vec{Y}H^T = (s_0, s_1, ...s_{n-k-1}). \quad (3.18)$$

Синдром залежить тільки від вектора помилок.

Якщо $\vec{S} = 0$, то помилки відсутні у прийнятому слові, інакше слово $\vec{Y}$ містить помилки і за значенням синдрому $\vec{S}$ можна визначати їх положення.

Породжувальна матриця циклічного коду така:

$$G = \begin{bmatrix} g_0 & g_1 & \ldots g_{n-k} & 0 & \ldots & 0 \\ 0 & g_0 & g_1 & \ldots g_{n-k} & \ldots 0 \\ \ldots & \ldots & & \ldots & \ldots & \ldots \\ 0 & \ldots & 0 & g_0 & g_1 & \ldots & g_{n-k} \end{bmatrix}. \qquad (3.19)$$

Для компактного запису циклічного (n,k) коду часто використовується породжувальний або твірний поліном g(x) степені n-k, з коефіцієнтами з поля GF(q) (для двійкових кодів q={0,l}):

$$g(x) = g_0 + g_1 x + g_2 x^2 + \ldots + g_{n-k} x^{n-k}. \qquad (3.20)$$

Якщо у вигляді полінома подати інформаційне повідомлення U(x), то кодові слова коду утворюються шляхом множення полінома, що відповідає інформаційному повідомленню на породжувальний поліном коду:

$$C(x) = U(x)g(x). \qquad (3.21)$$

Коефіцієнти твірних поліномів об'єднують в двійкове слово і подають у восьмиричній системі числення, а також твірні поліноми можна записувати шляхом перерахування степенів з ненульовими коефіцієнтами. Наприклад, запис $g(x) = 1 + x + x^4 + x^6$: g = $1010011_2 = 123_8$; g = (0,1,4,6).

Лінійні двійкові блокові коди. У лінійному двійковому блоковому коді довжиною n символів 2^k кодових слів утворюють k-вимірний підпростір векторного простору n-послідовностей двійкового поля GF(2).

Поле – це множина математичних об'єктів, з якими можна виконувати такі математичні операції: додавати, віднімати, множити і ділити. Поле, що містить скінченну кількість елементів, називається полем Галуа (G). Для поля, що складається з двох елементів – нуля «0» і одиниці «1» (поле Галуа, позначається GF(2)), визначимо операції додавання і множення:

0 + 0=0, 0 · 0=0;
0 + 1=1, 0 · 1=0;
1 + 0=1, 1 · 0=0;
1 + 1=0, 1 · 1=1.

Такі операції додавання і множення називаються додаванням за модулем 2 (mod 2) і множенням за модулем 2. З рівності 1+1 = 0 витікає, що 1=-1 і 1+1=1-1, а з рівності 1· 1=1 , що 1:1=1.

До поля GF(2) можна застосувати будь-які методи лінійної алгебри, зокрема матричні операції, а дії в двійкових кодах виконуються за модулем 2.

Двійковий код є лінійним, якщо сума за модулем 2 двох кодових слів також є кодовим словом цього коду. Крім того, лінійний двійковий код є груповим (група – множина математичних об'єктів, які можна додавати, віднімати), оскільки сукупність кодових комбінацій, що входять в нього, утворюють групу [28].

Лінійний двійковий блоковий систематичний (n, k)-код містить незмінну інформаційну частину завдовжки k символів і надмірну (перевірну), довжиною n-k символів (рис. 3.2).

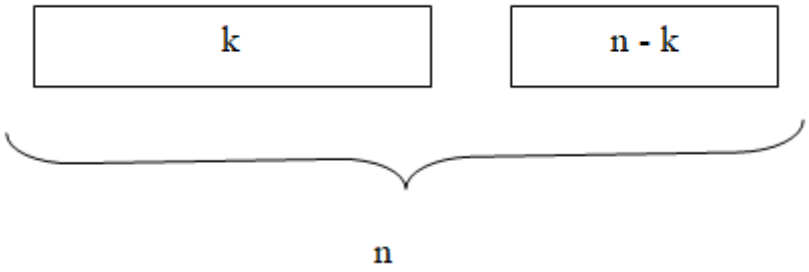

Рисунок 3.2 – Формат систематичного коду

Прості коди з перевіркою на парність (непарність) – високошвидкісні коди з поганими корегувальними характеристиками [24]. До k інформаційних біт додається (k+1)-й біт перевірки на парність, щоб загальна кількість одиниць в кодовому слові була парною *(parity)*. Це (k+1, k)-код або (n, n-1)-код. Для цього коду кодова відстань дорівнює двом, тому одиничні помилки тільки виявляються, але не виправляються.

Біт парності обчислюється як сума за модулем 2 інформаційних бітів:

$$\vec{U}_i = (u_1, u_2, ..., u_k). \tag{3.22}$$

$$u_{k+1} = u_1 + u_2 + ... + u_k, \tag{3.23}$$

де знак «+» – операція додавання за модулем 2;

$u_1, u_2, ..., u_k$ – інформаційні біти повідомлення;

u_{k+1} – біт перевірки на парність.

Якщо під час передачі послідовність $(u_1, u_2, ..., u_k, u_{k+1})$ довжини k+1, що містить парне число символів «1», виникне помилка в одному з символів, то кількість одиниць стане непарною і сума $u_1 + u_2 + ... + u_k + u_{k+1}$ буде дорівнювати одиниці, що і дозволяє виявляти виникнення одиничних помилок.

Прості коди з повторенням. Це низькошвидкісні коди з високими корегувальними характеристиками [24]. Інформаційний символ повторюється n раз (звичайно n непарне), тобто це (n,1)-коди. Наприклад (3,1)-код формується так:

$$0 – 000,$$
$$1 – 111.$$

Кодова відстань дорівнює 3, а кількість помилок, які може виправляти цей ко, дорівнює:

$$q_d = (n-1)/2 = (3-1)/2 = 1. \quad (3.24)$$

Коди Гемінга. Це такі лінійні коди [5]:

- коди з кодовою відстанню $d_{min}=3$ – виправляють всі одиничні помилки;
- коди з кодовою відстанню $d_{min}=4$ – виправляють всі одиничні помилки і виявляють подвійні.

Розглянемо перший випадок. Кількість надлишкових бітів для кодів Гемінга з виразу (3.7) така:

$$2^m \geq n+1, \quad (3.25)$$

де m=n-k – кількість перевірних бітів;
k – кількість інформаційних бітів.

Для знаходження перевірних бітів потрібно m незалежних перевірок на парність – перевірок, у яких ніяка сума за модулем 2 одних перевірок не збігається з будь-якою іншою перевіркою.

Після прийому закодованої послідовності, якщо написати символ «0» для перевірок, що виконались, і символ «1» для перевірок, що не виконалась, то отримаємо синдром (розпізнавач), який можна розглядати як число, що містить m двійкових символів і може розрізняти не більше 2^m подій. Якщо синдром S=0, то помилок не виявлено. Але синдром має також виявляти місце помилки у будь-якій з n позицій повідомлення. Саме ця ідея, лежить в основі кодів Гемінга. Тому перевірки на парність мають бути такі: у першу перевірку на парність входить кожна позиція, для якої остання цифра її номера, записаного в двійковому поданні, дорівнює «1»; в другу перевірку на парність мають входити позиції, у яких друга справа цифра її номера, записаного в двійковому поданні, дорівнює «1» і т. д. Тобто, перша перевірка на парність – позиції 1, 3, 5, 7, 9, 11, ... ; друга – 2, 3, 6, 7, 10, ... ; третя – 4, 5, 6, 7, 12, 13, 14, ...; четверта – 8, 9, 10, 11, 12, 13, 14, 15 і т. д.

Розглянемо приклад. Нехай кількість інформаційних символів k=4. Тоді з (3.25) знайдемо:

$$2^m \geq 5 + m.$$

Звідки кількість перевірних символів (також кількість незалежних перевірок на парність) m = 3, загальна кількість символів (інформаційних і перевірних) становитиме:

$$n = 4+3 = 7.$$

Позиції, що відповідають степеню «2», тобто 1, 2, 4 використовуються для перевірних символів, а позиції 3, 5, 6, 7 – інформаційні:

Позиція	1	2	3	4	5	6	7
Повідомлення	_	_	1	_	0	1	1.

Після першої перевірки на парність (позиції 1, 3, 5, 7) отримаємо перший перевірний символ, що дорівнює «0» (сума за модулем 2 символів, що знаходяться в позиціях 3, 5, 7):

$$a_1 \oplus a_3 \oplus a_5 \oplus a_7 = 0.$$

$$a_1 = a_3 \oplus a_5 \oplus a_7 = 1 + 0 + 1 = 0$$

Позиція	1	2	3	4	5	6	7
Повідомлення	0	_	1	_	0	1	1.

Після другої перевірки на парність (позиції 2, 3, 6, 7) отримаємо другий символ – «1»:

$$a_2 \oplus a_3 \oplus a_6 \oplus a_7 = 0.$$

$$a_2 = a_3 \oplus a_6 \oplus a_7 = 1 + 1 + 1 = 1$$

Позиція	1	2	3	4	5	6	7
Повідомлення	0	1	1	_	0	1	1.

Після третьої перевірки на парність (позиції 4, 5, 6, 7) отримаємо третій перевірний символ – «0»:

$$a_4 \oplus a_5 \oplus a_6 \oplus a_7 = 0.$$

$$a_4 = a_5 \oplus a_6 \oplus a_7 = 0 + 1 + 1 = 0$$

Позиція	1	2	3	4	5	6	7
Повідомлення	0	1	1	0	0	1	1.

Нехай під час передачі повідомлення виникла помилка в третьому символі зліва. Тобто прийнято повідомлення:

Позиція	1	2	3	4	5	6	7
Повідомлення	0	1	0	0	0	1	1.

Виконаємо перевірки на парність:

$$a_1 \oplus a_3 \oplus a_5 \oplus a_7 = 0+0+0+1 = 1;$$
$$a_2 \oplus a_3 \oplus a_6 \oplus a_7 = 1+0+1+1 = 1;$$
$$a_4 \oplus a_5 \oplus a_6 \oplus a_7 = 0+0+1+1 = 0$$

Синдром $S=011_2=3_{10}$. Змінюємо символ в третій позиції прийнятого повідомлення:

Позиція	1	2	3	4	5	6	7
Повідомлення	0	1	1	0	0	1	1.

Повідомлення відповідає переданому, тобто помилку виправлено.

Коди з відстанню $d_{min}=4$ отримують шляхом введення додаткової перевірки на парність всієї кодової комбінації і ще однієї позиції. Якщо синдром ненульовий, а додаткова перевірка на парність дорівнює «0», то виявлено подвійну помилку. Її виправити неможливо.

Загальні принципи побудови двійкового групового коду. Порівняно з кодом Гемінга, який є окремим випадком загальної задачі побудови групового коду, побудувати код з більшою виправною здатністю – задача складніша. Етапи побудови групового коду такі [2, 3]:

1. Визначити кількість перевірних символів згідно з (3.7) для заданої кратності помилок. Часто перевірних символів необхідно більше цієї теоретичної межі.

2. Скласти таблицю розпізнавачів (синдромів). Наприклад, для коду код Гемінга (7,4) таблицю розпізнавачів наведено в табл. 3.1, де вектор помилки указує розряд, в якому виникла помилка, а синдром є двійковим поданням номера цього розряду.

Таблиця 3.1 – Таблиця розпізнавачів для коду (7,4)

Вектор помилок	Розпізнавач (синдром)	Вектор помилок	Розпізнавач
0000001	001	0010000	101
0000010	010	0100000	110
0000100	011	1000000	111
0001000	100		

Розглянемо випадок виправлення одиничних і подвійних незалежних помилок. Як синдром для одиничних помилок в першому і другому розряді можна взяти комбінації 0..001 та 0...010. Однак, для одиничної помилки в третьому розряді комбінацію 0...011 брати не можна, оскільки така комбінація відповідає помилці одночасно в першому і другому

розрядах. Тому синдром одиничної помилки в третьому розряді буде 0...100.

Вектор помилки в першому і третьому розряді 0...0101 можна розглядати як результат сумарної дії двох векторів 0...0100 та 0...0001 і відповідно до цього йому може бути поставлено у відповідність розпізнавач, який подає суму за модулем 2 розпізнавачів цих помилок, тобто 0…0101. Аналогічно розпізнавач для вектора помилки 0...0110 є комбінацією 0...0110.

Розпізнавачем для одиничної помилки в четвертому розряді буде невикористана трирозрядна комбінація 111, а для векторів подвійних помилок в четвертому і молодших розрядах (0...01001, 0...01010, 0...01100) визначимо розпізнавачі як суму за модулем 2 розпізнавачів цих помилок аналогічно попереднім подвійним помилкам. Одержимо такі комбінації: 0...0110, 0...0101, 0...0011. Однак ці комбінації уже використовувались для векторів помилок 0...0110, 0...0101, 0...0011, тому комбінація 0...0111 не може бути використана як розпізнавач для помилки в четвертому розряді і для розпізнавача помилки в четвертому розряді необхідно взяти комбінацію 1000. Тобто, це складний процес із перевірками і поверненням за необхідності на попередні етапи, тому потрібно моделювання з використанням ЕОМ.

В табл. 3.2 наведено розпізнавачі одиничних помилок в перших 15 розрядах для коду, який виявляє і виправляє одиничні і подвійні помилки. Розпізнавачі подвійних помилок отримують як суми за модулем 2 розпізнавачів відповідних одиничних помилок.

Аналогічно визначаються таблиці розпізнавачів для помилок будь-якої кратності.

Таблиця 3.2 – Таблиця розпізнавачів для одиничних помилок

Номер розряда	Розпізнавач (синдром)	Номер розряда	Розпізнавач (синдром)	Номер розряда	Розпізнавач (синдром)
1	0000 0001	6	0001 0000	11	0110 1010
2	0000 0010	7	0010 0000	12	1000 0000
3	0000 0100	8	0011 0011	13	1001 0110
4	0000 1000	9	0100 0000	14	1011 0101
5	0000 1111	10	0101 0101	15	1101 1011

3. Визначити перевірні рівності аналогічно кодам Гемінга.

Циклічні коди. Циклічні коди – це різновид поліноміальних кодів [28], в яких елементи a_1, a_2, …a_{n-1} деякого кодового слова є коефіцієнтами полінома від x:

$$f(x) = a_0 + a_1 x + a_2 x^2 + ... + a_{n-1} x^{n-1}. \qquad (3.26)$$

Поліноміальний код – це множина всіх многочленів степені не більше ніж n-1, що містять як множник деякий фіксований многочлен g(x), який називається породжувальним многочленом. Процес кодування є результатом множення полінома m(x), що відповідає інформаційній послідовність, на породжувальний многочлен g(x), а декодування – це результат ділення на цей поліном.

Поліноміальні коди за деяких значень n мають властивість циклічності – циклічна перестановка деякого кодового слова приводить до кодового слова цього коду (див. п. 3.6).

Переваги циклічних кодів такі:

- операції кодування і декодування виконуються дуже просто з використанням регістрів зсуву;
- ці коди характеризуються алгебраїчною структурою, що дозволяє знайти простіші і ефективніші способи їх декодування.

На практиці, поряд з циклічними кодами в полі Галуа GF(2), знаходять широке використання циклічні коди з компонентами з розширених полів Галуа GF(2^m) – коди Боуза–Чоудхурі–Хоквінгема (БЧХ) і коди Ріда–Соломона (РС) [19, 23 – 24], зокрема, коди РС використовуються в програвачах компакт-дисків [19].

Основні властивості циклічних (n,k)-кодів такі [28]:

1. Кожен ненульовий поліном має мати степінь, принаймні (n-k), але не більше n-1.
2. Існує тільки один кодовий поліном степені (n-k), який називається породжувальним поліномом коду:
$$g(x)=1+g_1x+g_2x^2+...+g_{n-k}x^{n-k}, \quad (3.27)$$
3. Кожен поліном u(x), що відповідає закодованим даним, є кратним породжувальному поліному g(x), оскільки:
$$u(x)=m(x)\cdot g(x). \quad (3.28)$$
4. Породжувальний поліном циклічного коду ділить без залишку двочлен x^n+1, тобто $x^n+1=g(x)h(x)+0$. Поліном h(x) – частка від ділення x^n+1 на g(x) – називається перевірним поліномом. Оскільки h(x) і g(x) однозначно пов'язані, то за допомогою перевірного полінома h(x) теж можна проводити кодування, тобто він також визначає код.

Якщо просто помножити інформаційний поліном на перевірний в полі GF(2) згідно з властивістю 3, то отриманий код хоча і придатний для кодування, але не є систематичним.

Для побудови систематичного циклічного коду помножимо інформаційний поліном на x^{n-k} і як результат отримаємо поліном n-1 степеня або менше:

$$x^{n-k}m(x) = m_0x^{n-k} + m_1x^{n-k+1} + ... + m_{k-1}x^{n-1}. \quad (3.29)$$

Далі, згідно з теоремою про ділення поліномів, отримаємо:

$$x^{n-k}m(x) = q(x)g(x) + p(x), \quad (3.30)$$

де q(x) і p(x) – частка і залишок від ділення полінома на породжувальний поліном g(x).

Степінь p(x) має бути (n-k-1) або менше, оскільки степінь g(x) дорівнює (n-k):

$$p(x) = p_0 + p_1x + ... + p_{n-k-1}x^{n-k-1}. \quad (3.31)$$

З виразу (3.30) маємо:

$$p(x) + x^{n-k}m(x) = q(x)g(x). \quad (3.32)$$

Тобто, поліном $p(x) + x^{n-k}m(x)$ – це кодовий поліном, відповідний кодованій інформаційній послідовності т(x), він кратний g(x) і має степінь n-1 або менший.

З виразу (3.32) отримаємо:

$$\begin{aligned} p(x) + x^{n-k}m(x) = p_0 + p_1x + ... + p_{n-k-1}x^{n-k-1} + \\ + m_0x^{n-k} + m_1x^{n-k+1} + ... + m_{k-1}x^{n-1}. \end{aligned} \quad (3.33)$$

А кодове слово таке:

$$U = (p_0, p_1, ...p_{n-k-1}, m_0, m_1, ...m_{k-1}), \quad (3.34)$$

де $p_0, p_1, ...p_{n-k-1}$ – перевірні символи;

$m_0, m_1, ...m_{k-1}$ – інформаційні символи.

Декодування циклічних кодів виконується як із застосуванням алгебраїчних методів, так і неалгебраїчних [28]. Найчастіше використовуються алгебраїчні методи декодування, зокрема, синдромне декодування циклічних кодів.

Нехай u(x) і r(x) – поліноми, що відповідають закодованій послідовності і прийнятій з каналу зв'язку. Якщо поділити r(x) на g(x), то отримаємо:

$$r(x)=q(x)g(x)+s(x), \quad (3.35)$$

де *q(x)* – частка від ділення;

s(x) – залишок від ділення.

Якщо *s(x)*=0, то кодовану послідовність прийнято без помилок, інакше у прийнятій послідовності наявні помилки. Тобто s(x) синдром прийнятої

послідовності, який визначає наявність і місце помилки. Синдром s(x) можна подати так:

$$S(x) = s_0 \cdot x^0 + s_1 \cdot x^1 + s_2 \cdot x^2 + ... + s_{n-k-1} \cdot x^{n-k-1}. \qquad (3.36)$$

Схема обчислення синдрому є схемою ділення.

Згортні коди. Згортні коди – це лінійні деревоподібні коди [22 – 25]. Згортні коди, завдяки прості реалізації, відіграють провідну роль в сучасних системах зв'язку. Відмінності згортних кодів від блокових такі:

1. Згортні коди виконують кодування і декодування потоків даних безперервно в часі.
2. Згортні коди не потребують блокової синхронізації.
3. Згортні коди дозволяють досягти високої надійності передачі інформації.
4. Згортні коди знаходять шляхом моделювання [22].

Ці коди називають згортними, оскільки кодер розглядається як цифровий фільтр, який виконує згортку послідовності інформаційних символів з імпульсною характеристикою кодера (реакція кодера на одиничний імпульс (1000...)).

Кодера згортного коду містить k_0 регістрів зсуву, зв'язки між елементами яких визначаються набором породжувальних поліномів $g_{i,j}(x)$, де i=0, 1 ... k_0-1 – номер вхідного потоку, а j = 0, 1 ..., n_0-1 – номер вихідного потоку. На практиці найчастіше використовуються коди з одним вхідним потоком (k_0=1) і на відміну від блокових кодів, які описуються єдиним породжувальним поліномом, згортний код потребує декілька породжувальних поліномів.

Згортні коди можна подати у несистематичній і систематичній формі, окремим випадком якої є рекурсивні систематичні згортні (Recursive Systematic Convolutional – RSC) коди.

Розглянемо кодер несистематичного згортного коду (рис. 3.6), заданий породжувальними поліномами:

$$G_0(x)=1+x^2;$$
$$G_1(x)= 1 + x + x^2 .$$

Параметри цього коду такі: кількість інформаційних гілок k_0=1, кількістю перевірних гілок n_0=2, кодова швидкість r=1/2, довжина кодового обмеження n_A=6, конструктивна довжина коду K=3.

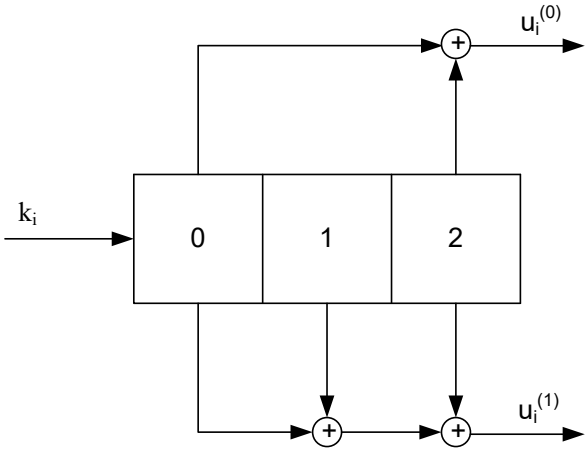

Рисунок 3.3– Кодер згортного несистематичного коду

Відомі різні способи опису згортних кодів, наприклад з використанням породжувальної матриці, але більш зручним способом задання згортних кодів є задання з використанням імпульсної характеристики або породжувальних поліномів (поліноміальне подання).

Наприклад, для кодера на рис. 3.3 імпульсна характеристика така:

$$g=(11\ 10\ 11\ 00\ 00\ \ldots).$$

А породжувальні поліноми такі:

$$\left.\begin{aligned} G^1(x) &= 1 \oplus x \oplus x^2 \\ G^2(x) &= 1 \oplus x^2 \end{aligned}\right\}. \qquad (3.37)$$

Запишемо коефіцієнти поліномів у двійковій формі:

$$G^1(x) = 111$$
$$G^2(x) = 101.$$

У вісімковій формі запису:

$$G^1(x) = 7,$$
$$G^2(x) = 5,$$

або G=(7,5).

Процес кодування – це результат множення многочлена вхідної послідовності на породжувальний многочлен коду:

$$u_i(x) = k(x) \cdot G_i(x). \qquad (3.38)$$

Згортний кодер – це автомат з кінцевим числом станів і може бути описаний діаграмою станів.

Для кодера G=(7,5) з K=3 діаграму станів наведено на рис. 3.4, стан кодера визначають K-1 розрядів регістра зсуву, починаючи від виходу кодера. Діаграма описує всі можливі переходи кодера з одного стану в інший, а також містить значення виходів кодера, які супроводжують ці переходи. Чотири можливих стани кодера такі: S_1S_2=00, 10, 11, 01. Якщо на вхід кодера надходить інформаційний символ «0», то перехід позначається суцільною лінією, а пунктирними – символу «1».

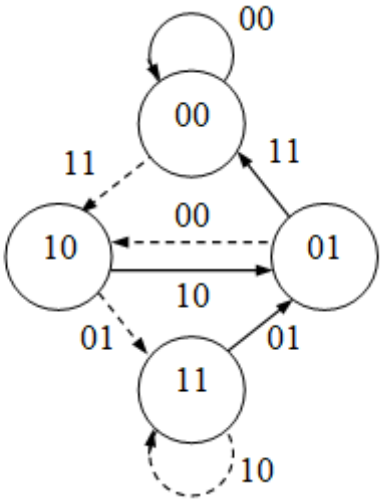

Рисунок 3.4 – Діаграма станів кодера

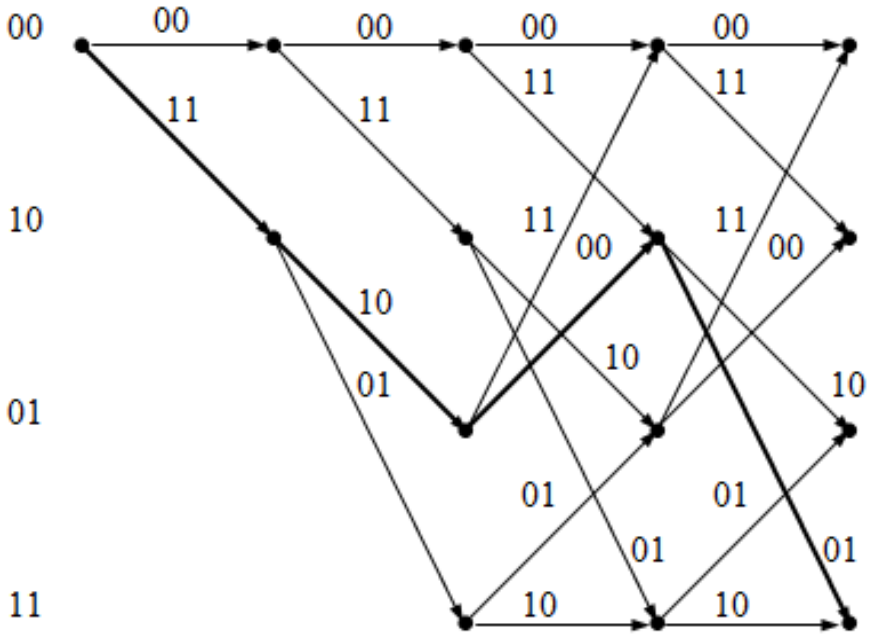

Рисунок 3.5 – Ґратчаста діаграма згортного коду

Якщо розгорнути діаграму станів в часі, то отримаємо ґратчасту діаграму (рис. 3.5). На гратці стани – це вузли, а переходи – лінії, що їх з’єднують. Перехід з одного стану в інший супроводжується зсувом на

один крок вправо. Діаграма наглядно показує всі дозволені шляхи, по яких просувається кодер під час кодування. Кожній інформаційній послідовності на вході кодера відповідає єдиний шлях на ґратці.

Кодування з використанням ґратчастої діаграми – проста процедура: черговий символ вхідної послідовності визначає напрям руху з вузлів ґратки: «0» – по верхньому ребру, «1» – по нижньому ребру. Наприклад, вхідна послідовність k=(1011…), а шлях по ґратчастій діаграмі (жирна лінія) 11 10 00 01….

Ґратчаста діаграма, починаючи з третього кроку, має постійну ширину, що є її перевагою перед ще одним способом подання згортних кодів у вигляді кодового дерева, у якого експоненціально зростає кількість гілок зі зростанням довжини вхідної послідовності.

З використанням ґратки можна визначити поняття вільної відстані d_{free}. Вільна відстань d_{free} – це мінімальна відстань, за Гемінгом, між двома різними шляхами по ґратці, які починаються і закінчуються в одному стані. Мінімальна вільна відстань для цього коду дорівнює «5» – загальна вага шляху $0\to1\to2\to0$, заданого вершинами на діаграмі.

Породжувальні многочлени згортного коду мають задовольняти таку рівність:

$$\gcd(g_1(x), g_2(x), ..., g_{n_0}(x)) = x^n, \qquad (3.39)$$

де gcd – найбільший спільний дільник (greatest common divisor);

$g_1(x), g_2(x), ..., g_{n_0}(x)$ – породжувальні многочлени коду, $x^n = 1$.

Тобто, вони є взаємнопростими і задача знаходження згортного коду є задачею пошуку взаємнопростих породжувальних многочленів.

Коди, які не задовольняють цю умову, називаються катастрофічними і не можуть використовуватись для побудови згортних кодів.

Однак, навіть взаємно прості многочлени не гарантують необхідної виправної здатності, тому оптимальні коди знаходять шляхом комп'ютерного моделювання. У літературі [19] наведено численні таблиці породжувальних многочленів оптимальних згортних кодів.

Декодування згортних кодів. Відомо два підходи до декодування згортних кодів: алгебраїчні та імовірнісні методи [22, 28 – 30].

Алгебраїчні методи (синдромне та мажоритарне (порогове) декодування) ґрунтуються на використанні алгебраїчних властивостей кодових послідовностей. Хоча деколи такі методи забезпечують просту реалізацію кодеків, однак вони не є оптимальними, оскільки алгебраїчні процедури декодування орієнтовані на виправлення конкретних конфігурацій помилок в каналі.

Імовірнісні методи декодування ближчі до оптимального прийому, оскільки декодер оцінює і порівнює імовірності різних гіпотез та виносить рішення про передані символи.

Типовим алгоритмом декодування з використанням імовірнісних характеристик є алгоритм Вітербі, який є алгоритмом максимальної правдоподібності.

Алгоритм Вітербі. Декодування виконується з використанням ґратчастої діаграми. Введемо такі поняття [30]:

- метрика гілки (МГ) – відстань Гемінга між набором символів $x^{(1)}x^{(2)}$ на вході декодера і набором символів $a^{(1)}a^{(2)}$, що відповідають цій гілці на ґратчастій діаграмі;
- метрика шляху (МШ) – сума метрик гілок, що утворюють цей шлях на ґратчастій діаграмі;
- метрика стану (МС) – дорівнює метриці шляху, що закінчується в цьому стані.

В кожному стані ґратчастої діаграми на кожному кроці декодування виконуються такі операції:

- додавання метрик попередніх станів до метрик відповідних гілок;
- в кожному стані порівнюються метрики шляхів, що входять в цей стан, і вибирається шлях з меншою метрикою (шлях, що вижив), величина якої використовується як метрика цього стану. Якщо метрики шляхів однакові, то вибір одного з двох шляхів виконується випадковим чином.

Коли шляхи, що вижили, зливаються, приймається рішення про передану послідовність. Глибина, на якій зливаються шляхи, величина випадкова, залежить від кратності та імовірності помилок. Часто на практиці не чекають злиття шляхів, а встановлюють фіксовану глибину декодування в діапазоні $n_A<b\le n_A+q$, q – кратність помилок, що виправляються цим кодом. Оскільки пам’ять декодера не нескінченна, то алгоритм Вітербі не є строго оптимальним.

Розглянемо приклад з використанням ґратчастої діаграми (рис. 3.5) для кодера G=(7, 5). Нехай передано закодоване послідовність U = (00000…), а прийнято r = (10000…), тобто в першому кадрі кодового слова виникла помилка.

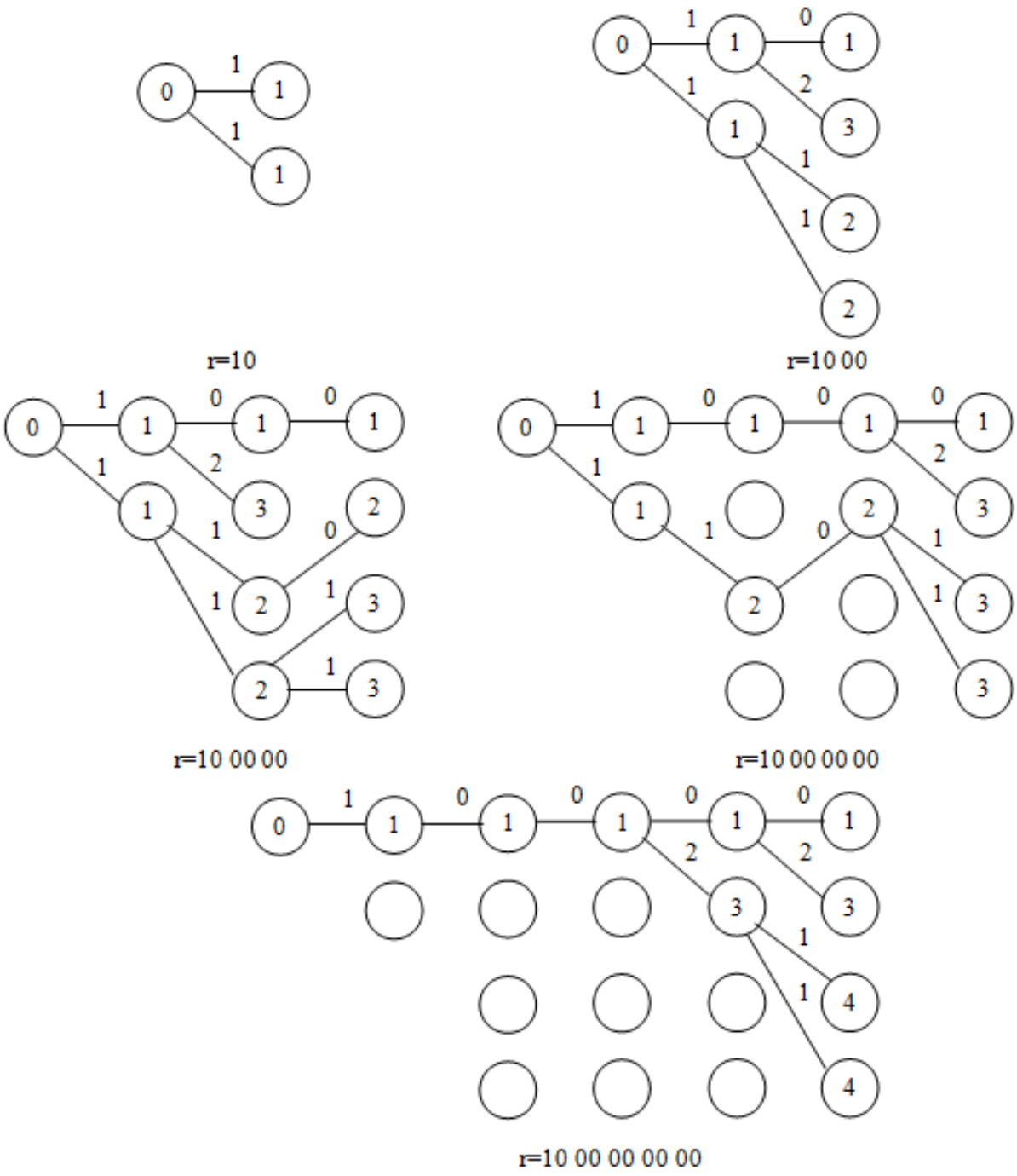

Рисунок 3.6 – Декодування методом Вітербі прийнятої послідовності

В початковий момент часу декодер знаходиться в стані 00 з метрикою цього стану МС (00) = 0.

Для вхідної послідовності r=10 метрики гілок 00 і 11, що виходять з цього стану, будуть МГ(00)=1 і МГ(11)=1. Оскільки метрика попереднього стану МС(00) = 0, то МС(00)=1 і МС(10)=1 (рис. 3.6). Далі на вхід декодера надходить пара символів 00, тобто прийнято послідовність, що дорівнює r=10 00, тоді метрики станів визначаються як сума метрик гілок, що входять в ці стани, і метрик попередніх станів:

МС (00) = 1 + 0 = 1;
МС (10) = 2 + 1 = 3;
МС (01) = 1 + 1 = 2;
МС (00) = 1 + 1 = 2.

Починаючи з рівня три, потрібно у кожному стані вибирати шлях з найменшою метрикою, оскільки у цей стан входять два шляхи. Шляхи, що вижили під час прийому, r=10 00 00, r=10 00 00 00, r=10 00 00 00 00 наведено на рис. 3.6. Якщо шляхи, що входили в поточний стан, мали однакову метрику, то завжди вибирався верхній шлях на ґратчастій діаграмі.

На п’ятому кроці декодування і приймається рішення, що перші три переданих символи – 000, оскільки перші 3 ребра всіх шляхів, що вижили, збігаються.

Хоча уже на кроці 4 (r=10 00 00 00) можна було б обмежити глибину декодування, оскільки відмінності метрик правильного і неправильного шляхів достатньо великі: $МШ_{пр}$=1, а $МШ_{пом}$=3-4.

3.2 Приклади розв’язування задач

Приклад 3.1. Визначити кодову відстань d_{min} для такого коду: A_1=0000, A_2=0011, A_3=1100, A_4=1010.

Розв’язання. Знайдемо суми за модулем 2 всіх можливих кодових комбінацій і значення відстані між ними як кількість одиниць в отриманій сумі:

0000	0000	0000
+	+	+
0011	1100	1010
_____	_____	_____
0011	1100	1010
$d_{12} = d_{21} = 2$	$d_{13} = d_{31} = 2$	$d_{14} = d_{41} = 2$
0011	0011	1100
+	+	+
1100	1010	1010
_____	_____	_____
1111	1001	0110
$d_{23} = d_{32} = 4$	$d_{24} = d_{42} = 2$	$d_{34} = d_{43} = 2$

Таким чином, найменшим значенням d_{ij} є 2, тобто кодова відстань для цього коду така: $d_{min}=2$.

Приклад 3.2. Нехай передається повідомлення, що складається з чотирьох символів A_1, A_2, A_3, A_4, які подано такими кодовими комбінаціями $A_1 = 00$, $A_2 = 01$, $A_3 = 10$, $A_4 = 11$. Побудувати код з перевіркою на парність.

Розв'язання. Згідно з (3.23) сформуємо біт перевірки на парність для кожної кодової комбінації:

$$A_1 - 0 \oplus 0 = 0,$$
$$A_2 - 0 \oplus 1 = 1,$$
$$A_3 - 1 \oplus 0 = 1,$$
$$A_4 - 1 \oplus 1 = 0.$$

До заданих k = 2 інформаційних бітів дописується (k+1)-й додатковий біт перевірки на парність, отримаємо новий код:

$$A_1 = 0\ 0\ 0,$$
$$A_2 = 0\ 1\ 1,$$
$$A_3 = 1\ 0\ 1,$$
$$A_4 = 1\ 1\ 0.$$

Визначимо кодову відстань d_{min} для заданого коду. Для цього визначимо кількість одиниць в сумі за модулем 2 всіх пар кодових комбінацій:

000	000	000	011	011	101
+	+	+	+	+	+
011	101	110	101	110	110
———	———	———	———	———	———
011	101	110	110	101	011
$d_1 = 2$	$d_2 = 2$	$d_3 = 2$	$d_4 = 2$	$d_5 = 2$	$d_6 = 2$

Таким чином, $d_{min}=2$ і заданий код може виявляти лише одну помилку в кодовій комбінації (блоці). Якщо виникне дві помилки в межах одного блока, то вони переведуть помилкову комбінацію в іншу дозволену комбінацію з парною кількістю одиниць.

Приклад 3.3. Побудувати твірну матрицю і визначити всі комбінації двійкового систематичного (групового) коду, здатного виправляти одиничні помилки для $N_0 = 8$ повідомлень.

Розв'язання. Кількість інформаційних розрядів коду $k = \log_2 8 = 3$. Кількість перевірних розрядів визначається як найменше ціле r, яке

задовольняє нерівності $2^r \geq k + r + 1$; таким значенням буде $r = 3$. Довжина коду $n = k + r = 6$. Таким чином, твірна матриця $G_{n,k}$ має 6 стовпців та 3 рядки, а перевірна підматриця Cr,k має 3 стовпці та 3 рядки.

Згідно з правилом побудови підматриці $C_{r,k}$ кількість одиниць у кожному рядку цієї підматриці має бути не менша за $d_{min} - 1 = = 3 - 1 = 2$, а кодова відстань між окремими рядками цієї підматриці – не менша за $d_{min} - 2 = 3 - 2 = 1$. Тому з триелементних комбінацій для підматриці $C_{3,3}$ вибираємо тільки ті, які задовольняють ці умови, тобто 110, 101, 011.

За інформаційну підматрицю I_k твірної матриці обирають одиничну підматрицю. Дописавши до неї перевірну підматрицю, одержимо твірну матрицю систематичного коду, здатного виправляти однократні помилки:

$$G_{6,3} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

За допомогою одержаної твірної матриці $G_{6,3}$ визначимо всі 8 кодових комбінацій, які належать до цього систематичного коду: 1 – 000000; 2 − 100110; 3 – 010101; 4 – 001011; 5 – 110011 (2 ⊕ 3); 6 – 101101 (2 ⊕ 4); 7 – 011110 (3 ⊕ 4); 8 – 111000 (2 ⊕ 3 ⊕ 4) [12].

Приклад 3.4. Побудувати перевірну матрицю двійкового систематичного коду, здатного виправляти однократні помилки з $d_{min} = 3$. Закодувати за допомогою одержаної перевірної матриці комбінації первинного двійкового коду 111 та 011.

Розв'язання. Для побудови перевірної матриці систематичного коду, здатного виправляти однократні помилки, скористаємось твірною матрицею, побудованою для одержання 8 комбінацій систематичного коду в прикладі 3.3.

Потрібно, щоб перевірна матриця $H_{n,r}$ мала $r = 3$ рядки та $n = 6$ стовпців. Вона утворюється з двох підматриць: $D_{3,3}$, що містить три стовпці і три рядки, кожний рядок якої відповідає стовпцю перевірної підматриці $C_{3,3}$ твірної матриці $G_{6,3}$, та одиничної підматриці I_3.

Таким чином:

$$H_{6,3} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Перевірні елементи, відповідно до матриці $H_{6,3}$, можна визначити так:

$$b1 = a1 \oplus a2; \quad b2 = a1 \oplus a3; \quad b3 = a2 \oplus a3.$$

За допомогою одержаної перевірної матриці $H_{6,3}$ виконуємо кодування систематичним (груповим) кодом комбінацій первинного коду 111 та 011, для чого визначаємо перевірні елементи для заданих комбінацій. Для комбінації 111:

$$b1 = 1 \oplus 1 = 0\ ;\ b2 = 1 \oplus 1 = 0\ ;\ b3 = 1 \oplus 1 = 0,$$

а для комбінації 011:

$$b1 = 0 \oplus 1 = 1\ ;\ b2 = 0 \oplus 1 = 1;\ b3 = 1 \oplus 1 = 0.$$

Таким чином, кодові комбінації систематичного (групового) коду будуть мати вигляд: 111000 та 011110.

Приклад 3.5. Для групового (7, 4) коду, що виправляє однократні помилки, побудувати перевірну матрицю $H_{7,3}$ і закодувати за її допомогою комбінацію двійкового простого коду 1101, якщо твірна матриця має вигляд:

$$G_{7,4} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}.$$

Визначити синдром для виправлення однократних помилок в комбінаціях цього коду. Показати на прикладі виправлення однократної помилки.

Розв'язання. Згідно з (3.17) перевірна матриця $H_{7,3}$ для (7, 4) – коду буде складатись з двох підматриць: $D_{4,3}$, кожний рядок якої відповідає транспонованому стовпцю перевірної підматриці $C_{3,4}$ твірної матриці $G_{7,4}$, та одиничної підматриці I_3. Отже, перевірна матриця $H_{7,3}$ буде мати вигляд:

$$H_{7,3} = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Перевірні елементи, згідно з матрицею $H_{7,3}$, будуть визначатись за такими виразами:

$$b1 = a1 \oplus a2 \oplus a3\ ;\ \ b2 = a1 \oplus a2 \oplus a4\ ;\ \ b3 = a1 \oplus a3 \oplus a4\ .$$

Користуючись ними, закодуємо комбінацію [a1 a 2 a 3 a 4] = 1101, тобто визначимо перевірні елементи:

$$b1 = 1 \oplus 1 \oplus 0 = 0\ ;\ b2 = 1 \oplus 1 \oplus 1 = 1\ ;\ b3 = 1 \oplus 0 \oplus 1 = 0.$$

Таким чином, комбінація групового коду буде мати вигляд 1101010 .

У декодері для виявлення і виправлення однократної помилки у прийнятій кодовій комбінації систематичного групового коду виконують перевірку – визначають синдром помилки. Для одержаної перевірної матриці елементи синдрому помилки визначаються таким чином :

$$s1 = a_1^* \oplus a_2^* \oplus a_3^* \oplus b_1^*;\ s2 = a_1^* \oplus a_2^* \oplus a_4^* \oplus b_2^*;$$

$$s3 = a_1^* \oplus a_3^* \oplus a_4^* \oplus b_3^*.$$

Знайдемо і виправимо однократну помилку, наприклад, у комбінації $[\,a_1^* a_2^* a_3^* a_4^* b_1^* b_2^* b_3^*\,] = 1001010$.

Для цього визначимо кодовий синдром помилки: $s1 = 1 \oplus 0 \oplus 0 \oplus 0 = 1$; $s2 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$; $s3 = 1 \oplus 0 \oplus 1 \oplus 0 = 0$, тобто синдром має вигляд 110, що відповідає другому стовпцю перевірної матриці $H_{7,3}$. Синдром показує, що помилка знаходиться у другому розряді прийнятої кодової комбінації. Для виправлення помилки інвертуємо значення цього розряду, тобто замість «0» записуємо «1». Виправлена кодова комбінація групового коду буде мати вигляд 1101010 .

Приклад 3.6. Закодувати традиційним двійковим кодом Гемінга комбінацію двійкового простого коду 10110 і показати на прикладі процес виправлення будь-якої однократної помилки. Визначити надмірність коду.

Розв'язання. Згідно із співвідношенням (3.25) за $k = 5$ кількість перевірних елементів $r = 4$; довжина коду $n = k + r = 5 + 4 = 9$. Перевірні елементи будуть розташовані на позиціях 1, 2, 4, і 8. Побудуємо перевірну матрицю коду Гемінга розмірами $r = 4$ рядків та $n = 9$ стовпців:

$$H_{9,4} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

$$b1\ \ b2\ \ a1\ \ b3\ \ a2\ \ a3\ \ a4\ \ b4\ \ a5$$

Під матрицею для полегшення процесу кодування записано у загальному вигляді кодову комбінацію, де через a_i та b_j позначено інформаційні та перевірні елементи, відповідно. Користуючись побудованою перевірною матрицею $H_{9,4}$, визначимо значення перевірних елементів для [a1 a 2 a 3 a 4 a 5] = 10110:

$$b1 = a1 \oplus a2 \oplus a4 \oplus a5 = 1 \oplus 0 \oplus 1 \oplus 0 = 0;$$
$$b2 = a1 \oplus a3 \oplus a4 = 1 \oplus 1 \oplus 1 = 1;$$
$$b3 = a2 \oplus a3 \oplus a4 = 0 \oplus 1 \oplus 1 = 0;$$
$$b4 = a5 = 0.$$

Кодова комбінація традиційного коду Гемінга буде мати вигляд: 011001100.

Виконаємо декодування одержаної кодової комбінації з виправленням однократної помилки. Припустимо, що під час передавання сталося спотворення і замість 011001100 було прийнято кодову комбінацію 011001000.

Для виявлення і виправлення помилки у декодері виконують перевірки на парність з урахуванням перевірних елементів, тобто знаходять синдром помилки згідно з перевірною матрицею $H_{9,4}$:

$$s\,1 = b\,1 \oplus a\,1 \oplus a\,2 \oplus a\,4 \oplus a\,5 = 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 = 1;$$
$$s\,2 = b\,2 \oplus a\,1 \oplus a\,3 \oplus a\,4 = 1 \oplus 1 \oplus 1 \oplus 0 = 1;$$
$$s\,3 = b\,3 \oplus a\,2 \oplus a\,3 \oplus a\,4 = 0 \oplus 0 \oplus 1 \oplus 0 = 1;$$
$$s\,4 = b\,4 \oplus a\,5 = 0 \oplus 0 = 0.$$

Маємо синдром 0111. Таким чином, визначаємо, що спотворено елемент із порядковим номером $0111_2 = 7_{10}$, тобто елемент а4. Виправляємо його за допомогою інверсії та одержуємо правильну кодову комбінацію – 011001100.

Надмірність коду $R = 1 - k / n = 1 - k / (k + r) = r / n = 4 / 9$.

Приклад 3.7. Використовуючи табл. 3.2 знайти правила побудови коду (8,2), який виправляє всі одиничні та подвійні помилки.

Розв'язання. З табл. 3.2 видно, що за восьми розрядів (n=8) кількість перевірних символів – шість і відповідні перевірні рівності такі:

$$a_1 \oplus a_5 \oplus a_8 = 0,$$
$$a_2 \oplus a_5 \oplus a_8 = 0,$$
$$a_3 \oplus a_5 = 0,$$
$$a_4 \oplus a_5 = 0,$$
$$a_6 \oplus a_8 = 0,$$
$$a_7 \oplus a_8 = 0.$$

Звідки визначимо перевірні символи:

$$a_1 = a_5 \oplus a_8,$$
$$a_2 = a_5 \oplus a_8,$$
$$a_3 = a_5,$$
$$a_4 = a_5,$$
$$a_6 = a_8,$$
$$a_7 = a_8.$$

Для заданого коду $d_{min}=5$ і він, згідно з виразами (3.2, 3.3), може виправляти одиничні і подвійні помилки та виявляти помилки кратності від 1 до 4.

Приклад 3.8. З використанням циклічного коду, що задається породжувальним поліномом $g(x)=1+x+x^3$, закодувати послідовність m = (0111).

Розв'язання. Послідовності m =(0111) відповідає поліном $m(x)=x+x^2+x^3$. Помножимо m(x) на x^{n-k}:

$$x^{n-k}m(x)=m(x)x^3=x^3(x+x^2+x^3)=x^4+x^5+x^6.$$

Розділимо $m(x)x^{n-k}$ на породжувальний поліном $g(x)$:

$$\begin{array}{ll} x^6+x^5+x^4 & \big|\underline{x^3+x\ \ +1} \\ \underline{x^6+0+x^4+x^3} & \quad x^3+x^2=q(x) \\ \quad x^5+0+x^3 & \\ \quad \underline{x^5+0+x^3+x^2} & \\ \qquad\qquad x^2=p(x) & \end{array}$$

Таким чином, кодовий поліном для інформаційної послідовності m= (0111), матиме такий вигляд:

$$u(x)=0\cdot x^0+0\cdot x^1+1\cdot x^2+0\cdot x^3+1\cdot x^4+1\cdot x^5+1\cdot x^6,$$

а відповідне кодове слово U = (0010111).

Код з породжувальним поліномом $g(x)=1+x+x^3$, який розглянуто в прикладі, називається циклічним (7,4) кодом Гемінга (цей код має властивість циклічності).

Приклад 3.9. Визначити згортку двійкових послідовностей g(m)=(1,0,1,1) та k(n)=(1,0,1), де g(m) – імпульсний відгук кодера, а k(n) – інформаційна послідовність.

Розв'язання. Згортка задається таким виразом:

$$u(n)=k(n)\otimes g(n)=\sum_{m=0}^{m}g(m)k(n-m).$$

Оскільки g(m) містить чотири двійкових символи, то m = 0, 1, 2, 3, а вираз для u(n) буде таким:

$$u(n) = k(n) \otimes g(n) = \sum_{m=0}^{3} g(m)k(n-m).$$

Знайдемо елементи u(n):

$$u(0) = g(0)\cdot k(0) \oplus g(1)\cdot k(0-1) \oplus g(2)\cdot k(0-2) \oplus g(3)\cdot k(0-3) = 1\cdot 1 \oplus 0\cdot 0 \oplus 1\cdot 0 \oplus 1\cdot 0 = 1;$$

$$u(1) = g(0)\cdot k(1) \oplus g(1)\cdot k(0) \oplus g(2)\cdot k(-1) \oplus g(3)\cdot k(-2) = 1\cdot 0 \oplus 0\cdot 1 \oplus 1\cdot 0 \oplus 1\cdot 0 = 0;$$

$$u(2) = g(0)\cdot k(2) \oplus g(1)\cdot k(1) \oplus g(2)\cdot k(0) \oplus g(3)\cdot k(-1) = 1\cdot 1 \oplus 0\cdot 0 \oplus 1\cdot 1 \oplus 1\cdot 0 = 0;$$

$$u(3) = g(0)\cdot k(3) \oplus g(1)\cdot k(2) \oplus g(2)\cdot k(1) \oplus g(3)\cdot k(0) = 1\cdot 0 \oplus 0\cdot 1 \oplus 1\cdot 0 \oplus 1\cdot 1 = 1;$$

$$u(4) = g(0)\cdot k(4) \oplus g(1)\cdot k(3) \oplus g(2)\cdot k(2) \oplus g(3)\cdot k(1) = 1\cdot 0 \oplus 0\cdot 0 \oplus 1\cdot 1 \oplus 1\cdot 0 = 1;$$

$$u(5) = g(0)\cdot k(5) \oplus g(1)\cdot k(4) \oplus g(2)\cdot k(3) \oplus g(3)\cdot k(2) = 1\cdot 0 \oplus 0\cdot 0 \oplus 1\cdot 0 \oplus 1\cdot 1 = 1.$$

Приклад 3.10. Закодувати інформаційну послідовність k=(1011100…) з використанням імпульсної характеристики g=(11 10 11 00 00 …).

Розв'язання. Кодування виконується шляхом зсуву імпульсної характеристики на два біти вправо для кожного інформаційного символу із заповненням вільних бітів нулями і додаванням за модулем 2 результатів зсуву, відповідних одиничним символам вхідної послідовності:

$$\begin{array}{l} U = 11\ \ 10\ \ 11\ \ 00\ \ 00\ \ 00 \\ \qquad\quad 11\ \ 10\ \ 11\ \ 00\ \ 00\ \ 00 \\ \qquad\qquad\quad 11\ \ 10\ \ 11\ \ 00\ \ 00\ \ 00 \\ \qquad\qquad\qquad\quad 11\ \ 10\ \ 11\ \ 00\ \ 00\ \ 00 \\ \hline \qquad 11\ \ 10\ \ 00\ \ 01\ \ 10\ \ 01\ \ 11\ \ 00\ \ 00\ \ 00\ \ ... \end{array}$$

Приклад 3.11. Закодувати інформаційну послідовність k=(1011100…) з використанням згортного кодера G=(7,5).

Розв'язання. Процес кодування може бути поданий як результат множення многочлена вхідної послідовності на породжувальні многочлени коду. Для заданого кодера породжувальні многочлени такі:

$$G^1(x) = 1 \oplus x \oplus x^2$$
$$G^2(x) = 1 \oplus x^2.$$

Многочлен вхідної послідовності такий: $k(x) = 1 \oplus x^2 \oplus x^3 \oplus x^4$.

Тоді отримаємо:

$$u_1(x) = k(x) \cdot G^1(x) = (1 \oplus x^2 \oplus x^3 \oplus x^4) \cdot (1 \oplus x \oplus x^2) = 1 \oplus x \oplus x^4 \oplus x^6;$$
$$u_2(x) = k(x) \cdot G^1(x) = (1 \oplus x^2 \oplus x^3 \oplus x^4) \cdot (1 \oplus x^2) = 1 \oplus x^3 \oplus x^5 \oplus x^6,$$

що відповідає кодовим послідовностям:

$$U_1 = 1 \quad 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \ ...$$

$$U_2 = 1 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \ ...$$

$$U = 11 \ \ 10 \ \ 00 \ \ 01 \ \ 10 \ \ 01 \ \ 11 \ \ 00 \ \ 00...$$

Тобто, аналогічно попередньому прикладу.

3.3 Контрольні питання і задачі

1. Основні види та причини завад в комп'ютерних мережах.
2. Які способи боротьби з випадковими завадами?
3. Сформулюйте та поясніть теорему Шеннона про кодування каналу із завадами.
4. Наведіть основні принципи завадостійкого кодування.
5. Наведіть класифікацію завадостійких кодів.
6. Наведіть граничні співвідношення між параметрами завадостійких кодів.
7. Охарактеризуйте блокові корегувальні коди.
8. Які способи подання лінійних блокових кодів?
9. Що таке породжувальна і перевірна матриці?
10. Поясніть такі поняття як породжувальний і перевірний поліноми.
11. Дайте характеристику деревоподібних кодів.
12. Які Вам відомі алгоритми декодування згортних кодів?
13. Охарактеризуйте алгоритм декодування Вітербі.

Задачі

1. Визначити кодову відстань d_{min} для такого коду: A_1=000000, A_2=000111, A_3=111000, A_4=111111. Відповідь: d_{min}=3.
2. Скільки помилок може виявляти і виправляти такий код: A_1=000000, A_2=000111, A_3=111000, A_4=111111. Відповідь: виявляє 2 помилки, виправляє 1 помилку.
3. Нехай передається повідомлення, що складається з восьми символів A_1, A_2, A_3 , A_4, A_5, A_6, A_7, A_8, які подано такими кодовими комбінаціями A_1 = 000, A_2 = 001, A_3 = 010, A_4 = 011, A_5 = 100, A_6 = =101, A_7 = 110, A_8 = 111. Побудувати код з перевіркою на парність.

Відповідь: 0000, 0011, 0101, 0110, 1001, 1010, 1100, 1111.

4. Побудувати код Гемінга з $d_{min}=3$ для такої послідовності: 1101. Відповідь: 1010101.
5. Побудувати код Гемінга з $d_{min}=4$ для такої послідовності: 1101. Відповідь: 10101010.
6. Прийнята послідовність 1110001101 подає код Гемінга з $d_{min}=3$. Визначити інформаційні символи. Відповідь: 110101.
7. Визначити кількість помилок, що виправляє код з повторенням типу (3,1). Відповідь: одну помилку.
8. Використовуючи табл. 3.2, знайти правила побудови коду (10,3), який виправляє всі одиничні та подвійні помилки (див. прикл. 3.3).
9. Використовуючи табл. 3.2, побудувати код (10,3) для такої послідовності: 101. Відповідь: 0101110011.
10. З використанням циклічного несистематичного коду, що задається породжувальним поліномом $g(x)=1+x+x^3$, закодувати послідовність m = (0111). Відповідь: 0100011.
11. Визначити синдроми для всіх векторів помилок циклічного несистематичного коду з вправи 10. Відповідь: для 1 – 7 розрядів S=4, 2 1, 6, 3, 7, 5.
12. З використанням циклічного систематичного коду (див. прикл. 3.4), що задається породжувальним поліномом $g(x)=1+x+x^3$, закодувати послідовність m=(1101). Відповідь: 0001101.
13. Закодувати інформаційну послідовність k=(1001100…) з використанням згортного кодера з імпульсною характеристикою g=(11 10 11 00 00 …). Відповідь: 1110111101011100...
14. Закодувати інформаційну послідовність k=(1001100…) з використанням згортного кодера G=(7,5). Відповідь: 1110111101011100...
15. З використанням ґратчастої діаграми на рис. 3.9 декодувати методом Вітербі такий згортний код: 1010111101011100... Відповідь: 1001100…
16. Розробити програму завадостійкого кодування–декодування файлів з використанням коду Гемінга (7,4). Мова програмування С++ .
17. Розробити програму завадостійкого кодування–декодування файлів з використанням згортного коду G=(7,5). Мова програмування С++ .

4 ОСНОВИ КРИПТОЛОГІЇ

4.1 Теоретичні положення

Основні означення і терміни. Криптологія (kryptos – таємний, logos – слово) – наука про захист інформації шляхом її перетворення. Криптологія поділяється на криптографію і криптоаналіз [31 – 33].

Криптографія – наука про способи двонаправленого перетворення інформації з метою конфіденційної передачі її по незахищеному каналу зв'язку. Криптографія модифікує дані так, щоб без знання ключа шифру найсучасніші комп'ютери за прийнятний період часу не змогли відновити початковий текст.

Криптоаналіз, навпаки, досліджує можливості розшифрування інформації (відновлення початкового тексту) без знання ключа шифру.

Наукова криптологія ґрунтується на роботі К. Шеннона «Теорія зв'язку в секретних системах» (1949 р.), де показано, що кількість знаків шифротексту, які потрібні криптоаналітику для відновлення ключа шифру (розкриття шифру), становить:

$$n = \frac{H(Z)}{r \log N}, \tag{4.1}$$

де H(Z) – ентропія ключа;

r – надмірність відкритого тексту;

N – обсяг алфавіту.

Вираз (4.1) показує, що зниження надмірності (ущільнення даних) збільшує криптостійкість навіть для коротких ключів [1].

У загальному випадку алгоритми, що забезпечують конфіденційну передачу, діляться на тайнопис (стеганографія – steganos – таємниця; graphy – запис) [34] і криптографію [31 – 33, 35 – 40]. Тайнопис (стеганографія) відрізняється тим, що відправник і одержувач виконують над повідомленням перетворення, відомі тільки їм. Багато фахівців не відносить тайнопис до криптографії.

Сучасна криптографія містить чотири великі розділи:

- симетричні криптосистеми;
- криптосистеми з відкритим ключем;
- системи електронного підпису;
- управління ключами.

Інформація, що підлягає шифруванню і дешифруванню, розглядається як текст, побудований на деякому *алфавіті.*

Алфавіт – скінченна множина знаків, що використовується для кодування інформації.

Текст – впорядкований набір з елементів алфавіту.

Шифрування *(encryption)* – процес перетворення: початковий текст, або відкритий текст, замінюється шифрованим текстом.

Дешифрування *(decryption)* – на основі ключа шифрований текст перетворюється на початковий.

Ключ *(key)* – інформація, необхідна для шифрування і дешифрування текстів.

Криптографічна система – сімейством **T** перетворень відкритого тексту. Члени цього сімейства індексуються або позначаються символом *k*; параметр *k* є ключем. Простір ключів *K* – це набір можливих значень ключа. Ключ є послідовним рядом букв алфавіту.

Терміни «розподіл ключів» і «управління ключами» – змістом цих термінів є складання і розподіл ключів між користувачами.

Електронний (цифровий) підпис – це приєднуване до тексту його криптографічне перетворення, яке дозволяє перевірити авторство і достовірність повідомлення.

Криптостійкість – характеристика шифру, яка визначає його стійкість до дешифрування без знання ключа (тобто до криптоаналізу). Деякі показники криптостійкості такі:

- кількість всіх можливих ключів;

- середній час, необхідний для криптоаналізу.

Ефективність шифрування залежить від збереження таємниці ключа і криптостійкості шифру.

Класифікація алгоритмів шифрування. Розрізняють:

- симетричне шифрування (із секретним ключем)
- несиметричне шифрування (із відкритим ключем).

Симетричне шифрування – шифрування і дешифрування виконується одним і тим самим ключем (рис. 4.1). Ключ зберігається в таємниці і передається способом, що виключає перехоплення. Перевагою симетричного шифрування є висока швидкість шифрування, недоліком – ключ шифрування може бути перехоплений.

Несиметричне шифрування надійніше, але і складніше (рис. 4.2). Під час асиметричного шифрування процес шифрування виконується одним ключем (відкритим), а дешифрування – іншим (секретним), який відомий тільки отримувачу повідомлення.

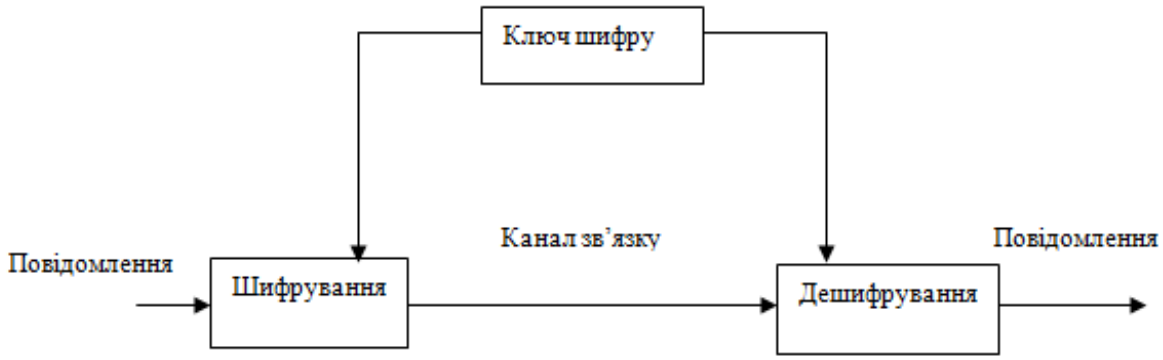

Рисунок 4.1 – Симетричне шифрування

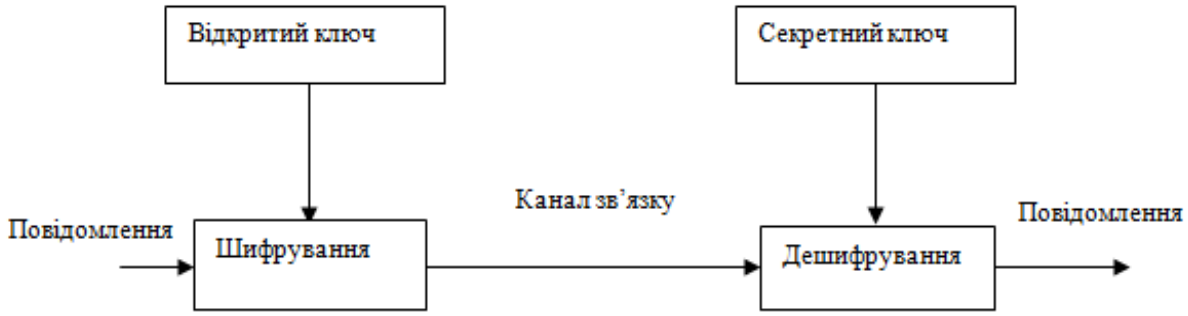

Рисунок 4.2 – Несиметричне шифрування

Якщо крипостійкість забезпечується збереженням секретності алгоритму шифрування, то такі алгоритми називають обмеженими. Малопридатні для застосування через велику вартість.

Симетричні алгоритми шифрування поділяють на:

- потокові алгоритми шифрування
- блокові алгоритми шифрування.

В потокових шифрах кожний біт відкритого тексту перетворюється незалежно від інших.

В блокових шифрах перетворення виконуються над блоками фіксованої довжини. Відомі такі блокові шифри:

- шифри перестановки
- шифри заміни (підстановки).

Шифри перестановки – це елементи відкритих даних у деякому новому порядку. Відомі шифри вертикальної, горизонтальної та інших перестановок.

Шифри підстановки замінюють за певним правилом елементи відкритих даних на інші елементи. Шифри підстановки поділяються на:

- моноалфавітні (код Цезаря)
- поліалфавітні (шифр Віженера та ін.)

Шифри, що використовують і підстановки, і перестановки, називаються складеними. Вони більш стійкі порівняно з шифрами, які використовують один спосіб. Саме складені шифри використовують стандарти шифрування

Найпоширеніші стандарти шифрування такі:

- симетричні – DES (Data Encryption Standard), AES (Advanced Encryption Standard), ГОСТ 28147-89 та ін.;
- несиметричні – RSA (Rivest, Shamir, Alderman), PGP (Pretty Good Privacy) [31].

Симетричні системи шифрування. Шифри Віженера, Цезаря, Вернама. Шифр Віженера реалізовує багатоалфавітну підстановку. У шифрі Віженера кожна буква алфавіту нумерується. Ключ – це послідовність букв, які з повтореннями підписують під повідомленням.

Зашифроване значення (криптограма) є результатом додавання з приведенням за модулем розміру алфавіту (26 для англійського алфавіту, 33 для українського) цифрових еквівалентів букви повідомлення і букви ключа, що лежить під нею (приклад 4.1).

Шифр Цезаря – це шифр Віженера з однобуквенним ключем, а шифр Вернама – з необмеженим ключем, що не повторюється [2, 3].

Гамування. Межа між гамуванням і шифрами Віженера доволі умовна. Під час *шифрування* гамуванням за ключем генерується гама шифру з використанням генератора псевдовипадкових чисел, яка накладається на відкриті дані, наприклад, використовуючи додавання за модулем 2, тобто так, щоб можна було відновити початкові дані. Під час *дешифрування* повторно генерується та ж сама гама шифру і накладається на зашифровані дані.

Якщо період гами шифру перевищує довжину зашифрованого тексту і ніякий фрагмент відкритого тексту невідомо, то шифр розкривається тільки прямим перебором (пробій на ключ) і криптостійкість визначається тільки розміром ключа. Однак, якщо зловмисникові стає відомо фрагмент відкритого тексту і відповідна йому шифрограма, то шифр легко розкривається.

Для формування гами шифру найчастіше використовуються лінійні конгруентні генератори ПВЧ – послідовність псевдовипадкових чисел T_1, T_2,…, T_m , що формується згідно з виразом:

$$T_{i+1} = (aT_i + c) \bmod m, \qquad (4.2)$$

де a і c – константи.

T_0 – породжувальне число генератора.

Значення *m* вибирається таким, що дорівнює 2^n, n – довжина машинного слова в бітах. Для забезпечення максимальної довжини

(періоду) m вихідної послідовності генератора потрібно, щоб c було непарним і $a \bmod 4 = 1$ [2].

Стандарт шифрування DES. Стандарт DES (рис. 4.3) є основою міжнародного стандарта ISO 8372-87[35-36, 39-40]. У стандарті DES використовується мережа Фейстеля, яка забезпечує багатократне використання ключа. Дані розділяються на блоки довжиною 64 біти, ключ шифру також 64-бітовий. До кожного блока застосовується початкова перестановка (IP), далі – складна обчислювальна процедура, залежна від ключа, в кінці – обернена перестановка (IP^{-1}). Початкова перестановка IP визначається так:

58 50 42 34 26 18 10 02
60 52 44 36 28 20 12 04
62 54 46 38 30 22 14 06
64 56 48 40 32 24 16 08

57 49 41 33 25 17 09 01
59 51 43 35 27 19 11 03
61 53 45 37 29 21 13 05
63 55 47 39 31 23 15 07

Де елемент a(i,j) – номер біта вхідної 64-бітової послідовності.
Обернена перестановка IP^{-1} така:

40 08 48 16 56 24 64 32
39 07 47 15 55 23 63 31
38 06 46 14 54 22 62 30
37 05 45 13 53 21 61 29
36 04 44 12 52 20 60 28
35 03 43 11 51 19 59 27
34 02 42 10 50 18 58 26
33 01 41 09 49 17 57 25

L, R – 32-бітові послідовності (рис. 4.3);
K1...K16 – 48-бітові послідовності, які вибираються з 64-бітового ключа;
«+» – додавання за модулем 2.

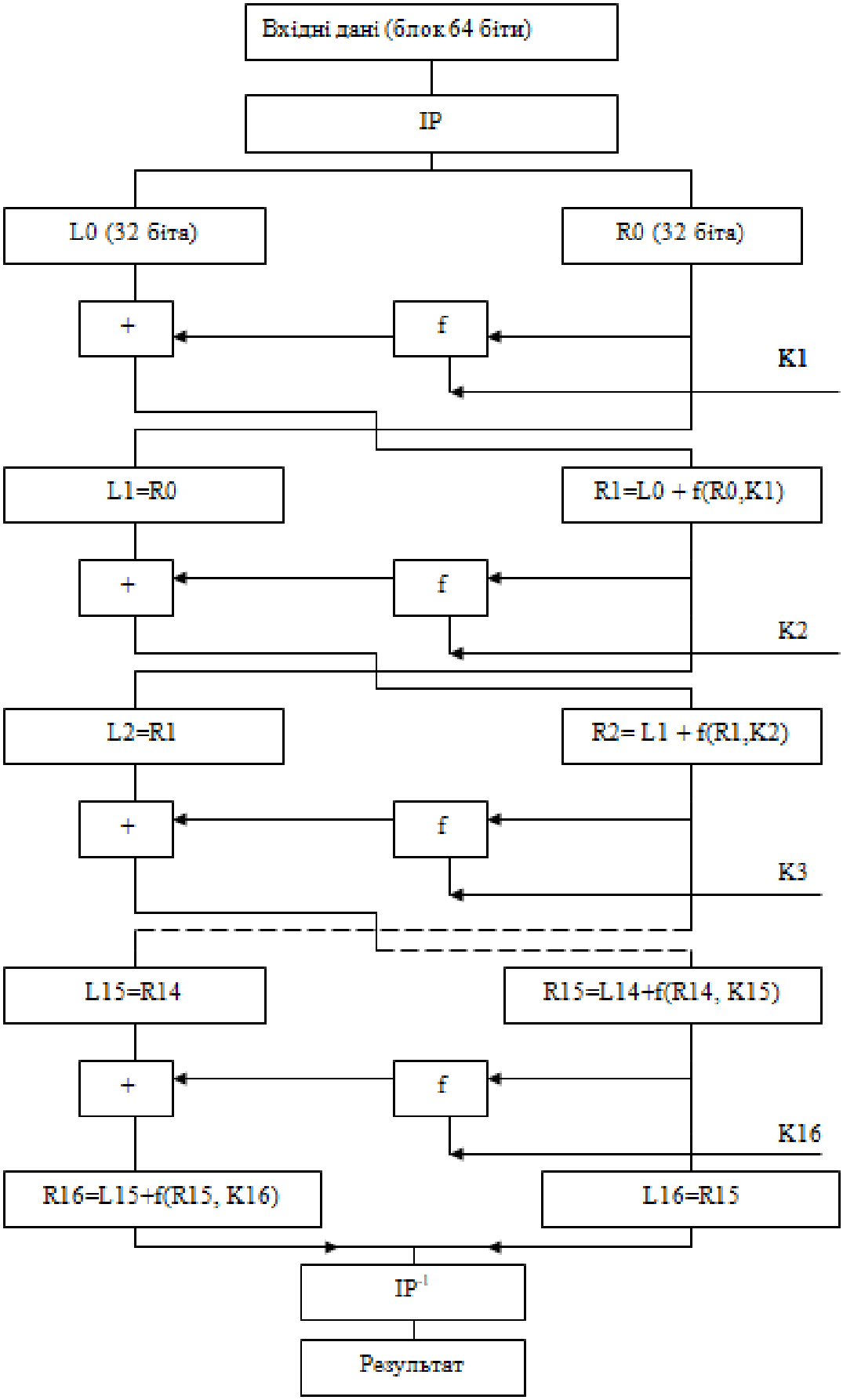

Рисунок 4.3 – Шифрування інформації згідно зі стандартом DES

Процес шифрування описується так:

$$\left.\begin{aligned} L(N) &= R(N-1) \\ R(N) &= L(N-1) + f(R(N-1), K(N)) \\ K(N) &= KS(N, KEY) \end{aligned}\right\}, \qquad (4.3)$$

де KS – функція вибору ключів;
f(R,K) – функція шифрування.

З (4.3) випливає:

$$\left.\begin{aligned} R(N-1) &= L(N) \\ L(N-1) &= R(N) + f(R(N-1), K(N)) \end{aligned}\right\}. \qquad (4.4)$$

Тобто процес дешифрування симетричний процесу шифрування, тільки ключі вибираються у зворотному порядку.

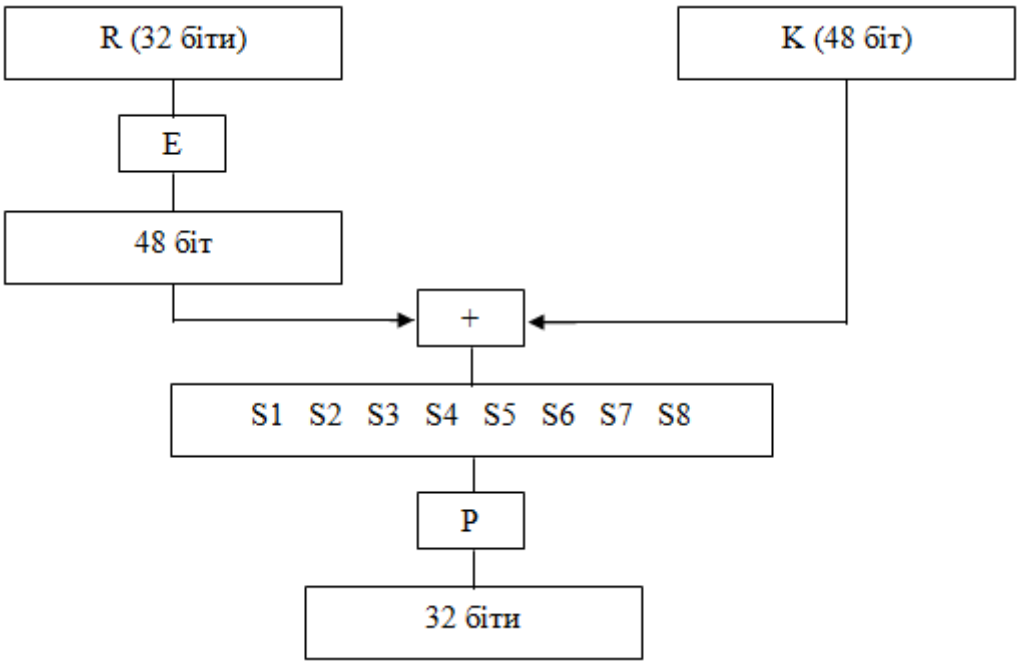

Рисунок 4.4 – Обчислення функції шифрування f(R,K)

Загальну схему обчислення f(R,K) наведено на рис. 4.4. «E» позначає функцію, для якої вхідним є блок із 32 біт, а вихідним – блок із 48 біт, які отримують шляхом вибору бітів із вхідних даних блока відповідно до такої таблиці:

Таблиця вибору бітів для «Е»

32 01 02 03 04 05
04 05 06 07 08 09
08 09 10 11 12 13
12 13 14 15 16 17
16 17 18 19 20 21
20 21 22 23 24 25
24 25 26 27 28 29
28 29 30 31 32 01

Функція «Р» – перестановка, що породжує 32-бітовий результат із 32-бітових вхідних послідовностей:

Перестановка «Р»

16 07 20 21
29 12 28 17
01 15 23 26
05 18 31 10
02 08 24 14
32 27 03 09
19 13 30 06
22 11 04 25

Для функцій S1...S8 вхідні дані – блоки по 6 біт, у яких перший і останній біти – номер рядка, а 4 середніх біти – номер стовпця. Наприклад, для S1:

011011	– вхідний блок;
01(1)	– номер рядка;
1101(13)	– номер стовпця;
0101(5)	– вихідні дані.

Нумерація стовпців і рядків починається з «0». Примітивні функції S1, ... , S8 такі:

S1:
14 04 13 01 02 15 11 08 03 10 06 12 05 09 00 07
00 15 07 04 14 02 13 01 10 06 12 11 09 05 03 08
04 01 14 08 13 06 02 11 15 12 09 07 03 10 05 00
15 12 08 02 04 09 01 07 05 11 03 14 10 00 06 13
S2:
15 01 08 14 06 11 03 04 09 07 02 13 12 00 05 10
03 13 04 07 15 02 08 14 12 00 01 10 06 09 11 05
00 14 07 11 10 04 13 01 05 08 12 06 09 03 02 15
13 08 10 01 03 15 04 02 11 06 07 12 00 05 14 09
S3:
10 00 09 14 06 03 15 05 01 13 12 07 11 04 02 08
13 07 00 09 03 04 06 10 02 08 05 14 12 11 15 01
13 06 04 09 08 15 03 00 11 01 02 12 05 10 14 07
01 10 13 00 06 09 08 07 04 15 14 03 11 05 02 12
S4:
07 13 14 03 00 06 09 10 01 02 08 05 11 12 04 15
13 08 11 05 06 15 00 03 04 07 02 12 01 10 14 09
10 06 09 00 12 11 07 13 15 01 03 14 05 02 08 04
03 15 00 06 10 01 13 08 09 04 05 11 12 07 02 14
S5:
02 12 04 01 07 10 11 06 08 05 03 15 13 00 14 09
14 11 02 12 04 07 13 01 05 00 15 10 03 09 08 06
04 02 01 11 10 13 07 08 15 09 12 05 06 03 00 14
11 08 12 07 01 14 02 13 06 15 00 09 10 04 05 03
S6:
12 01 10 15 09 02 06 08 00 13 03 04 14 07 05 11
10 15 04 02 07 12 09 05 06 01 13 14 00 11 03 08
09 14 15 05 02 08 12 03 07 00 04 10 01 13 11 06
04 03 02 12 09 05 15 10 11 14 01 07 06 00 08 13
S7:
04 11 02 14 15 00 08 13 03 12 09 07 05 10 06 01
13 00 11 07 04 09 01 10 14 03 05 12 02 15 08 06
01 04 11 13 12 03 07 14 10 15 06 08 00 05 09 02
06 11 13 08 01 04 10 07 09 05 00 15 14 02 03 12
S8:
13 02 08 04 06 15 11 01 10 09 03 14 05 00 12 07
01 15 13 08 10 03 07 04 12 05 06 11 00 14 09 02
07 11 04 01 09 12 14 02 00 06 10 13 15 03 05 08
02 01 14 07 04 10 08 13 15 12 09 00 03 05 06 11

Функція вибору ключів(KS). Схему вибору 48-бітових блоків K1, ..., K16 з 64-бітового ключа наведено на рис. 4.5, а способи отримання блоків C, D та ключів – на рис. 4.6.

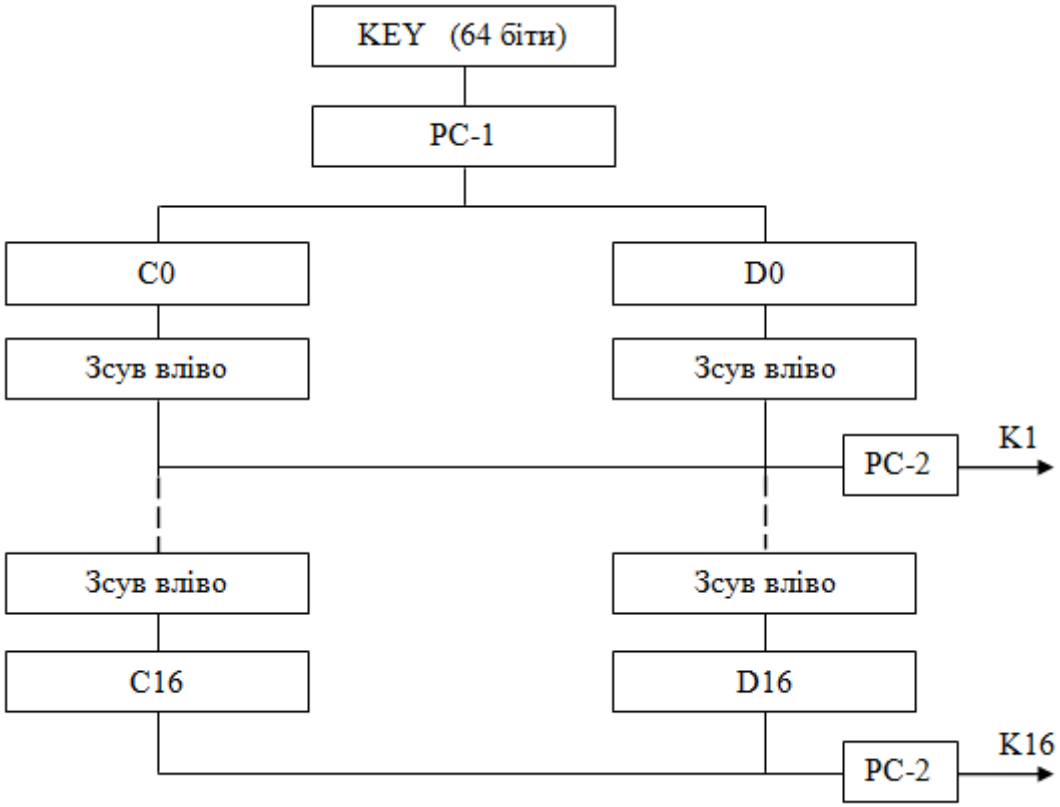

Рисунок 4.5 – Вибір ключів

Системи з відкритим ключем. Основною проблемою симетричної криптографічної системи є проблема розподілу ключів [35 – 36]. Вирішенням цієї проблеми є системи з відкритим ключем, у яких відкритий текст шифрується відкритим ключем адресата, а дешифрування повідомлення можливе тільки з використанням закритого (секретного) ключа, який відомий тільки самому адресатові.

Основою криптографічних систем з відкритим ключем є необоротні або односторонні функції, для яких при заданому значенні x просто обчислити значення f(x), але якщо відомо y=f(x), то обчислення значення x неможливе або вимагає неприйнятної обчислювальної складності.

Відомо безліч класів необоротних функцій, що породжує різноманітність систем з відкритим ключем, але не всі необоротні функції придатні для побудови несиметричних шифрів, оскільки під необоротністю розуміють не теоретичну необоротність, а практичну неможливість обчислення зворотного значення на сучасному етапі розвитку обчислювальних засобів за досяжний інтервал часу.

Номер ітерації	Число зсувів вліво
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Перестановка PC-1.

C0:
57 49 41 33 25 17 09
01 58 50 42 34 26 18
10 02 59 51 43 35 27
19 11 03 60 52 44 36

D0:

63 55 47 39 31 23 15
07 62 54 46 38 30 22
14 06 61 53 45 37 29
21 13 05 28 20 12 04

Примітка. Біти 8,16, ... , 64 використовуються для контролю за непарністю в кожному байті.Зсув вліво циклічний.

Перестановка PC-2

14 17 11 24 01 05
03 28 15 06 21 10
23 19 12 04 26 08
16 07 27 20 13 02
41 52 31 37 47 55
30 40 51 45 33 48
44 49 39 56 34 53
46 42 50 36 29 32

Рисунок 4.6 – Способи отримання блоків C і D та ключів

Щоб гарантувати надійний захист інформації системи з відкритим ключем (СВК) мають відповідати таким вимогам:

1. Перетворення має виключати можливість дешифрування на основі відкритого ключа.

2. Визначення закритого ключа за відомим відкритим ключем має бути неможливим на сучасному технічному рівні. Бажана точна нижня оцінка кількості операцій для розкриття шифру.

Загалом, криптосистеми з відкритим ключем використовують такі необоротні перетворення [35 – 36]:

- розкладання великих чисел на прості множники;
- обчислення логарифма в скінченному полі;
- обчислення коренів алгебраїчних рівнянь.

Криптосистеми з відкритим ключем (СВК) використовують в таких напрямках:

1. Як самостійні засоби криптографічного захисту.
2. Як засоби для розподілу ключів для симетричних систем шифрування.
3. Як засоби автентифікації користувачів (електронний підпис).

Алгоритм шифрування RSA. Світовим стандартом де-факто для відкритих систем є алгоритм RSA (Рівеста (Rivest R.), Шаміра (Shamir A.) і Едлемана (Adleman L.) – автори алгоритму) [35 – 36]. RSA грунтується на вірі, що модульне піднесення до степеня за фіксованої експоненти і модуля є однонаправленою (односторонньою) функцією з потайним ходом. Для розуміння роботи RSA бажано ознайомитись з елементами теорії чисел [2, 36].

Першим етапом роботи асиметричного алгоритму є створення відкритого і закритого ключів. Для RSA створення ключів містить такі операції:

1. Вибираються два великих простих числа p і q.
2. Знаходиться добуток n=p·q (модуль шифру).
3. Вибирається ціле число e взаємно просте з φ (p,q)=(p-1) · (q-1), що не дорівнює p або q і 0<e<(p-1) · (q-1). На практиці e беруть – 3, 17 або 65537 [32].
4. Обчислюють ціле число d>0, яке відповідає такій умові:

$$(e \cdot d) \bmod [(p-1) \cdot (q-1)] = 1 \text{ або } e \cdot d + (p-1) \cdot (q-1) \cdot y = 1.$$

   Невідомі змінні d і y знаходять або методом перебору, або з використанням узагальненого алгоритму Евкліда. Якщо результат обчислення за Евклідом d<0, то у цьому випадку:

$$d = d + (p-1) \cdot (q-1).$$

5. Два числа (e, n) — публікуються як відкритий ключ.
6. Число d – закритий (секретний) ключ, який дозволяє дешифрувати повідомлення, зашифровані за допомогою відкритого ключа (e, n).

Шифрування виконується так.

1. Повідомлення розбивається на блоки довжиною $k=[\log_2 n]$ біт (ціла частина дробового числа). Це число m_i в діапазоні від 0 до 2^{k-1} і $m_i<n$.
2. Кожне таке число шифрується з використанням такого виразу:

$$c_i=(m_i)^e \bmod n, \quad (4.5)$$

де c_i – зашифроване повідомлення.

Для розшифрування цього повідомлення необхідний секретний ключ:

$$m_i=(c_i)^d \bmod n. \quad (4.6)$$

Криптоалгоритм RSA є частиною багатьох стандартів шифрування, Зокрема, ISO 9796, S/MINE та інші.

Недолік RSA: якщо будуть знайдені ефективні методи розкладу n на співмножники p і q, то секретний ключ d буде отримано за прийнятний час, тому рекомендовано такі розміри модуля n:

- 768 біт – для приватних осіб;
- 1024 біт – для комерційної інформації;
- 2048 біт – для особливо секретної інформації [36].

Шифрування даних під час ущільнення. Зменшення обчислювальних витрат можна досягти за рахунок суміщення операцій ущільнення і шифрування даних [15], тобто, за один прохід виконувати ущільнення інформації з її одночасним шифруванням.

Серед алгоритмів ущільнення даних мінімум затрат на адаптацію до розв'язування задач шифрування мають два алгоритми (табл. 4.1):

- ущільнення методом MTF (Move To Front);
- імовірнісний метод ущільнення.

Ці алгоритми є адаптивними, тобто не потребують передачі додаткової інформації, яка могла бути використана зловмисниками для злому шифру, передбачають перед виконанням ущільнення формування таблиці символів, яку можна сформувати за ключем шифру, характеризуються доволі простою технічною реалізацією.

Таблиця 4.1 – Характеристики методів ущільнення

Метод ущільнення	Коефіцієнт ущільнення	Обчислювальні затрати	Адаптивний	Додаткові обчислювальні затрати для шифрування
Словниковий	Близький до оптимального для великих масивів	Середні	Так	Так
Гаффмана	Оптимальний	Середні	Ні	Так
MTF	Близький до оптимального	Середні	Так	Ні
Імовірнісний	Близький до оптимального	Малі	Так	Ні
Арифметичний	Найбільший	Великі	Так	Так

Як приклад розглянемо шифрування і ущільнення методом MTF (кодування за ступенем новизни). Нехай передається повідомлення:

ω**=ABCDDAAACAB**

в алфавіті **AL={A,B,C,D}**.

Результати кодування за ступенем новизни демонструє табл. 4.2, Вихід 1.

Якщо порядок символів в списку алфавіту «AL» змінити, наприклад згенерувати його, використовуючи генератор ПВЧ, то вихідна послідовність стане іншою (табл. 4.2, Вихід 2, Алфавіт 2). Без знання початкового порядку символів алфавіту неможливо дешифрувати повідомлення, оскільки тут реалізовано шифрування методом багатоалфавітної підстановки і ущільнення за один прохід.

Таблиця 4.2 – Кодування за ступенем новизни

Вхід	Вихід 1	Алфавіт 1	Вихід 2	Алфавіт 2
A	0_2	{ABCD}	110_2	{CBAD}
B	10_2	{ABCD}	110_2	{ACBD}
C	110_2	{BACD}	110_2	{BACD}
D	111_2	{CBAD}	111_2	{CBAD}
D	0_2	{DCBA}	0_2	{DCBA}
A	111_2	{DCBA}	111_2	{DCBA}
A	0_2	{ADCB}	0_2	{ADCB}

Схема одночасного ущільнення і шифрування така:

1. Генерується алфавіт повідомлення за ключем шифру з використанням генератора ПВЧ.
2. Виконується ущільнення згідно з наведеним методом.

Додаткові затрати відсутні, оскільки генерація символів алфавіту виконується у будь-якому випадку.

Комп’ютерна стеганографія. Принцип стеганографії – один масив інформації приховується в іншому. Комп'ютерні файли (зображення, звукові записи і т. п.) мають області, що не використовуються або не є важливими (наприклад, молодші розряди в цифровому поданні значення піксела зображення або відліку звуку), тому існує можливість розмістити тут спеціальну інформацію. Створено цілий клас методів і програмних продуктів для розміщення прихованої інформації у цифрові документи майже всіх типів [34].

Стеганографія відома протягом тисячоліть [34], однак комп'ютерна стеганографія – молодий напрямок, що розвивається. Наведемо деякі означення.

Стеганографічна система (стегосистема) – сукупність засобів та методів для формування прихованого каналу передавання інформації.

Контейнер – інформація, у якій приховуються таємні повідомлення. Порожній контейнер – контейнер без вбудованого повідомлення; наповнений контейнер (стеганоконтейнер), контейнер з вбудованою інформацією.

Вбудоване (приховане) повідомлення – повідомлення, що вбудовується в контейнер.

Стеганографічний канал – канал передавання прихованого повідомлення.

Стегоключ – секретний ключ для приховування інформації. У стегосистемі може бути один або кілька стегоключів.

Стеганографічні системи вирішують такі основні задачі:

- захист конфіденційної інформації від несанкціонованого доступу;
- захисту авторських прав на деякі види інтелектуальної власності;
- подолання систем моніторингу і керування мережними ресурсами;
- камуфляжу програмного забезпечення;
- створення прихованих від законного користувача каналів витоку важливої інформації.

Основи комп’ютерної стеганографії розглядаються в [34].

4.2 Приклади розв’язування задач

Приклад 4.1. Зашифрувати слово CAREFULLY кодом Віженера з ключем PIES.

Розв’язання. Запишемо букви повідомлення, розташувавши під ними їх цифрові еквіваленти.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
0	1	2	3	4	5	6	7	8	9	10	11	12	13
O	P	Q	R	S	T	U	V	W	X	Y	Z		
14	15	16	17	18	19	20	21	22	23	24	25		

Аналогічно внизу запишемо ключ.

C	A	R	E	F	U	L	L	Y
2	0	17	4	5	20	11	11	24
P	I	E	S	P	I	E	S	P
15	8	4	18	15	8	4	18	15

Додаючи верхні і нижні цифрові еквіваленти з приведенням за модулем 26, одержимо:

17 8 21 22 20 2 15 3 13.

Що відповідає криптограмі RIVWUCPDN.

Приклад 4.2. Дешифруйте криптограму [37]:

ВДОЩТФЩЯИТФГЯІИС,

яка отримана внаслідок застосування лінійного афінного шифру до тексту українською (розділові знаки і пропуски між словами вилучено), за умови, що біграмі ТФ у криптограмі відповідає біграма ЕР відкритого тексту. Запишіть рівняння зашифрування та розшифрування.

Розв'язання. Рівняння зашифрування шукатимемо у вигляді y=(kx + t) mod33 (33 – кількість літер українського алфавіту). Запишемо цифровий еквівалент криптограми (тобто номери літер в алфавіті):

2 5 18 29 22 24 29 32 10 22 24 3 32 11 10 21

та букв у заданих біграмах Е = 6; Р = 20; Т = 22; Ф = 24.

Тоді

$$22 = (6k + t) \bmod 33,$$

$$24 = (20k + t) \bmod 33.$$

Віднявши рівняння почленно і розділивши на 2 (оскільки 2 і 33 взаємо прості числа), приходимо до порівняння

$$1 = 7k \bmod 33.$$

Звідки k=19. Тоді з першого рівняння:

$$22 = (6 * 19 + t) \bmod 33;$$

$$22 = (114 + t) \bmod 33;$$

$$114 \bmod 33 = 15$$

$$t = 22 - 15 = 7$$

Отже рівняння шифрування таке:

$$y=(19x + 7) \bmod 33.$$

Знайдемо рівняння розшифрування:

$$y - 7 = 19x \bmod 33;$$

$$x = (19^{-1}(y - 7)) \bmod 33;$$

$$(19 * m) \bmod 33 = 1$$

$$19^{-1} = m = 7$$

$$x = 7(y - 7) \bmod 33 = (7y + 17) \bmod 33$$

Використавши отримане рівняння, розшифровуємо криптограму:

А	Б	В	Г	Ґ	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32

ВДОЩТФЩЯИТФГЯІИС
В
x=(7*2+17) mod 33=31 Ю
Д
x=(7*5+17) mod 33=19 П
О
x=(7*18+17) mod 33=11 І
Щ
x=(7*29+17) mod 33=22 Т
…
С
x=(7*21+17) mod 33=32 Я
ЮПІТЕР, ТИ СЕРДИШСЯ.

Приклад 4.3. Уявіть, що Ви знайшли старий зашифрований текст, за припущенням, зашифрований з використанням шифру Віженера, а у відкритому повідомленні йшлося про різні стародавні шифри. Ви виявляєте, що рядок РВМФПГІЦЛЬМЩ присутній у шифротексті двічі: вперше він починається з позиції 10-го символу тексту, а вдруге – з позиції 241-го (відлік символів тексту з одиниці, алфавіт український). Ви припускаєте, що цей фрагмент шифротексту відповідатиме у відкритому тексті слову КРИПТОГРАФІЯ. Якщо це припущення правильне, то яким був ключ шифрування?

Розв'язання. Оцінимо період гами на основі тесту Казіскі [37]. Відстань між двома появами фрагмента 241 – 10 = 231 = 3×7×11 знаків. Тому можливі значення періоду – 3, 7 або 11.

Знайдемо різницю номерів символів шифрованого тексту і слова «криптографія»:

РВМФПГІЦЛЬМЩ	КРИПТОГРАФІЯ	Різниця	Символ
20	14	6	Е
2	20	15	Л
16	10	6	Е
24	19	5	Д
19	22	30	Ь
3	18	18	О
11	3	8	Ж
26	20	6	Е
15	0	15	Л
30	24	6	Е
16	11	5	Д
29	32	30	Ь

Таким чином виникає ЕЛЕДЬОЖЕЛЕДЬ. В такому разі період гами не може бути кратним ані 3, ані 11, бо в цих випадках словоподібні структури не з'являються за будь-якого зсуву. Отже, ключове слово ОЖЕЛЕДЬ має довжину 7 та починається з 15 позиції.

Приклад 4.4. За допомогою трьох класичних шифрів відкритий текст

СВІТ ЛОВИВ МЕНЕ, АЛЕ НЕ ПІЙМАВ

було зашифровано та отримано три шифротексти:

1) ЬРРЇЬНІШІЕСМЛОФЩЯДШЩТЕЛЬ;
2) ХЕЛЦПТЕКЕРИСИҐПИСИУЛНРҐЕ;
3) ІОЛСТВИВЕАЕВНМЕЛПМЙНІЕВА.

Визначити, які шифри використано, та встановити ключі шифрування. Яку атаку Вам потрібно провести, якщо перед шифруванням у відомому висловлюванні Г. Сковороди вилучили кому та пропуски між словами.

Розв'язання. Аналіз пари «відкритий текст – відповідний шифрований текст» виявляє:

1. У першому шифрі шифропозначення букв залежить від їх положень у відкритому тексті. Так, буква Е переходить у букви С, Л, Щ або Д. Тому моноалфавітна підстановка не використовувалась. Не було задіяно й шифр перестановки, бо у шифрограмі є букви, відсутні у відкритому тексті. Перевіряємо, а чи не був це шифр Віженера. Якщо від букв шифротексту відняти букви відкритого тексту, то отримаємо ЗОЗУЛЯЗОЗУЛЯЗОЗУЛЯЗОЗУЛЯ. Отже, це був шифр Віженера із ключем ЗОЗУЛЯ.

2. У другій шифрограмі присутні букви, яких не було у відкритому тексті, але шифропозначення букв не дублюються. Все це свідчить на користь того, що найімовірніше – це проста моноалфавітна підстановка. Зібравши дані у таблицю, приходимо до таких замін:

А	Б	В	Г	Ґ	Д	Е	Є	Ж	З	И	І	Ї	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ь	Ю	Я
↓		↓			↓					↓	↓			↓	↓		↓		↓	↓	↓	↓	↓									
Ґ		Е			И					К	Л			Н	П		Р		С	Т	У			Х	Ц							

Як бачимо, це може бути шифр зсуву з ключем 4.

3. Третя криптограма отримана шифруванням за допомогою шифру перестановки, бо перші її вісім букв – результат перестановки перших восьми букв відкритого тексту. Тут і далі для 8-буквених блоків відкритого тексту застосовано перестановку (3 6 5 1 4 2 8 7).

У всіх випадках проведено атаку на основі пари «відкритий текст – шифрований текст».

Приклад 4.5. Зашифрувати з використанням конгруентного генератора ПВЧ таку послідовність байтів: 65, 66, 67.

Розв'язання. Виберемо параметри генератора ПВЧ: a=21, c=13, m=256. За таких параметрів генератор буде мати максимальну довжину 256, що забезпечує зміну всіх розрядів байтів повідомлення. Як T_0 введемо ключ шифру, нехай $T_0 = 10$. Тоді порядок шифрування такий.

1. Обчислюємо T_1 згідно з виразом (4.2):

$$T_1 = (aT_0 + c) \bmod m = (21*10+13) \bmod 256 = 223.$$

Зашифруємо перший байт повідомлення, взявши суму за модулем 2 чисел 65 і 223, поданих у двійковій формі:

$$\begin{array}{r} 11011111 \\ \underline{01000001} \\ 10011110 \end{array}.$$

В десятковій формі: 158.

2. Обчислюємо T_2 згідно з виразом (4.2):

$$T_2 = (aT_1 + c) \bmod m = (21*223+13) \bmod 256 = 88.$$

Зашифруємо другий байт повідомлення, взявши суму за модулем 2 чисел 66 і 88, поданих у двійковій формі:

$$\begin{array}{r} 01011000 \\ \underline{01000010} \\ 00011010 \end{array}.$$

В десятковій формі: 26.

3. Обчислюємо T_3 згідно з виразом (4.2):

$$T_3 = (aT_2 + c) \bmod m = (21*88+13) \bmod 256 = 69.$$

Зашифруємо третій байт повідомлення, взявши аналогічно суму за модулем 2 чисел 67 і 69. Отримаємо 6. Таким чином зашифроване повідомлення таке: 158, 26, 6.

Приклад 4.6. Обчислити інверсією числа c=7 за модулем m=11, тобто обчислити $7^{-1} \bmod 11$.

Розв'язання. Позначимо інверсію d. Інверсія d має задовольняти умову:

$$7 \cdot d \bmod 11 = 1, \text{ або } 11 \cdot y + 7 \cdot d = 1.$$

Для розв'язання цього рівняння використаємо узагальнений алгоритм Евкліда [1]:

11	1	0	
7	0	1	
4	1	-1	q=1
3	-1	2	q=1
1	2	-3	q=1
0	-7	11	q=3.

Звідки d = -3 і значення інверсії таке: d=d mod 11 = 11-3 = 8, тобто $7^{-1}\cdot \bmod 11 = 8$. Перевіримо результат: $7\cdot 8 \bmod 11 = 56 \bmod 11 = 1$.

Приклад 4.7. Піднести число 4 до степеня 7 за модулем 13.

Розв'язання. Використаємо алгоритм піднесення до степеня справа наліво [1]:

ВХІД: Цілі числа a, $x = (x_t x_{t-1} \ldots x_0)_2$, p.
ВИХІД: Число $y = a^x \bmod p$.
1. y←1, s←a.
2. FOR i = 0,l...,t DO
3. IF $x_i = 1$ THEN y←y·s mod p;
4. s ← s·s mod p.
5. RETURN y.

Оскільки x=6= $(110)_2$, a=4, p=13, тоді початкові значення y=1, s=4. Далі переглядаються біти числа x, починаючи з молодшого і виконуються обчислення згідно з алгоритмом:

i:	0	1	2
x_i:	0	1	1
y:	1	3	1
s:	3	9	3

Таким чином, $y=7^6 \bmod 13=1$

Приклад 4.8. Піднести число 3 до степеня 23 за модулем 29.

Розв'язання. Використаємо алгоритм піднесення до степеня зліва направо [1]:

ВХІД: Цілі числа a, $x = (x_t x_{t-1} \ldots x_0)_2$, p.
ВИХІД: Число $y = a^x \bmod p$.
1. y←1.
2. FOR i = t, t-1,...,0 DO
3. y ← y·y mod p;
4. IF $x_i = 1$ THEN y←y·a mod p;
5. RETURN y.

Оскільки x=23= $(10111)_2$, a=3, p=29, тоді початкові значення y=1. Далі переглядаються біти числа x, починаючи з старшого і виконуються обчислення згідно з алгоритмом:

i:	4	3	2	1	0
x_i:	1	0	1	1	1
y:	3	9	11	15	8

Таким чином, $y=3^{23} \bmod 29=8$

Приклад 4.9. Знайти відкритий і секретний ключ для RSA, якщо p=101; q=131.

Розв'язання.

1. Обчислюємо добуток n=p·q =101·131=13 231 (модуль алгоритма).

2. Вибираємо довільне число e, яке є взаємно простим з (p-1)·(q-1)=100·130=13 000, тобто gcd(e, 13000)=1: e=601.

3. Визначаємо таке число d, для якого істинним є співвідношення: (e·d) mod[(p-1)·(q-1)]=1 або (601·d)mod13000 =1. Невідомі змінні d і y знаходимо з використанням узагальненого алгоритму Евкліда:

13000*y + 601*d = 1			
13000	1	0	
601	0	1	q = 21
379	1	−21	q = 1
222	−1	22	q = 1
157	2	−43	q = 1
65	−3	65	q = 2
27	8	−173	q = 2
11	−19	411	q = 1
5	46	−995	q = 2
1	−111	2401	q = 2
0			q = 5

d=2401 mod 13000=2401.
Два числа (e, n)=(601,13231) – відкритий ключ.
Числа (d, n)= (2401, 13231) – закритий (секретний) ключ.

Приклад 4.10. Зашифрувати букви англійського алфавіту методом RSA.

Розв'язання. Букв в англійському алфавіті 26. Згенеруємо відкритий і секретний ключі.

1. Вибираємо p=3 і q=11.
2. Обчислюємо добуток n=p·q =3·11=33 (модуль алгоритма).

3. Вибираємо довільне число e, яке є взаємно простим з (p-1)·(q-1)=2·10=20, тобто gcd(e, 20)=1: e=7.
4. Визначаємо таке число d, для якого істинним є співвідношення: (e·d) mod[(p-1)·(q-1)]=1 або (7·d) mod20 =1. Невідомі змінні d і y знаходимо з використанням узагальненого алгоритму Евкліда: d=3.
5. Два числа (e, n)=(7,33) – відкритий ключ.
6. Числа (d, n)= (3, 33) – закритий (секретний) ключ.

Нехай цифрові еквіваленти букв такі:

A	B	C	D	…	Z
4	5	6	7		29.

Для їх подання потрібні блоки довжиною 5 біт, що відповідає вимозі на довжину блока $k=[\log_2 n]= [\log_2 33]=5$ біт.

Зашифруємо наприклад букву A:

$$c_i=(m_i)^e \bmod n=4^7 \bmod 33 = 16384 \bmod 33 = 16.$$

Для дешифрування використаємо такий вираз:

$$m_i=(c_i)^d \bmod n = 16^3 \bmod 33 = 4.$$

Відновлене значення відповідає цифровому еквіваленту букви A.

Приклад 4.11. Сформувати цифровий підпис для повідомлення m=(0100000100100010). Обчислення хеш-функції повідомлення виконати з використанням циклічного надлишкового коду (CRC) з твірним поліномом $g(x)=x^4+x+1$, а для формування цифрового підпису використати асиметричний алгоритм шифрування RSA з p=101; q=131.

Розв'язання. Порядок формування цифрового підпису на передавальній стороні такий.

1. Визначимо хеш-функцію повідомлення. У такому випадку це залишок від ділення полінома, що відповідає інформаційній послідовності, на твірний поліном циклічного коду. Використаємо простий алгоритм, у якому виконується звичайне ділення в стовпчик, але замість віднімання використовують операцію XOR. Оскільки твірному поліному відповідає двійкова послідовність g=10011 (коефіцієнти при степенях полінома), то отримаємо:

010000010100010
^10011
11010100010
^10011
1001100010
^10011
00010

Хеш-функція повідомлення така: $CRC=0010_2=2_{10}$.

2. Для асиметричного алгоритму RSA визначаємо відкритий і секретний ключі (дивись приклад 4.8), але за цифрового підпису відкритий ключ стає секретним, а секретний – відкритим. Тому:

- (e, n)=(601,13231) – секретний ключ;
- (d, n)= (2401, 13231) – відкритий ключ.

Автор шифрує хеш-функцію повідомлення з використанням секретного ключа, а дешифрування виконується з використанням відкритого ключа на сторонні приймача повідомлення.

3. Хеш-функція зашифровується секретним ключем:

$$c_i=(h_i)^e \bmod n=2^{601} \bmod 13231 = 1012_{10}=00001111110100_2.$$

4. Додаємо цифровий підпис до повідомлення:

$$m'=(010000010100010, 1111110100).$$

5. Тепер приймач повідомлення m’ може ідентифікувати відправника повідомлення, знаючи відкритий ключ.

На приймальній стороні порядок перевірки відправника такий:

1. Дешифрується цифровий підпис з використанням відкритого ключа:
$$h_i=(c_i)^d \bmod n = 1012^{2401} \bmod 13231 = 2.$$
2. Обчислюється хеш-функція h’ на основі інформаційної частини m=(010000010100010) повідомлення m’.
3. Обчислене значення хеш-функції порівнюється з дешифрованим, якщо вони не дорівнюють одне одному h≠h’, то повідомлення не належить відправнику або спотворене.

Приклад 4.12. Аліса і Боб вирішили обмінюватись шифрованими повідомленням, але їм потрібно узгодити ключ шифрування. Аліса знайшла в Інтернеті протокол обміну криптографічними ключами Діффі-Геллмана [31], який знав також Боб. Аліса вибрала два простих числа g=13, p=29 і секретний ключ a=8, невідомий Бобу, а Боб вибрав секретний ключ b=17, невідомий Алісі. Як Боб і Аліса визначили спільний секретний ключ, не передаючи його по лінії зв’язку?

Розв'язання. Аліса виконала такі обчислення та дії:

1. Визначила число A:
   $A=g^a \bmod p=13^8 \bmod 29=16$.
2. Передала по Бобу такі дані: A, g, p

Боб, отримавши A, g, p зробив таке:

1. Визначив число B, яке передав Алісі:
   $B=g^b \bmod p=13^{17} \bmod 29=22$
3. Визначив спільний секретний ключ:
   $K=A^b \bmod p=16^{17} \bmod 29=7$.

Аліса, отримавши B, також визначила спільний секретний ключ:

$K=B^a \bmod p=22^8 \bmod 29=7$.

Ключі K у Аліси і Боба збіглися, тепер вони можуть обмінюватись шифрованими повідомленнями.

4.3 Контрольні питання і задачі

1. Які методи захисту інформації застосовують в ОС?
2. Що таке матриця доступу?
3. Поясніть застосування списків доступу.
4. Які недоліки методів захисту інформації за допомогою паролів, матриць доступу та списків доступу?
5. Які питання розглядає криптологія?
6. Чим відрізняється криптографія від криптоаналізу?
7. Хто є основоположником наукової криптології?
8. Які основні розділи містить криптографія?
9. Наведіть класифікацію алгоритмів шифрування інформації.
10. Охарактеризуйте шифри Віжінера, Цезаря, Вернама.
11. В чому різниця між DES і RSA схемами шифрування?
12. Поясніть принципи роботи алгоритму DES.
13. Поясніть принципи роботи алгоритму RSA.
14. Поясніть порядок генерацію відкритого і секретного ключів в RSA.
15. Поясніть роботу узагальненого алгоритму Евкліда за генерації ключів в RSA.
16. Наведіть алгоритм виконання операції піднесення до степеня за модулем справа наліво.
17. Наведіть алгоритм виконання операції піднесення до степеня за модулем зліва направо.
18. Поясніть принципи шифрування інформації за допомогою генератора ПВЧ.
19. Які алгоритми ущільнення найбільш прийнятно використовувати для шифрування даних?
20. Охарактеризуйте комп'ютерну стеганографію.
21. Дайте порівняльний аналіз алгоритмів шифрування інформації.

Задачі

1. Дешифрувати шифротекст RIVWUCPDN, отриманий за допомогою шифру Віжінера з ключем PIES (див. прикл. 4.1). Відповідь: CAREFFULY.
2. Дешифрувати шифротекст 158, 26, 6, отриманий в прикладі 4.2. Відповідь: 65, 66, 67.
3. Зашифрувати текст з використанням шифру Віжінера (див. прикл. 4.1) і ключа VIGINERE : QRSTUVWXYZ. Відповідь: LZYBGZNBTG.
4. Зашифрувати з використанням конгруентного генератора ПВЧ таку послідовність байтів: 170, 85, 255. Параметри генератора ПВЧ такі: a=9, c=11, m=256, T_0=10. Відповідь: 207, 205, 156.
5. За допомогою алгоритму Евкліда знайти gcd (21, 12), gcd (35, 15), gcd (45, 27), gcd (33, 16). Відповідь: 3, 5, 9, 1.
6. Визначити з використанням алгоритму Евкліда, які з пар чисел є взаємно прості: (25, 12), (25, 15), (13, 39), (40, 27). Відповідь: (25, 12), (40, 27).
7. За допомогою узагальненого алгоритму Евкліда знайти значення x і y в рівняннях:

   $21x + 12y = \gcd(21, 12)$;
   $35x + 15y = \gcd(35, 15)$;
   $45x + 27y = \gcd(45, 27)$;
   $33x + 16y = \gcd(33, 16)$.

   Відповідь: 3, -1, 2; 5, 1, -2; 9, -1, 2; 1, 1, -2.
8. Обчислити, використовуючи швидкі алгоритми піднесення до степеня, такі вирази: $2^8 \bmod 10$, $3^7 \bmod 10$, $7^{19} \bmod 100$, $7^{57} \bmod 100$. Відповідь: 6, 7, 43, 7.
9. Знайти відкритий і секретний ключі в RSA, якщо p=11, q=17. Відповідь: e=13, d=37.
10. Дешифрувати методом RSA повідомлення c_i=23, якщо p=5, q=11, e=3. Відповідь: m_i=12.
11. Зашифрувати методом RSA повідомлення m_i=15, якщо p=7, q=11, e=13. Відповідь: c_i=64.
12. Розробити програму шифрування-дешифрування файлів методом Віжінера. Ключ шифру 8 байт. Мова програмування C++ .
13. Розробити програму шифрування-дешифрування файлів методом гамування. Ключ шифру 2 байти. Мова програмування C++ .
14. Розробити програму шифрування-дешифрування файлів методом RSA. Мова програмування C++ .

ВИСНОВКИ

З огляду на зростання ролі комп'ютерних мереж у повсякденному житті людей, інтерес до теорії інформації та її невід'ємних складових – кодування і криптології – знову збільшується. Причому, широке застосування знаходять однаковою мірою три основні складові теорії інформації, які мають прикладний характер: економне кодування, завадостійке кодування та криптологія. Методи теорії інформації, які раніше були лише предметом теоретичних досліджень, стали основою промислових стандартів, що використовуються у мережах. Це викликано тим, що через мережі передаються повідомлення не тільки розважального характеру, але і важливі як з погляду окремої людини, так і для суспільства в цілому, зокрема і секретні. Пошкодження або потрапляння в руки зловмисників таких документів може призвести до непередбачуваних наслідків.

Про зростання інтересу до теорії інформації та її складових свідчить поява в останні роки сайтів в Інтернет, які спеціалізуються на поширенні безкоштовної навчальної інформації про методи теорії інформації. Можна відзначити наприклад сайт www.compression.ru, який дає всебічну інформацію про методи ущільнення даних та про сучасний стан досліджень в цій галузі. Основним автором цього сайту є Ватолін Д., там само розміщені і глави його підручника [13]. Є також ряд сайтів, які дозволяють безкоштовно для некомерційного використання скачувати відсканові посібники, підручники, монографії з теорії інформації як ті, що давно видавались і їх практично неможливо знайти в паперовому вигляді, так і ті, що тільки вийшли з друку.

Деякі з них: www.mirknig.com, www.knigka.info, www.ihtik.lib.ru, http://lib.prometey.org, www.litportal.kiev.ua, http://physicsbooks.narod.ru.

Цей навчальний посібник можна розглядати як друге видання посібника «Кодування та захисту інформації» автора Майданюка В. П. від 2009 року [2]. У посібнику зменшено кількість теоретичного матеріалу до рівня необхідного для розв'язання типових задач, але значно збільшено кількість прикладів розв'язання типових задач, що буде сприяти кращому засвоєнню навчальних матеріалів з орієнтацією на практичне застосування. Крім того, посібник містить контрольні питання і задачі для самостійного розв'язування.

ЛІТЕРАТУРА

1. Шеннон К. Работы по теории информации и кибернетике / под ред. Р. Л. Добрушина и О. Б. Лупанова. – Пер. с англ. – М. : Иностранная литература, 1963. – 106 с.
2. Майданюк В. П. Кодування та захист інформації : навчальний посібник. – Вінниця : ВНТУ, 2009. – 164 с.
3. Дмитриев В. И. Прикладная теория информации. – М. : Высш. шк., 1989. – 420 с.
4. Кузьмин И. В., Кедрус В. А. Основы теории информации и кодирования. – К. : Вища шк., 1986. – 238 с.
5. Хэмминг Р. В. Теория кодирования и теория информации. – Пер. с англ. – М. : Радио и связь, 1983. – 176 с.
6. Харкевич А. А. Борьба с помехами. – М. : Наука, 1965. – 270 с.
7. Колмогоров А. Н. Три подхода к определению понятия «количество информации» // Проблемы передачи информации. – 1965. – № 1. – С. 3–11.
8. Зюко А. Г. Элементы теории передачи информации. – К. : Техніка, 1969. – 300 с.
9. Питерсон У., Уэлдон Э. Коды, исправляющие ошибки. – Пер. с англ. – М. : Мир, 1976. – 589 с.
10. Шульгин В. И. Основы теории передачи информации. Ч. 1. Экономное кодирование : учебное пособие. – Харьков : Нац. аэрокосм. ун-т «Харьк. авиац. ин-т», 2003. – 102 с.
11. Таненбаум Э., Уэзеролл Д. Компьютерные сети. – СПб. : Питер, 2012. – 960 с.
12. Жураковський Ю. П., Гніліцький В. В. Теорія інформації та кодування в задачах : навчальний посібник. – Житомир : ЖІТІ, 2002. – 230 с.
13. Ватолин Д., Ратушняк А., Смирнов М., Юкин В. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео. – М. : ДИАЛОГ-МИФИ, 2002. – 384 с.
14. Лужецький В. А. Високонадійні математичні Фібоначчі-процесори : монографія. – Вінниця : ВДТУ, 2000. – 248 с.
15. Майданюк В. П. Методи і засоби комп'ютерних інформаційних технологій. Кодування зображень : навчальний посібник. – Вінниця : ВДТУ, 2001. – 63 с.
16. Балашов К. Ю. Сжатие информации: анализ методов и подходов. – Минск : 2000. – 42 с.
17. Семенюк В. В. Экономное кодирование дискретной информации. – СПб. : СПбГИТМО (ТУ), 2001. – 115 с.
18. Кричевский Р. Е. Сжатие и поиск информации. – М. : Радио и связь, 1989. – 168 с.

19. Фомин А. А. Основы сжатия информации. – СПб. : СПбГТУ, 1998. – 85 с.
20. Гонсалес Р., Вудс Р. Цифровая обработка изображений. – М. : Техносфера, 2005. – 1072 с.
21. Сэломон Д. Сжатие данных, изображений и звука. – М. : Техносфера, 2004. – 368 с.
22. Вернер М. Основы кодирования : учебник для ВУЗов. – М. : Техносфера, 2004. – 288 с.
23. Золотарёв В. В., Овечкин Г. В. Помехоустойчивое кодирование. Методы и алгоритмы : Справочник / Под. ред. чл.-кор. РАН Ю. Б. Зубарева. – М. : Горячая линия-Телеком, 2004. – 126 с.
24. Блейхут Р. Теория и практика кодов, контролирующих ошибки. – Пер. с англ. – М. : Мир, 1986. – 576 с.
25. Касами Т., Токура Н, Ивадари Е., Инагаки Я. Теория кодирования. – Пер. с японск. – М. : Мир, 1978. – 576 с.
26. Прокис Дж. Цифровая связь / Под ред. Д. Д. Кловского. – . Пер. с англ. – М. : Радио и связь, 2000. – 800 с.
27. Скляр Б. Цифровая связь и практическое применение. – Пер. с англ. – М. : Издательський дом «Вильямс», 2003. – 1104 с.
28. Шульгин В. И. Основы теории передачи информации. Ч. 2. Помехоустойчивое кодирование : учебное пособие. – Харьков : Нац. аэрокосм. ун-т «Харьк. авиац. ин-т», 2003. – 87 с.
29. Кларк Дж., Кейн Дж. Кодирование с исправлением ошибок в системах цифровой связи. – Пер. с англ. – М. : Радио и связь, 1987. – 392 с.
30. Банкет В. Л. Сверточные коды в системах передачи информации : учеб. пособие. – Одесса : Одесск. электротехн. ин-т связи им. А. С. Попова, 1986. – 57 с.
31. Технології захисту інформації [Електронний ресурс] : підручник для студ. спеціальності 122 «Комп'ютерні науки», спеціалізацій «Інформаційні технології моніторингу довкілля», «Геометричне моделювання в інформаційних системах» / Ю. А. Тарнавський; КПІ ім. Ігоря Сікорського. – Електронні текстові дані (1 файл: 2,04 Мбайт). – Київ : КПІ ім. Ігоря Сікорського, 2018. – 162 с.
32. Шнайдер Б. Секреты и ложь. Безопасность данных в цифровом мире. – СПб. : Питер, 2003. – 386 с.
33. Белов Е. Б., Лось В. П., Мещеряков Р. В., Шелупанов А. А. Основы информационной безопасности : учебное пособие для вузов. – М. : Горячая линия–Телеком, 2006. – 544 с.
34. Хорошко В. О., Азаров О. Д., Шелест Ю. С., Яремчук Ю. Є. Основи комп'ютерної стеганографії : навчальний посібник. – Вінниця : ВДТУ, 2003. – 143 с.
35. Брассар Ж. Современная криптология. – Пер. с англ. – М. : Издательско-полиграфическая фирма ПОЛИМЕД, 1999. – 176 с.

36. Рябко Б. Я., Фионов А. Н. Криптографические методы защиты информации : учебное пособие для вузов. – М. : Горячая линия-Телеком, 2005. – 229 с.
37. Бабенко Т. В., Гулак Г. М., Сушко С. О., Фомичова Л. Я. Криптологія у прикладах, тестах і задачах : навч. посібник. – Д. : Національний гірничий університет, 2013. – 318 с.
38. ГОСТ 28147-89. Системы обработки информации. Защита криптографическая. Алгоритмы криптографического преобразования.
39. Методичні вказівки до виконання лабораторних робіт з дисципліни «Операційні системи ЕОМ» / Укладачі: Майданюк В. П., Романюк О. Н. – Вінниця : ВДТУ, 1997. – 41 с.
40. Соколов А. В., Степанюк О. М. Защита от компьютерного терроризма. – СПб. : БХВ-Петербург, 2002. – 496 с.
41. Казарин О. В. Безопасность программного обеспечения компьютерных систем. – М. : МГУЛ, 2003.
42. Беляев А. Методы и средства защиты информации. Курс лекций в Череповецком филиале СПбГТУ. – Режим доступу: http://science.metacom.ru
43. Каплун В. А., Майданюк В. П. Захист операційних систем : навчальний посібник. – Вінниця : ВНТУ, 2007. – 185 с.

СЛОВНИК ОСНОВНИХ ТЕРМІНІВ (GLOSSARY)

Алгоритм – algorithm
Апостеріорний – aposterior
Апріорний – aprior
Відкритий ключ – public key
Двійкова одиниця – binary digit(bit)
Двійковий симетричний канал – binary symmetric channel
Двійкові коди – binary codes
Декодер – decoder
Дешифрування – decryption
Джерело без пам’яті – memoryless source
Джерело з пам’яттю – source with memory
Джерело повідомлень – source of messages
Дискретний – discrete
Ентропія – entropy
Ергодичність – ergodic
Завада – noise
Закритий ключ – private key
Зв’язок – communication
Знак – character
Імовірність – probability
Канал без завад – noiseless channel
Канал з завадами – channel with noise
Канали без пам’яті – memoryless channel
Канали з пам’яттю – channels with memory
Кібернетика – cybernetics
Кодек – codec
Кодер – coder
Кодування – coding, encoding
Кодування без втрат – lossless coding
Кодування з втратами – lossy coding
Комбінаторний – combinatory
Корегувальні коди – correctings codes
Криптографія – cryptography
Криптологія – criptologic
Ланцюг Маркова – Markov chain
Миттєві коди – instantaneous code
Модель – model
Надмірність – redundancy
Неперервний – continuous
Оптимальне кодування – optimal coding
Пам’ять – memory

Парність – parity
Префіксні коди – prefix codes
Програмне забезпечення – software
Продуктивність джерела – source productivity
Пропускна здатність – capacity
Сигнал – signal
Синтез – synthesis
Стан – state
Стаціонарний – stationary
Теорія інформації – information theory
Швидкість маніпуляції – manipulation speed
Швидкість передачі інформації – information rate
Шифрування – encryption

Електронне навчальне видання
комбінованого використання.
Можна використовувати в локальному та мережному режимах

Володимир Павлович Майданюк
Олександр Никифорович Романюк
Станіслав Євгенович Тужанський

ОСНОВИ ТЕОРІЇ ІНФОРМАЦІЇ ТА КОДУВАННЯ

Навчальний посібник

Рукопис оформлено *В. Майданюком*

Редактор *Т. Старічек*

Оригінал-макет виготовлено *Т. Старічек*

Підписано до видання 09.08.2022.
Гарнітура Times New Roman.
Зам. № Р2022-063.

Видавець та виготовлювач
Вінницький національний технічний університет,
Редакційно-видавничий відділ.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95, м. Вінниця, 21021.
Тел. (0432) 65-18-06.
press.vntu.edu.ua;
E-mail: irvc.ed.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.