

Materiały pomocnicze do kursu Kodowanie – ćwiczenia

Kod kursu ETE0055C
Semestr letni, rok akad. 2004 / 2005

Przygotował Piotr Kocyan
pok. 804 C-5

Spis treści

1	INFORMACJE OGÓLNE.....	4
1.1	ZASADY ORGANIZACYJNE.....	4
1.2	ZASADY ZALICZENIA KURSU	4
2	WIADOMOŚCI PODSTAWOWE	5
2.1	LICZBY BINARNE — PODSTAWOWE DZIAŁANIA	5
2.2	DZIELENIE WIELOMIANÓW.....	7
2.3	WAGA HAMMINGA	8
2.4	ODLEGŁOŚĆ HAMMINGA	9
3	KODY BLOKOWE LINIOWE I CYKLICZNE	10
3.1	KODY LINIOWE — WŁAŚCIWOŚĆ LINIOWOŚCI.....	12
3.2	KODY CYKLICZNE — WŁAŚCIWOŚĆ CYKLICZNOŚCI.....	14
3.3	ODLEGŁOŚĆ MINIMALNA KODU	16
3.4	ZDOLNOŚĆ DETEKCYJNA I KOREKCYJNA.....	17
4	TEST NR 1 — ZADANIA.....	19
5	KODOWANIE INFORMACJI	23
5.1	KODOWANIE ZA POMOCĄ WIELOMIANU	23
5.2	KODOWANIE ZA POMOCĄ MACIERZY GENERUJĄCEJ.....	26
6	TEST NR 2 — ZADANIA.....	30
7	DEKODOWANIE KOREKCYJNE	33
7.1	SYNDROM	33
7.2	MACIERZ KOREKCYJNA	37
7.3	UPROSZCZONY ALGORYTM DEKODOWANIA DLA KODÓW CYKLICZNYCH	39
8	TEST NR 3 — ZADANIA.....	44

Wykaz oznaczeń

Wielkości wektorowe oznaczono czcionką pogrubioną, a wielkości skalarne czcionką zwykłą. Zmienne zostały oznaczone kursywą.

$GF(p)$ – ciało Galois (*Galois Field*) o liczbie elementów równej p

$w_H(\mathbf{u})$ – waga Hamminga wektora \mathbf{u}

$d_H(\mathbf{u}, \mathbf{v})$ – odległość Hamminga między wektorami \mathbf{u} i \mathbf{v}

n – długość słowa kodowego

k – długość słowa informacyjnego (części informacyjnej słowa kodowego)

d – odległość minimalna kodu

l – zdolność detekcyjna kodu

t – zdolność korekcyjna kodu

\mathbf{G} – macierz generująca

\mathbf{H} – macierz korekcyjna

Poniżej podane wektory mają swoje odpowiedniki w zapisie wielomianowym np. $\mathbf{u} \leftrightarrow u(x)$

\mathbf{c} – wektor kodowy (*codeword*)

$\mathbf{c}^{(+i)}$ – wektor kodowy przesunięty cyklicznie o i pozycji w lewo

$\mathbf{c}^{(-i)}$ – wektor kodowy przesunięty cyklicznie o i pozycji w prawo

\mathbf{e} – wektor błędu (*error*)

\mathbf{g} – wektor generujący (wielomian generujący)

\mathbf{m} – wektor informacyjny (*message*)

\mathbf{r} – reszta z dzielenia dwóch wektorów (wielomianów)

\mathbf{s} – wektor reprezentujący syndrom

\mathbf{u}, \mathbf{v} – dowolne słowa binarne lub wektory

\mathbf{w} – wynik dzielenia dwóch wektorów (wielomianów)

1 Informacje ogólne

1.1 Zasady organizacyjne

Kurs składa się z siedmiu spotkań zarówno dla zajęć prowadzonych w tygodniach parzystych jak i nieparzystych. Ósme terminy tygodni nieparzystych są traktowane jako terminy dodatkowe przeznaczone na powtórne pisanie testów. Poniżej przedstawiono plan i kalendarz zajęć kursu.

Tabela 1 Plan kursu

Termin	Temat zajęć
1	Podanie zasad organizacyjnych, wprowadzenie do metod ochrony informacji przed błędami i zastosowań kodowania w telekomunikacji
2	Ćwiczenia obejmujące zadania dotyczące rozdziału 2. i 3.
3	Test nr 1 (60 min), po teście omówienie zadań i podanie prawidłowych odpowiedzi
4	Ćwiczenia obejmujące zadania dotyczące rozdziału 5.
5	Test nr 2 (60 min), po teście omówienie zadań i podanie prawidłowych odpowiedzi
6	Ćwiczenia obejmujące zadania dotyczące rozdziału 7.
7	Test nr 3 (60 min), po teście omówienie zadań i podanie prawidłowych odpowiedzi

Tabela 2 Kalendarz zajęć w semestrze letnim roku akad. 2004/2005

Termin Grupa	1	2	3	4	5	6	7	dodatk.
PN/N 15:15 s. 32 C-4	21-02	7-03	21-03	4-04	18-04	16-05	30-05	
WT/N 13:15 s. 32 C-4	22-02	8-03	22-03	5-04	19-04	17-05	31-05	14-06
ŚR/N 15:15 s. 33 C-4	23-02	9-03	23-03	6-04	20-04	4-05	18-05	1-06
ŚR/N 17:05 s. 32 C-4	23-02	9-03	23-03	6-04	20-04	4-05	18-05	1-06
CZ/N 11:15 s. 32 C-4	24-02	10-03	7-04	21-04	5-05	2-06	8-06	13-06
CZ/N 13:15 s. 32 C-4	24-02	10-03	7-04	21-04	5-05	2-06	8-06	13-06
CZ/N 17:05 s. 33 C-4	24-02	10-03	7-04	21-04	5-05	2-06	8-06	13-06
PN/P 15:15 s. 32 C-4	28-02	14-03	11-04	25-04	9-05	23-05	6-06	
WT/P 13:15 s. 32 C-4	1-03	15-03	12-04	26-04	10-05	24-05	7-06	
ŚR/P 15:15 s. 33 C-4	2-03	16-03	30-03	13-04	27-04	11-05	25-05	
CZ/P 11:15 s. 32 C-4	3-03	17-03	31-03	14-04	28-04	12-05	9-06	
CZ/P 13:15 s. 32 C-4	3-03	17-03	31-03	14-04	28-04	12-05	9-06	

Każda osoba wpisana na kurs może przyjść na wszystkie terminy dodatkowe. Co najmniej 7 dni przed danym terminem dodatkowym, na drzwiach pok. 804 C-5 zostanie wywieszona lista z liczbą pozycji odpowiadającą liczbie miejsc w sali. Na listę należy wpisać nr indeksu oraz nr testu, który osoba zainteresowana zamierza pisać w danym terminie. Na każdym terminie dodatkowym można pisać jeden, dowolny test. Punkty uzyskane z testu pisanego w terminie dodatkowym **bezwzględnie zastępują** poprzednio uzyskane punkty z danego testu.

1.2 Zasady zaliczenia kursu

Ocena końcowa jest obliczana na podstawie średniej arytmetycznej wyników uzyskanych z trzech testów. Wynik z każdego testu stanowi stosunek liczby uzyskanych punktów do maksymalnej liczby punktów możliwych do uzyskania na danym teście wyrażony w punktach procentowych. Szczegółowe zasady punktacji zadań w poszczególnych testach są podane w rozdziałach zawierających pytania (rozdziały 4., 6. i 8.). Sposób obliczenia oceny końcowej z kursu podano w poniższej tabeli.

Średnia punktów z trzech testów	Ocena końcowa
50 – 59	dostateczny
60 – 69	dostateczny +
70 – 79	dobry
80 – 89	dobry +
90 – 100	bardzo dobry

2 Wiadomości podstawowe

2.1 Liczby binarne — podstawowe działania

Najbardziej rozpowszechnionym systemem liczbowym jest system dziesiętny, jednak w technice cyfrowej stosuje się również inne systemy liczbowe, których podstawą jest liczba 2 lub jej wielokrotność. Przykładami takich systemów liczbowych są: system binarny (dwójkowy), ósemkowy i szesnastkowy (heksadecymalny).

Nazwa każdego systemu liczbowego jest związana z jego podstawą, która określa liczbę możliwych wartości, które może przyjąć każda cyfra liczby. W zależności od pozycji, na której znajduje się cyfra, ma ona przypisaną odpowiednią wagę. Wagi kolejnych cyfr stanowią kolejne potęgi podstawy systemu liczbowego. Najmniej znacząca cyfra jest zapisywana po prawej stronie (potęga podstawy równa 0), a najbardziej znacząca po lewej stronie liczby. Wartość liczby oblicza się jako sumę iloczynów wag i odpowiadających im cyfr.

W systemie dziesiętnym, którego podstawą jest liczba 10, każda cyfra liczby może przyjąć jedną z dziesięciu wartości ($0 \div 9$), natomiast w systemie binarnym każda cyfra liczby może przyjąć jedną z dwóch wartości (0 lub 1). Poniżej przedstawiono przykład obliczenia wartości liczb zapisanych w systemie dziesiętnym i binarnym.

Liczby dziesiętne					Liczby binarne					
cyfry	1	9	5	1	cyfry	1	0	0	1	1
wagi	10^3	10^2	10^1	10^0	wagi	2^4	2^3	2^2	2^1	2^0
wartość	$1 \cdot 10^3 + 9 \cdot 10^2 + 5 \cdot 10^1 + 1 \cdot 10^0 = 1951$				wartość	$1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 19$				

W dalszych rozważaniach n -cyfrowa liczba binarna będzie nazywana **n -bitowym słowem** lub n -bitowym ciągiem binarnym, a poszczególne cyfry będą nazywane **bitami**. Liczbę taką, w odniesieniu do kodów nad GF(2), można również nazywać **wektorem**, a poszczególne cyfry jego **współrzednymi**.

Ciągi binarne często wygodniej jest zapisać w postaci wielomianowej. Pozwala to na opuszczenie wszystkich bitów mających wartość 0 i zapisanie tylko bitów o wartości 1. Opuszczenie niektórych bitów (o wartości 0) wymaga zapisania pozycji bitów mających wartość 1. Pozycje te przedstawia się w formie potęg pewnej symbolicznej zmiennej x i numeruje się zaczynając od 0. Przy takim sposobie numeracji potęga przy x jest równa potędze wagi odpowiadającego jej bitu w liczbie binarnej.

$$10011101 \leftrightarrow x^7 + x^4 + x^3 + x^2 + 1$$

Podstawową operacją używaną w procesie kodowania i dekodowania jest operacja dodawania bitowego, lub też dodawania wektorowego realizowanego jako dodawanie odpowiadających sobie współrzednych wektora. Z uwagi na to, że każdy bit (współrzedna) może przyjąć tylko dwie wartości (0 lub 1), dodawanie dwóch skalarów (bitów) a i b w GF(2) zdefiniowane jest następująco:

$$a \oplus b = (a + b) \bmod 2 \quad (1)$$

gdzie operator „mod” oznacza modulo czyli resztę z dzielenia.

Operator \oplus reprezentuje „dodawanie modulo dwa” i w operacjach bitowych jest oznaczany jako *ExOR* lub *ExclusiveOR*. W dalszych rozważaniach, przy operacjach na słowach binarnych i wielomianach, operator \oplus dla uproszczenia zapisu będzie oznaczany $+$.

Operację modulo można zdefiniować w przestrzeni liczb całkowitych następująco:

$$y = n \cdot x + r \Leftrightarrow y \bmod x = r \quad (2)$$

gdzie:

x, y – dowolne liczby całkowite,

n – pewna liczba całkowita,

r – liczba całkowita należąca do przedziału $\langle 0, x \rangle$.

Przykład 1

	$7 \bmod 3 = 1$
ponieważ	$7 = 2 \cdot 3 + 1$

Analizując operację dodawania w $GF(2)$, zdefiniowaną zależnością (1), można zauważyć, że:

$$1 + 1 = 0$$

więc

$$1 = 0 - 1$$

czyli

$$1 = -1$$

Jak widać, w $GF(2)$ operacja dodawania jest równoważna operacji odejmowania, stąd działanie *ExOR* często zwane jest różnicą symetryczną. W dalszych rozważaniach, przy operacjach na słowach binarnych i wielomianach, operacja dodawania może być rozumiana również jako odejmowanie.

Zakładając, że a jest elementem ciała $GF(2)$ (czyli 0 lub 1) można pokazać podstawowe własności dodawania w $GF(2)$:

1. Wynik dodawania dwóch bitów o tej samej wartości jest równy 0:

$$a + a = 0$$

2. Wynik dodawania dwóch bitów o przeciwnej wartości jest równy 1:

$$a + \bar{a} = 1$$

3. Dodanie bitu o wartości 0 do innego bitu nie zmienia jego wartości:

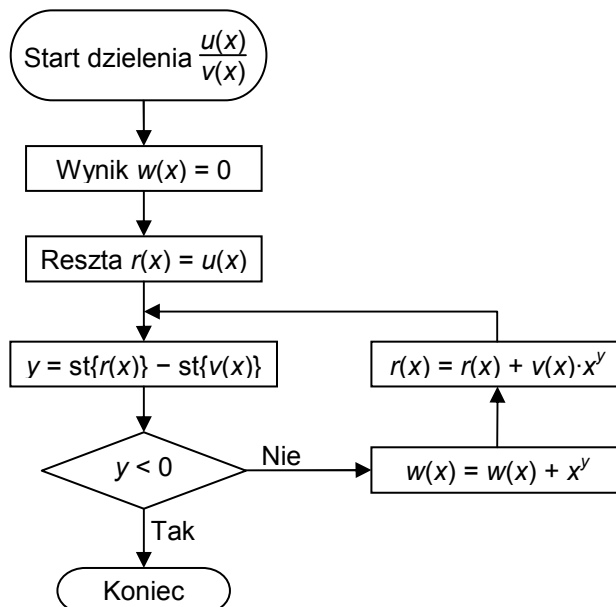
$$a + 0 = a$$

4. Dodanie bitu o wartości 1 do innego bitu zmienia jego wartość (negacja):

$$a + 1 = \bar{a}$$

2.2 Dzielenie wielomianów

Dzielenie wielomianów jest jednym z podstawowych działań używanych w procesie kodowania i dekodowania informacji za pomocą kodów cyklicznych. Algorytm dzielenia został pokazany na poniższym rysunku.



Rys. 1 Algorytm dzielenia wielomianów

Przykład 2

Podzielić wielomian $x^7 + x^5 + x^4 + x + 1$ przez wielomian $x^2 + x$.

$$\begin{array}{r}
 x^5 + x^4 + x^2 + x + 1 \\
 x^2 + x \overline{) x^7 + x^5 + x^4 + x + 1} \\
 \underline{x^7 + x^6} \\
 x^6 + x^5 + x^4 + x + 1 \\
 \underline{x^6 + x^5} \\
 x^4 + x + 1 \\
 \underline{x^4 + x^3} \\
 x^3 + x + 1 \\
 \underline{x^3 + x^2} \\
 x^2 + x + 1 \\
 \underline{x^2 + x} \\
 1
 \end{array}$$

Wynikiem dzielenia jest wielomian $x^5 + x^4 + x^2 + x + 1$, a reszta z dzielenia wynosi 1 (czyli wielomian x^0).

Na podstawie powyższego przykładu oraz (2) można napisać:

$$\begin{aligned}
 (x^7 + x^5 + x^4 + x + 1) \bmod (x^2 + x) &= 1 \\
 x^7 + x^5 + x^4 + x + 1 &= (x^5 + x^4 + x^2 + x + 1) \cdot (x^2 + x) + 1
 \end{aligned}$$

Oczywiście, dzielenie wielomianów ma sens tylko wtedy, gdy stopień wielomianu dzielonego (dzielnej) jest większy lub równy stopniowi wielomianu, przez który dzielimy (dzielnika). W przeciwnym wypadku wynikiem dzielenia jest wielomian zerowy, a reszta jest

równa dzielnej. Można również zauważyć, że stopień reszty z dzielenia będzie zawsze mniejszy od stopnia dzielnika oraz stopień wyniku dzielenia jest równy różnicy stopni dzielnej i dzielnika.

Dzielenie można również przeprowadzić na reprezentacji bitowej wielomianów pokazanych w poprzednim przykładzie:

Przykład 3

Podzielić słowo [10110011] przez słowo [110].

$$\begin{array}{r}
 110111 \\
 110 \overline{) 10110011} \\
 \underline{110} \\
 1110011 \\
 \underline{110} \\
 010011 \\
 \underline{000} \\
 10011 \\
 \underline{110} \\
 1011 \\
 \underline{110} \\
 111 \\
 \underline{110} \\
 01
 \end{array}$$

Wynikiem dzielenia jest słowo [110111], a reszta z dzielenia wynosi [1].

Jak widać, zarówno wynik jak i reszta z dzielenia są zgodne z tymi uzyskanymi poprzednio. Należy tutaj zaznaczyć, że przy dzieleniu bitowym można opuścić wiersze następujące bezpośrednio po reszcie pośredniej rozpoczynającej się zerem, jednak zwiększa to ryzyko popełnienia błędu. Oczywiście nie trzeba do każdego wiersza reszty przepisywać wszystkich bitów z ciągu dzielonego – wystarczy dopisywać na bieżąco po jednym bicie w każdym następnym wierszu, jednak przepisywanie wszystkich bitów zmniejsza prawdopodobieństwo pomyłki.

2.3 Waga Hamminga

Waga Hamminga wektora jest równa liczbie niezerowych jego współrzędnych. Dla ciągów (słów) binarnych waga Hamminga jest równa liczbie bitów o wartości równej 1. Należy zaznaczyć, że **pojęcie wagi Hamminga dotyczy jednego słowa (wektora)** i jest pojęciem ogólnym definiowanym dla dowolnego słowa, które nie musi być słowem kodowym. W postaci wielomianowej, z uwagi na zapisywanie pozycji bitów (współrzędnych) mających wartości równe 1, waga Hamminga jest równa liczbie składników wielomianu.

$$\begin{aligned}
 w_H(10011) &= 3 \\
 w_H(x^8 + x^5 + x) &= 3
 \end{aligned}$$

2.4 Odległość Hamminga

Odległość Hamminga między dwoma słowami (wektorami) można zdefiniować jako liczbę odpowiadających sobie współrzędnych posiadających różne wartości. Dla ciągów binarnych jest ona równa liczbie pozycji, na których bity jednego słowa mają wartości różne od wartości bitów na tych samych pozycjach w drugim słowie. Oczywiście **odległość Hamminga jest określana dla dwóch dowolnych słów binarnych o jednakowej długości**. Odległość Hamminga między dwoma słowami mówi o tym jak bardzo (na ilu pozycjach) różnią się te słowa.

Odległość między dwoma słowami binarnymi można uzyskać poprzez obliczenie wagi Hamminga ich sumy (oczywiście modulo 2). Wynika to z tego, że suma bitów o jednakowych wartościach jest równa 0, a suma bitów o różnych wartościach jest równa 1, więc zliczenie bitów o wartości 1 w sumie tych słów (czyli wyznaczenie wagi Hamminga) odpowiada odległości między tymi słowami. Można zatem dla dwóch słów u i v zapisać:

$$d_H(u, v) = w_H(u + v) \quad (3)$$

W postaci wielomianowej odległość wyznacza się podobnie, tzn. najpierw sumuje się dwa wielomiany, a następnie określa się liczbę ich składników.

Przykład 4

Oblicz odległość Hamminga między dwoma słowami $u = [100111]$ i $v = [010011]$.

$$\begin{array}{r} 100111 \\ + 010011 \\ \hline = 110100 \end{array}$$

czyli:

$$d_H(u, v) = w_H(110100) = 3$$

Odległość Hamminga między wektorami u i v wynosi 3.

Można łatwo zauważyć, że dla każdego n -bitowego słowa binarnego istnieje tyle n -bitowych słów leżących w odległości Hamminga d od niego, ile jest kombinacji d -jedynkowych w słowach n -bitowych. Wynika to z liczby możliwych sum, o wadze równej d , dwóch n -bitowych ciągów binarnych. Liczbę tych słów M_d można zapisać za pomocą symbolu Newtona:

$$M_d = \binom{n}{d}$$

Przykład 5

Oblicz liczbę słów leżących w odległości 1, 2, 3 i 4 od słowa $[1011]$.

$$M_1 = \binom{4}{1} = \frac{4!}{(4-1)! \cdot 1!} = 4 \quad \text{Słowa: 1010, 1001, 1111, 0011}$$

$$M_2 = \binom{4}{2} = \frac{4!}{(4-2)! \cdot 2!} = 6 \quad \text{Słowa: 1000, 1110, 0010, 1101, 0001, 0111}$$

$$M_3 = \binom{4}{3} = \frac{4!}{(4-3)! \cdot 3!} = 4 \quad \text{Słowa: 1100, 0000, 0110, 0101}$$

$$M_4 = \binom{4}{4} = \frac{4!}{(4-4)! \cdot 4!} = 1 \quad \text{Słowo: 0100}$$

3 Binarne kody blokowe liniowe i cykliczne

W systemach cyfrowych często stosuje się podział przesyłanej lub przechowywanej informacji na bloki o stałej długości (np. dyski twarde, płyty CD, DVD). Podstawą ochrony przesyłanych bloków przed błędami jest wykrycie błędów po stronie odbiorczej, czyli odróżnienie błędnych danych od prawidłowych. Jeżeli przesyła się dane niekodowane, to każdy ciąg bitów docierający do odbiornika może stanowić nadaną informację i odróżnienie informacji błędnej od prawidłowej nie jest możliwe.

W celu umożliwienia wykrycia błędów należy ze zbioru wszystkich możliwych słów o długości równej długości przesyłanego bloku wybrać podzbiór słów uznawanych za prawidłowe. Taki podzbiór nazywamy **kodem**, a jego elementy **słowa (wektorami) kodowymi**.

Kody korekcyjne można podzielić na kody blokowe i kody splotowe. Wśród kodów blokowych wyróżnia się kody liniowe i kody cykliczne, przy czym kody cykliczne są podklasą kodów liniowych (spełniają kryterium liniowości). Kody blokowe oznacza się symbolem (n, k) , gdzie n jest **długością słowa kodowego**, a k jest **długością części informacyjnej**. Każde słowo kodowe składa się z części informacyjnej i części korekcyjnej (nadmiarowej). W trakcie kodowania ciąg informacyjny dzieli się na k -elementowe bloki, na podstawie których oblicza się $(n-k)$ -elementową część nadmiarową. Część nadmiarowa w kodach liniowych może być umieszczona przed lub za częścią informacyjną, jednak w kodach cyklicznych, z uwagi na dużo łatwiejszą realizację sprzętową koderów i dekodek, część nadmiarowa umieszczana jest na najmniej znaczących pozycjach słowa kodowego. Poniżej pokazano przykładowe słowo kodowe binarnego kodu blokowego $(8, 3)$ z częścią informacyjną umieszczoną na najbardziej znaczących pozycjach.

Część informacyjna
 $\underbrace{\hspace{1.5cm}}$
10100111
 $\underbrace{\hspace{1.5cm}}$
 Część nadmiarowa

Jeżeli w każdym słowie kodowym danego kodu część informacyjna jest identyczna z nadawaną informacją, to taki kod nazywamy **kodem systematycznym**. Dalsza część niniejszego opracowania będzie dotyczyła systematycznych, liniowych i cyklicznych kodów binarnych z częścią informacyjną umieszczoną na najbardziej znaczących pozycjach.

Liczba słów kodowych danego kodu jest uzależniona od długości słów informacyjnych i jest równa liczbie możliwych słów informacyjnych, ponieważ każdemu słowu informacyjnemu musi odpowiadać dokładnie jedno słowo kodowe, oraz każdemu słowu kodowemu musi odpowiadać dokładnie jedno słowo informacyjne. Dla kodów binarnych liczbę słów informacyjnych, a co za tym idzie liczbę słów kodowych, można obliczyć jako liczbę wszystkich możliwych rozmieszczeń zer i jedynek na k pozycjach, która jest równa 2^k . Oczywiście z uwagi na to, iż każde słowo kodowe ma długość równą n , istnieje $2^n - 2^k$ słów o długości n , które nie należą do kodu.

Poniżej przedstawiono przykład prostego kodu binarnego $(3, 2)$, w którym długość części informacyjnej wynosi 2, a długość części korekcyjnej wynosi 1. Wartość bitu części korekcyjnej jest obliczana poprzez dodanie bitów części informacyjnej (nadawanej informacji).

Przykład 6

Przykład binarnego, systematycznego kodu blokowego (3, 2) z bitem parzystości.

Zbiór wszystkich możliwych słów informacyjnych: {00, 01, 10, 11}

Zbiór wszystkich możliwych słów o długości równej 3: {000, 001, 010, 011, 100, 101, 110, 111}

Przypisujemy do każdego słowa informacyjnego takie słowo 3-bitowe żeby wartość bitu w części nadmiarowej stanowiła sumę bitów z części informacyjnej:

Część informacyjna (słowo informacyjne)	Suma bitów części informacyjnej	Słowo kodowe
00	0	000
01	1	011
10	1	101
11	0	110

Kod stanowi zbiór słów: {000, 011, 101, 110}

Zbiór słów niekodowych stanowią słowa: {001, 010, 100, 111}

Jak widać na powyższym przykładzie, odebranie przez dekodery któregośkolwiek słowa ze zbioru słów kodowych {000, 011, 101, 110} pozwala na jednoznaczne zdekodowanie nadawanej informacji, która jest równa części informacyjnej odebranego słowa kodowego, czyli jego dwóm najbardziej znaczącym bitom. Jeżeli w torze transmisyjnym nastąpi przekłamanie jednego bitu w nadanym słowie kodowym, to w każdym przypadku (niezależnie od położenia przekłamanego bitu) dekodery odbierze słowo niekodowe i zasygnalizuje błąd.

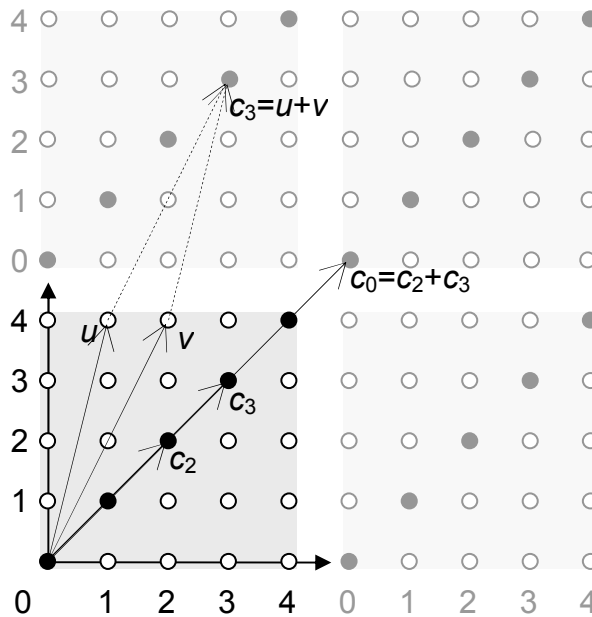
Oczywiście binarny kod blokowy (3, 2) można skonstruować wybierając dowolne cztery słowa ze zbioru słów 3-bitowych. Zakładając, że kod ma być systematyczny można wybrać słowa {001, 011, 101, 111}. W takim przypadku sprawdzenie poprawności transmisji po stronie odbiorczej polegałoby na sprawdzeniu wartości bitu części nadmiarowej, która zawsze powinna wynosić 1. Niestety przy takiej konstrukcji kodu mogą wystąpić przypadki, kiedy w wyniku przekłamania jednego bitu w nadanym słowie kodowym otrzymamy inne słowo kodowe i dekodery nie będzie w stanie wykryć błędu. Na przykład przekłamanie środkowego bitu w pierwszym słowie kodowym [001] daje drugie słowo kodowe [011].

3.1 Kody liniowe — właściwość liniowości

Jak pokazano w poprzednim przykładzie, jakość kodu zależy od wyboru słów kodowych. W celu uproszczenia metod tworzenia kodów oraz badania ich właściwości jakościowych stosuje się algebrę ciał skończonych.

W ujęciu algebraicznym, n -bitowe słowa mogą być traktowane jako punkty lub wektory n -wymiarowej liniowej przestrzeni wektorowej, która jest odpowiednikiem wspomnianego wcześniej zbioru wszystkich możliwych słów o długości n . Każda współrzędna punktu lub wektora odpowiada jednemu bitowi n -bitowego słowa. W takim wypadku, kod liniowy (n, k) jest reprezentowany przez k -wymiarową liniową podprzestrzeń wektorową¹ wspomnianej wcześniej przestrzeni n -wymiarowej. Podprzestrzeń ta tworzy k -wymiarową hiperpłaszczyznę, która przechodzi przez wektor (punkt) zerowy przestrzeni. Dla $k = 1$ hiperpłaszczyzna jest prostą, dla $k = 2$ jest płaszczyzną, a dla $k > 2$ każdy łatwo może sobie ją wyobrazić ☺

Na poniższym rysunku przedstawiono przykład kodu liniowego $(2, 1)$ nad ciałem prostym $\text{GF}(5)$ ², w którym dodawanie elementów ciała realizowane jest modulo 5. Białe kółka z obwódką w kolorze czarnym oraz kółka czarne reprezentują punkty dyskretnej, skończonej płaszczyzny, która odpowiada zbiorowi wszystkich możliwych słów o długości równej 2. Kółka czarne reprezentują słowa kodowe, a kółka w kolorze szarym i z szarą obwódką oznaczają kopie płaszczyzny ułożone w taki sposób, żeby zobrazować dodawanie modulo 5.



Rys. 2 Ilustracja kodu liniowego $(2, 1)$ nad $\text{GF}(5)$

Jak widać na rysunku, słowa kodowe tworzą prostą przechodzącą przez początek układu współrzędnych, która w sensie algebraicznym jest podprzestrzenią liniową 2-wymiarowej przestrzeni nad $\text{GF}(5)$. W rozpatrywanym kodzie część nadmiarowa każdego słowa kodowego jest tworzona poprzez powtórzenie jego części informacyjnej np. wektor

¹ Podprzestrzeń liniowa L_1 przestrzeni wektorowej L nad ciałem C jest zbiorem zamkniętym ze względu na dodawanie, odejmowanie i mnożenie przez skalar, czyli dla każdego λ należącego do C oraz każdych a i b należących do L_1 wektory $c = a + b$, $d = a - b$, oraz $e = \lambda \cdot a$ również należą do L_1 .

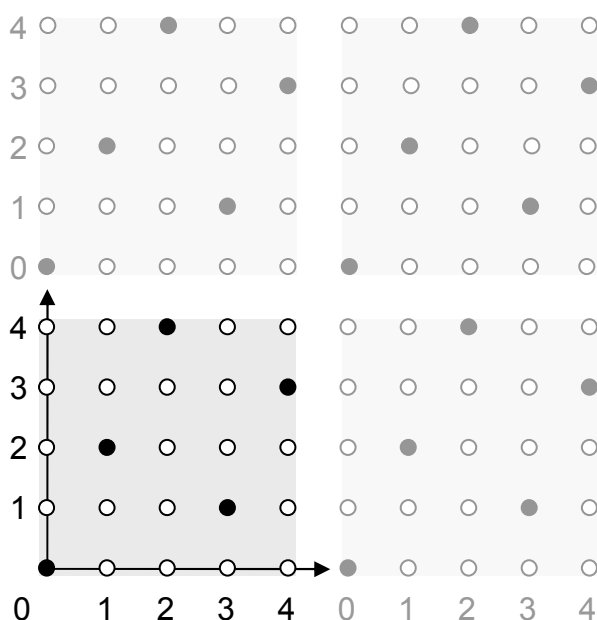
² Ciało proste $\text{GF}(5)$ zostało użyte tylko do zilustrowania kodów i nie będzie przedmiotem dalszych rozważań

kodowy $c_2 = [2, 2]$ oraz wektor kodowy $c_3 = [3, 3]$. Suma wektorów kodowych c_2 i c_3 daje wektor kodowy $c_0 = [0, 0]$ ponieważ w $GF(5)$ suma $2 + 3 = 0$. Można również zauważyć, że suma dwóch wektorów niekodowych u i v może dać w wyniku wektor kodowy. Wynika to z faktu, że wektory kodowe są elementami nie tylko kodu, ale również przestrzeni, do której ten kod należy. Dodatkowo widać, że suma wektora kodowego i wektora niekodowego daje w wyniku wektor niekodowy.

Uogólniając powyższe spostrzeżenia można sformułować podstawowe właściwości, które spełniają wszystkie kody liniowe i cykliczne.

1. Suma dwóch dowolnych słów kodowych danego kodu liniowego daje w wyniku ciąg będący również słowem kodowym tego samego kodu (kryterium liniowości).
2. Suma dowolnego słowa kodowego i dowolnego słowa niekodowego daje w wyniku słowo nienależące do kodu.

Inny przykład kodu liniowego $(2, 1)$ nad ciałem prostym $GF(5)$ pokazano na rysunku 3. Wektory kodowe tego kodu to $[0, 0]$, $[1, 2]$, $[2, 4]$, $[3, 1]$ i $[4, 3]$ czyli ten kod jest również kodem systematycznym. Również w tym przypadku kod tworzy prostą przechodzącą przez początek układu współrzędnych.



Rys. 3 Przykład kodu liniowego $(2, 1)$ nad $GF(5)$

Przykład 7

Jeżeli słowa 1011 i 0101 są słowami kodowymi pewnego liniowego kodu binarnego to ich suma równa 1110 jest również słowem kodowym tego samego kodu.

$$\begin{array}{rcl}
 & 1011 & x^3 + x + 1 \\
 + & 0101 & + \frac{x^2 + 1}{x^3 + x^2 + x} \\
 = & 1110 & = x^3 + x^2 + x
 \end{array}$$

3.2 Kody cykliczne — właściwość cykliczności

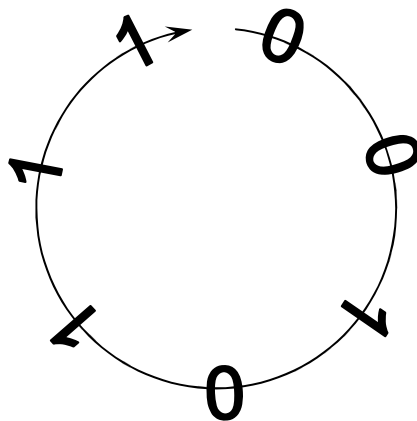
Najczęściej stosowaną grupą liniowych kodów blokowych są kody cykliczne. Ich szczególne właściwości pozwalają m.in. znacząco uprościć układy służące do korekcji błędów.

Wszystkie kody cykliczne spełniają **kryterium cykliczności**, czyli **cykliczne przesunięcie współrzędnych dowolnego wektora kodowego danego kodu cyklicznego daje w wyniku również wektor kodowy tego kodu**. Na przykład cykliczne przesunięcie (obróć) o jedno miejsce w lewo współrzędnych wektora kodowego c_1 daje wektor c_2 , który jest również wektorem należącym do tego samego kodu:

$$c_1 = [a_{n-1}, a_{n-2}, \dots, a_1, a_0]$$

$$c_2 = [a_{n-2}, a_{n-3}, \dots, a_0, a_{n-1}]$$

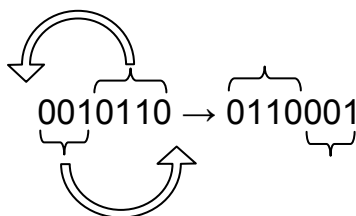
Właściwość cykliczności można zilustrować jako dyskretny okrąg złożony ze skończonej liczby punktów, która jest równa długości słowa kodowego (n). Każdy punkt okręgu reprezentuje jeden bit słowa kodowego. Po odczytaniu n bitów, rozpoczynając od dowolnego bitu otrzymamy n -bitowe słowo należące do danego kodu cyklicznego. Należy jednak pamiętać, aby zawsze czytać bity w jednym kierunku (np. zgodnie z ruchem wskazówek zegara).



Rys. 4 Ilustracja właściwości cykliczności

Na podstawie powyższego rysunku można odczytać następujące słowa: [0010111], [0101110], [1011100], [0111001], [1110010], [1100101], [1001011]. Każde z nich jest słowem kodowym tego samego kodu cyklicznego (7, 3).

Przy wykonywaniu cyklicznego przesunięcia bitów w zapisie binarnym należy pamiętać o długości słowa kodowego n . Nie wolno opuszczać najbardziej znaczących bitów o wartości równej 0! Poniżej zilustrowano przesunięcie cykliczne słowa binarnego o trzy pozycje w lewo.



W zapisie wielomianowym przesunięcie cykliczne polega na dodaniu (lub odjęciu, w zależności od kierunku przesunięcia) liczby przesuwanych miejsc do potęgi każdego składnika wielomianu, przy czym dodawanie jest realizowane modulo n (żadna potęga nie może przekroczyć $n - 1$ oraz nie może być ujemna). Na przykład dla kodu cyklicznego o długości równej 7 słowo kodowe c po przesunięciu cyklicznym o 3 pozycje w lewo daje słowo kodowe $c^{(+3)}$:

$$c = x^4 + x^2 + x \rightarrow c^{(+3)} = x^{(4+3) \bmod 7} + x^{(2+3) \bmod 7} + x^{(1+3) \bmod 7} \rightarrow c^{(+3)} = x^5 + x^4 + 1$$

Można zauważyć, że **przesunięcie cykliczne słowa kodowego o n pozycji w tą samą stronę nie zmienia jego postaci. Przesunięcie cykliczne słowa kodowego o i pozycji w jedną stronę jest równoważne przesunięciu cyklicznemu o $(n - i)$ pozycji w przeciwną stronę.**

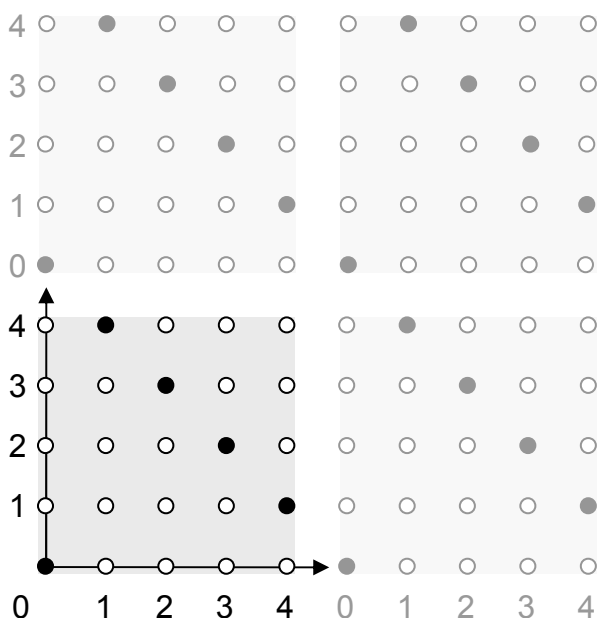
Przesunięcie cykliczne wielomianu o i pozycji w lewo można również zrealizować poprzez pomnożenie go przez x^i i obliczenie reszty z dzielenia tego iloczynu przez wielomian $(x^n + 1)$:

$$c^{(+i)} = (x^i \cdot c) \bmod (x^n + 1) \quad (4)$$

Łatwo to sprawdzić na poprzednim przykładzie (dzielenia wielomianów nie pokazano):

$$\begin{aligned} c &= x^4 + x^2 + x \\ x^3 \cdot c &= x^7 + x^5 + x^4 \\ (x^7 + x^5 + x^4) \bmod (x^7 + 1) &= x^5 + x^4 + 1 = c^{(+3)} \end{aligned}$$

Przykład kodu cyklicznego (2, 1) nad GF(5) pokazano na rysunku poniżej. Widać, że kod ten jest kodem liniowym, ponieważ tworzy prostą (1-wymiarową podprzestrzeń liniową) przechodzącą przez początek układu współrzędnych oraz spełnia kryterium liniowości. Wektorami kodowymi tego kodu są: $[0, 0]$, $[1, 4]$, $[2, 3]$, $[3, 2]$ oraz $[4, 1]$. Można zauważyć, że przesunięcie cykliczne (w tym przypadku zamiana) współrzędnych każdego wektora kodowego daje wektor również należący do tego kodu.



Rys. 5 Ilustracja kodu cyklicznego (2, 1) nad GF(5)

Pokazany powyżej kod jest jedynym kodem cyklicznym (2, 1) nad GF(5). Pozostałe pięć kodów liniowych (2, 1), które można skonstruować w tej przestrzeni nie spełniają kryterium cykliczności.

3.3 Odległość minimalna kodu

Bardzo ważnym parametrem charakteryzującym kod jest jego odległość minimalna określana jako najmniejsza odległość między słowami kodowymi tego kodu. Należy zaznaczyć, że **parametr ten charakteryzuje kod** rozumiany jako zbiór wszystkich słów kodowych, a nie dowolne słowa binarne lub kodowe. Odległość minimalną oznacza się najczęściej d , a **do jej wyznaczenia należy znać wszystkie słowa kodowe**. Sposób obliczenia odległości minimalnej polega na obliczeniu odległości dla wszystkich możliwych par słów kodowych i następnie wybraniu wartości najmniejszej. Niestety jest to bardzo pracochłonna metoda z uwagi na dużą liczbę operacji O_1 . Dla blokowych kodów binarnych o parametrach (n, k) liczbę możliwych par słów kodowych można wyrazić następująco:

$$O_1 = \binom{2^k}{2} = \frac{2^k!}{(2^k-2)! \cdot 2!} = \frac{(2^k-2)! \cdot (2^k-1) \cdot 2^k}{(2^k-2)! \cdot 2} = (2^k-1) \cdot 2^{k-1} \approx 2^{2k-1}$$

Na przykład, w przypadku kodu o $k=8$ należałoby wykonać 32640 operacji dodawania modulo 2 oraz tyle samo operacji wyznaczenia wagi Hamminga otrzymanych sum (zgodnie z zależnością (3)). Dodatkowo ze zbioru 32640 odległości należałoby wybrać wartość najmniejszą, co wiąże się z wykonaniem 32640 porównań.

Wyznaczenie odległości minimalnej dla blokowych kodów liniowych i cyklicznych można znacząco uprościć wykorzystując właściwość liniowości, którą w tych kodach spełniają wszystkie słowa kodowe. Wyznaczenie odległości między dwoma słowami kodowymi wiąże się z wyznaczeniem wagi Hamminga ich sumy, a z kryterium liniowości wiadomo, że suma dwóch słów kodowych jest również słowem kodowym. W takim wypadku, dysponując wszystkimi słowami kodowymi kodu liniowego lub cyklicznego, z góry znamy wszystkie możliwe wyniki sumowania słów kodowych, więc wystarczy obliczyć wagi Hamminga wszystkich słów kodowych oprócz słowa zerowego¹. Najmniejsza z nich będzie równa odległości minimalnej.

Porównując liczbę operacji $O_2 = 2^k - 1$ takiego podejścia z poprzednim przykładem widać, że liczba operacji ogranicza się do $2^8 - 1 = 255$ obliczeń wag Hamminga, a następnie wybranie ze zbioru 255 odległości, wartości najmniejszej.

Przykład 8

Poniżej podano wszystkie słowa cyklicznego kodu binarnego (8, 3). Wykorzystując właściwość liniowości obliczyć odległość minimalną tego kodu.

$c_0 = 00000000$	
$c_1 = 00110011$	$w_H(c_1) = 4$
$c_2 = 01010101$	$w_H(c_2) = 4$
$c_3 = 01100110$	$w_H(c_3) = 4$
$c_4 = 10011001$	$w_H(c_4) = 4$
$c_5 = 10101010$	$w_H(c_5) = 4$
$c_6 = 11001100$	$w_H(c_6) = 4$
$c_7 = 11111111$	$w_H(c_7) = 8$

$$d = \min\{w_H(c_1), w_H(c_2), w_H(c_3), w_H(c_4), w_H(c_5), w_H(c_6), w_H(c_7)\} = 4$$

Jak widać, minimalna waga Hamminga wszystkich niezerowych słów kodowych wynosi 4. W takim razie odległość minimalna dla tego kodu wynosi 4.

¹ Nieuwzględnienie słowa zerowego wiąże się z tym, że wynikiem sumowania dwóch **różnych** słów kodowych nigdy nie będzie słowo zerowe, więc nie jest ono uwzględniane w obliczeniu odległości minimalnej.

Podsumowując, na podstawie znajomości odległości minimalnej kodu można określić podstawowe właściwości jego słów kodowych. Jeżeli odległość minimalna blokowego kodu liniowego wynosi d to można napisać, że:

- waga każdego niezerowego słowa kodu wynosi co najmniej d , inaczej: nie istnieje niezerowe słowo kodowe, które ma mniej niż d bitów o wartości równej 1

$$w_H(c_{\neq 0}) \geq d$$

- odległość między dwoma dowolnymi słowami kodowymi wynosi co najmniej d , inaczej: słowa kodowe tego kodu różnią się między sobą wartością co najmniej d bitów na odpowiadających sobie pozycjach

$$d_H(c_i, c_j) \geq d$$

3.4 Zdolność detekcyjna i korekcyjna

Zdolność detekcyjna l kodu określa liczbę błędów (przekłamanych bitów) w dowolnym słowie kodowym, które zostaną wykryte z prawdopodobieństwem równym 100% niezależnie od rozkładu (wzoru) błędów. Oczywiście dla niektórych słów kodowych i niektórych rozkładów (rozmieszczeń) błędów, dekodery jest w stanie wykryć błąd przy wystąpieniu większej liczby przekłamań niż wynika to ze zdolności detekcyjnej kodu, ponieważ wykrycie błędu polega na sprawdzeniu czy ciąg odebrany jest słowem kodowym czy nie jest. Sytuacja, gdy przekłamanie spowoduje powstanie innego słowa kodowego wystąpi w przypadku, gdy wektor błędu ma postać dowolnego słowa kodowego danego kodu (właściwość liniowości). Należy jednak zauważyć, że **zdolność detekcyjna gwarantuje wykrycie l lub mniej przekłamań niezależnie od ich rozłożenia.**

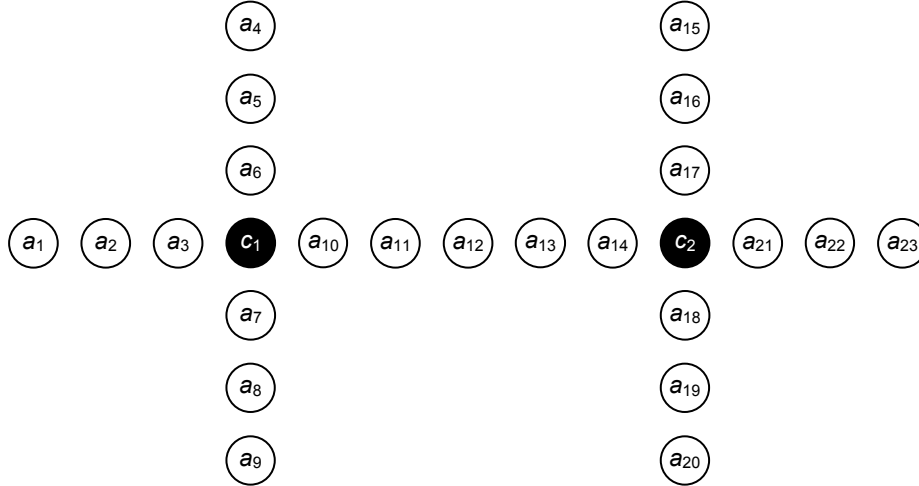
Do obliczenia zdolności detekcyjnej można wykorzystać odległość minimalną kodu. Jak już wspomniano, parametr ten mówi o tym, że każde słowo kodowe różni się co najmniej o d bitów od innego dowolnego słowa kodowego, czyli każde słowo kodowe ma wagę równą co najmniej d . W takim razie nie ma możliwości wygenerowania słowa kodowego poprzez dodanie dowolnego ciągu o wadze mniejszej od d do któregośkolwiek słowa kodowego. Inaczej mówiąc, suma słowa kodowego c i dowolnego ciągu błędów e o wadze mniejszej od d , będzie leżała w odległości mniejszej od d od słowa kodowego c , czyli nie będzie słowem kodowym (bo wszystkie leżą w odległości co najmniej d). Można więc zapisać, że zdolność detekcyjna kodu l jest równa:

$$l = d - 1$$

czyli kod jest w stanie wykryć $d - 1$ błędów (przekłamanych bitów) niezależnie od ich rozkładu.

Innym ważnym parametrem kodu jest jego **zdolność korekcyjna**, która **określa liczbę błędów (przekłamanych bitów) w dowolnym słowie kodowym, które kod może skorygować z prawdopodobieństwem równym 100% niezależnie od rozkładu (wzoru) błędów.** W procesie dekodowania odebranych ciągów, dekodery sprawdza czy słowo odebrane jest słowem kodowym. Jeżeli ciąg odebrany nie jest słowem kodowym, to należy znaleźć słowo kodowe leżące najbliżej ciągu odebranego (różniące się najmniejszą liczbą bitów) i przyjąć, że ciąg nadany był właśnie tym słowem kodowym (jeżeli przekłamań było dużo to nie musi to być prawdą). Oczywiście może zdarzyć się, że istnieje więcej niż jedno słowo kodowe leżące w takiej samej (najmniejszej) odległości od ciągu odebranego, wtedy korekcja błędów nie jest możliwa.

W celu obliczenia zdolności korekcyjnej kodu należy określić największą wagę wektora błędu e , który dodany do dowolnego słowa kodowego c da w wyniku słowo, którego najbliższym słowem kodowym będzie tylko słowo c . Można to zilustrować na przykładzie poniższego rysunku. Wszystkie kółka reprezentują przykładowe słowa ze zbioru 2^n n -bitowych słów binarnych, a wypełnione kółka reprezentują dwa z 2^k słów kodowych należących do pewnego kodu (n, k) , które leżą w odległości 6 od siebie. Dodatkowo przyjmujemy, że odległość minimalna tego kodu wynosi 6, a wszystkie kółka są rozmieszczone z zachowaniem odległości Hamminga między słowami.



Rys. 6 Ilustracja zdolności korekcyjnej

Można zauważyć, że przekłamanie jednego bitu w słowie kodowym c_1 da w rezultacie jedno ze słów niekodowych a_3 , a_6 , a_7 lub a_{10} . Oczywiście słowa te leżą w odległości 1 od słowa kodowego c_1 . Słowo a_{10} leży w odległości 5 od słowa c_2 , słowa a_6 i a_7 w odległości 6 od słowa c_2 , a słowo a_3 leży w odległości 7 od słowa c_2 . Jak widać, dla każdego ze słów niekodowych a_3 , a_6 , a_7 i a_{10} najbliższym słowem kodowym jest słowo c_1 , więc w przypadku odebrania któregośkolwiek z nich, podczas dekodowania można przyjąć, że nadanym słowem kodowym było słowo c_1 .

Podobnie wygląda sytuacja w przypadku przekłamania dwóch bitów w słowie c_1 , jednak w przypadku przekłamania trzech bitów, może powstać słowo niekodowe a_{12} , które leży w odległości 3 zarówno od słowa c_1 jak i c_2 , więc w wypadku odebrania słowa a_{12} dekodery nie może podjąć jednoznacznej decyzji, które słowo kodowe przyjąć za prawidłowe i wtedy korekcja nie jest możliwa. W takim razie, największa waga wektora błędu, która zapewni korekcję wynosi 2. Można zauważyć, że gdyby słowa c_1 i c_2 leżały w odległości 5 od siebie to największa waga wektora błędu, która zapewni korekcję wyniosłaby również 2. Zatem zdolność korekcyjną kodu t można wyrazić następującą zależnością:

$$t = \text{int}\left(\frac{d-1}{2}\right)$$

gdzie $\text{int}(z)$ oznacza część całkowitą liczby z .

4 Test nr 1 — zadania

Fragmenty zaznaczone kolorem czerwonym zostały podane jako dane przykładowe — na testach będą zastąpione innymi danymi.

1. Obliczyć wynik i resztę z dzielenia wielomianu $u = [0111001010]$ przez wielomian $v = [011100]$. Działania wykonać w zapisie **wielomianowym**.

Wynik i resztę należy podać w zapisie wielomianowym. Jeżeli zarówno wynik jak i reszta będą poprawne, zostaną przyznane 2 pkt, w przeciwnym wypadku 0 pkt.

2. Obliczyć wynik i resztę z dzielenia wielomianu $u(x) = x^8 + x^5 + x^3 + x$ przez wielomian $v(x) = x^2 + x + 1$. Działania wykonać w zapisie **bitowym**.

Wynik i resztę należy podać w zapisie bitowym. Jeżeli zarówno wynik jak i reszta będą poprawne, zostaną przyznane 2 pkt, w przeciwnym wypadku 0 pkt.

3. Dane są dwa wielomiany $u(x) = x^8 + x^5 + x^3 + x$, $v(x) = x^5 + x^2 + x + 1$.
Podać odpowiedzi na poniższe pytania (wyniki podać bez wykonywania dzielenia):
 - a) Jaki stopień będzie miał wielomian reprezentujący wynik dzielenia $u(x)$ przez $v(x)$?
 - b) Jaki stopień będzie miał wielomian reprezentujący resztę z dzielenia $u(x)$ przez $v(x)$?
 - c) Jaki stopień będzie miał wielomian reprezentujący wynik mnożenia $u(x) \cdot v(x)$?
 - d) Jaki stopień będzie miał wielomian reprezentujący wynik dodawania $u(x) + v(x)$?
 - e) Jaki stopień będzie miał wielomian reprezentujący wynik dodawania $u(x)$ i innego wielomianu tego samego stopnia?
 - f) Jaki stopień będzie miał wielomian reprezentujący wynik dodawania $v(x)$ i innego wielomianu tego samego stopnia?
 - g) Ile istnieje wielomianów o stopniu mniejszym od stopnia wielomianu $u(x)$?
 - h) Ile istnieje wielomianów o stopniu nieprzekraczającym stopnia wielomianu $v(x)$?

Za każdy poprawny wynik zostanie przyznane $\frac{1}{2}$ pkt, w przeciwnym wypadku 0 pkt.

4. Określ czy następujące zdania są prawdziwe czy fałszywe.

Pojęcie wagi Hamminga dotyczy

- dowolnego słowa binarnego
- słowa binarnego o długości równej długości słowa kodowego
- słowa binarnego o długości równej długości słowa informacyjnego
- dowolnego słowa kodowego
- dowolnego słowa informacyjnego
- tylko słowa kodowego
- tylko słowa informacyjnego
- tylko części informacyjnej słowa kodowego
- tylko części nadmiarowej słowa kodowego
- części informacyjnej słowa kodowego
- części nadmiarowej słowa kodowego
- pary dowolnych słów binarnych
- pary dowolnych słów binarnych o jednakowej długości
- pary dowolnych słów kodowych
- pary dowolnych słów kodowych o jednakowej długości
- pary dowolnych słów kodowych tego samego kodu
- pary dowolnych niezerowych słów kodowych tego samego kodu
- tylko takiej pary słów, z których co najmniej jedno jest słowem kodowym
- całego kodu
- zbioru słów kodowych danego kodu z wyłączeniem słowa zerowego

W pytaniu należy zaznaczyć czy zdanie jest prawdziwe czy fałszywe. Na teście będą cztery losowo wybrane możliwości z wymienionych powyżej. Za każdą prawidłowo zaznaczoną opcję będzie dodane $\frac{1}{2}$ pkt., a za każdą nieprawidłowo zaznaczoną opcję lub brak zaznaczenia będzie odjęte $\frac{1}{2}$ pkt. W przypadku uzyskania za pytanie punktów ujemnych nie będą one uwzględniane w ogólnej punktacji z testu.

5. Określ czy następujące zdania są prawdziwe czy fałszywe.
Pojęcie odległości Hamminga dotyczy

Możliwości i punktacja jak pytaniu 4.

6. Dane są dwa słowa binarne w postaci wielomianowej $u(x) = x^8 + x^5 + x^3 + x$,
 $v(x) = x^5 + x^2 + x + 1$. Oblicz odległość Hamminga między nimi.

Za poprawny wynik zostanie przyznany 1 pkt, w przeciwnym wypadku 0 pkt.

7. Określ czy następujące zdania są prawdziwe czy fałszywe.
Pojęcie odległości minimalnej dotyczy

Możliwości i punktacja jak pytaniu 4.

8. Odległość minimalna pewnego binarnego kodu liniowego wynosi 8.
Określ czy następujące zdania są prawdziwe czy fałszywe.

- W przypadku wystąpienia 4 przekłamań dekodery na pewno wykryje błąd.
- W przypadku wystąpienia 8 przekłamań dekodery może wykryć błąd.
- W przypadku wystąpienia 5 przekłamań dekodery może nie wykryć błędu.
- W przypadku wystąpienia 4 przekłamań dekodery na pewno skoryguje błąd.
- W przypadku wystąpienia 4 przekłamań dekodery może skorygować błąd.
- W przypadku wystąpienia 2 przekłamań dekodery może nie skorygować błędu.

W pytaniu należy zaznaczyć czy zdanie jest prawdziwe czy fałszywe. Za każdą prawidłowo zaznaczoną opcję będzie dodane ½ pkt., a za każdą nieprawidłowo zaznaczoną opcję lub brak zaznaczenia będzie odjęte ½ pkt. W przypadku uzyskania za pytanie punktów ujemnych nie będą one uwzględniane w ogólnej punktacji z testu.

9. Ile słów niekodowych, mających przekłamania wyłącznie w części nadmiarowej, może leżeć w odległości mniejszej niż 4 i większej niż 1 od słowa kodowego kodu blokowego (10, 4).

Rozwiązanie:

Z treści zadania wynika, iż należy uwzględnić tylko takie słowa, w których 4 najbardziej znaczące bity (część informacyjna) będą takie same jak w rozpatrywanym słowie kodowym. W takim razie obliczenia można ograniczyć do samej części nadmiarowej, czyli pozostałych 6 bitów.

Rozpatrywane odległości uwzględnianych słów niekodowych od rozpatrywanego słowa kodowego zawierają się w przedziale obustronnie otwartym (1, 4), czyli będą to odległości 2 i 3.

Można zatem napisać:

$$M_2 = \binom{6}{2} = \frac{6!}{(6-2)! \cdot 2!} = 15$$

$$M_3 = \binom{6}{3} = \frac{6!}{(6-3)! \cdot 3!} = 20$$

W odległości mniejszej niż 4 i większej niż 1 od słowa kodowego kodu blokowego (10, 4) może leżeć 35 słów.

Za poprawny wynik zostaną przyznane 2 pkt, w przeciwnym wypadku 0 pkt.

10. Dany jest pewien systematyczny, binarny kod liniowy (15, 4), w którym część informacyjna jest reprezentowana przez najbardziej znaczące bity słów kodowych. Podaj odpowiedzi na poniższe pytania (wszystkie pytania dotyczą podanego wyżej kodu):

- Ile różnych informacji można przekazać za pomocą tego kodu?
- Z ilu słów kodowych składa się ten kod?
- Jaka jest długość słów kodowych?
- Jaka jest długość ciągów informacyjnych?
- Ile istnieje słów o długości równej długości słów kodowych?
- Ile różnych słów zawierających przekłamanie może dotrzeć do dekodera?
- Jaki może być największy stopień wielomianu reprezentującego ciąg informacyjny?
- Jaki może być najmniejszy stopień wielomianu reprezentującego niezerowy ciąg informacyjny?
- Jaki może być największy stopień wielomianu reprezentującego słowo kodowe?
- Jaki może być najmniejszy stopień wielomianu reprezentującego niezerowe słowo kodowe?

Za każdy poprawny wynik zostanie przyznane $\frac{1}{2}$ pkt, w przeciwnym wypadku 0 pkt.

11. Zdolność korekcyjna pewnego kodu wynosi 3. Ile może wynosić jego zdolność detekcyjna oraz odległość minimalna (podaj wszystkie możliwości).

Za kompletny i poprawny wynik zostaną przyznane 2 pkt, w przeciwnym wypadku 0 pkt.

12. Poniżej podano 5 słów o długości 12. Pierwsze słowo należy do kodu cyklicznego (12, 4), a w czterech pozostałych wystąpiły błędy w części nadmiarowej. Wykorzystując właściwości liniowości i cykliczności zaznaczyć przekłamane bity.

$$\begin{aligned}c &= 111000111000 \\u_1 &= 011011011010 \\u_2 &= 100100101100 \\u_3 &= 010101010001 \\u_4 &= 000111001101\end{aligned}$$

Rozwiązanie:

Wykorzystując własności liniowości i cykliczności można napisać (pogrubioną czcionką wyróżniono części informacyjne, a podkreśleniem zaznaczono przekłamania):

$$\begin{aligned}u_1 &\neq c_1 + c_1^{(+2)} && \begin{array}{r} 111000111000 \\ 100011100011 \\ \hline 011011011010 \end{array} \\u_2 &\neq c_1 + c_1^{(-1)} && \begin{array}{r} 111000111000 \\ 011100011100 \\ \hline 100100101100 \end{array} \\u_3 &\neq c_1 + c_1^{(+1)} + c_1^{(-1)} && \begin{array}{r} 111000111000 \\ 110001110001 \\ 011100011100 \\ \hline 010101010001 \end{array} \\u_4 &\neq c_1^{(+3)} && \begin{array}{r} 000111001101 \end{array}\end{aligned}$$

Za wszystkie poprawnie zaznaczone przekłamania w jednym słowie zostanie przyznane 1,5 pkt, w przeciwnym wypadku 0 pkt.

13. Dane jest słowo kodowe kodu cyklicznego (12, 3) [011001100110]. Obliczyć odległość minimalną tego kodu.

Rozwiązanie:

Ponieważ $k = 3$ to kod składa się z $2^3 = 8$ słów kodowych, przy czym jest 7 słów niezerowych. W takim razie, wykorzystując własności liniowości i cykliczności, można napisać:

$c_{011} = 011001100110$	dane	$w_H(c_{011}) = 6$
$c_{110} = 110011001100$	$= c_{011}^{(+1)}$	$w_H(c_{110}) = 6$
$c_{100} = 100110011001$	$= c_{110}^{(+1)}$	$w_H(c_{100}) = 6$
$c_{001} = 001100110011$	$= c_{100}^{(+1)}$	$w_H(c_{001}) = 6$
$c_{101} = 101010101010$	$= c_{100} + c_{001}$	$w_H(c_{101}) = 6$
$c_{010} = 010101010101$	$= c_{101}^{(+1)}$	$w_H(c_{010}) = 6$
$c_{111} = 111111111111$	$= c_{101} + c_{010}$	$w_H(c_{111}) = 12$

Najmniejsza waga Hamminga z wszystkich niezerowych słów kodu wynosi 6, czyli odległość minimalna kodu $d = 6$.

W teście należy podać następujące parametry:

- liczba słów kodowych
- maksymalna waga Hamminga słów kodowych
- odległość minimalna kodu

Jeżeli odpowiedź a) jest niepoprawna to za zadanie zostanie przyznane 0 pkt., w przeciwnym wypadku, jeżeli odpowiedź b) jest niepoprawna to za zadanie zostanie przyznane ½ pkt., w przeciwnym wypadku, jeżeli odpowiedź c) jest niepoprawna to za zadanie zostaną przyznane 2 pkt., w przeciwnym wypadku za zadanie zostanie przyznane 7 pkt.

Tabela 3 Punktacja zadań testu nr 1

Numer zadania	Maksymalna liczba punktów
1	2
2	2
3	4
4	2
5	2
6	1
7	2
8	3
9	2
10	5
11	2
12	6
13	7
Suma	40

5 Kodowanie informacji

W procesie kodowania informacji za pomocą kodów blokowych można skorzystać z wielu metod. Jedną z nich jest wykorzystanie księgi kodowej, czyli tablicy wszystkich możliwych ciągów informacyjnych i odpowiadających im słów kodowych. Metoda ta jednak, w przypadku implementacji praktycznej, wymaga użycia pamięci, co znacząco zwiększa koszt urządzenia kodującego. Dodatkowo, dekodowanie korekcyjne taką metodą wymaga stosunkowo dużej mocy obliczeniowej.

Innym sposobem kodowania informacji jest metoda macierzowa, która może być stosowana dla wszystkich kodów liniowych. Metoda ta wykorzystuje właściwość liniowości kodu i polega na mnożeniu ciągu informacyjnego przez macierz generacyjną kodu. Sposób ten może być stosunkowo prosto zaimplementowany w formie układów kombinacyjnych, jednak w przypadku potrzeby korekcji błędów po stronie odbiorczej istnieje potrzeba użycia arytmetycznej jednostki obliczeniowej oraz pamięci w urządzeniu dekodującym.

Duże uproszczenie konstrukcji koderów można uzyskać wykorzystując operacje na wielomianach. Kodery takie można zrealizować za pomocą rejestrów przesuwnych jako proste układy sekwencyjne bez konieczności użycia układów pamięci.

5.1 Kodowanie za pomocą wielomianu

Rozważmy pewien kod blokowy o parametrach (n, k) . Najprostszym sposobem zakodowania informacji za pomocą algebry wielomianów jest pomnożenie pewnego wielomianu $g(x)$, zwanego **wielomianem generującym kod**, przez wielomian reprezentujący informację $m(x)$. Wielomian kodowy $c(x)$ odpowiadający¹ wielomianowi informacyjnemu $m(x)$ można obliczyć z zależności:

$$c(x) = m(x) \cdot g(x) \quad (5)$$

gdzie:

- $c(x)$ – wielomian kodowy (stopnia nie większego niż $n - 1$),
- $m(x)$ – wielomian informacyjny (stopnia nie większego niż $k - 1$),
- $g(x)$ – wielomian generujący kod (stopnia równego $n - k$).

Sprawdźmy czy taka metoda pozwala uzyskać zbiór słów kodowych spełniający warunki opisane w poprzednich rozdziałach:

1. **Stopień wielomianu kodowego $c(x)$ dla kodu (n, k) nie może być większy od $n - 1$, co oczywiście wynika z długości słowa kodowego tego kodu równej n .**

Jak widać z powyższej zależności, stopień wielomianu kodowego $c(x)$ będzie równy sumie stopnia wielomianu generującego $g(x)$ i wielomianu informacyjnego $m(x)$, czyli wyniesie maksymalnie $(k - 1) + (n - k) = n - 1$.

2. **Każdemu wielomianowi informacyjnemu musi odpowiadać dokładnie jeden wielomian kodowy oraz każdemu wielomianowi kodowemu musi odpowiadać dokładnie jeden wielomian informacyjny.**

Jak widać powyżej, każdy wielomian kodowy będzie iloczynem wielomianu generującego i wielomianu informacyjnego. W takim razie dla dwóch różnych wielomianów informacyjnych otrzymamy dwa różne wielomiany kodowe. Dla 2^k wielomianów informacyjnych otrzymamy 2^k wielomianów kodowych.

¹ „Odpowiadający” w sensie przypisania jednego elementu m ze zbioru słów informacyjnych do jednego elementu c ze zbioru słów kodowych

3. Odróżnienie wielomianów kodowych od wielomianów niekodowych, czyli wykrycie błędów, powinno być jednoznaczne.

Przy takiej metodzie kodowania, kod stanowią **wszystkie możliwe**¹ wielokrotności wielomianu generującego, których stopień nie przekracza $n - 1$, więc wszystkie pozostałe wielomiany stopnia nie większego niż $n - 1$ nie będą wielokrotnościami wielomianu generującego. W takim razie odróżnienie wielomianów kodowych od wielomianów niekodowych polega na podzieleniu wielomianu odebranego przez wielomian generujący i sprawdzeniu reszty z tego dzielenia. Jeżeli reszta jest równa 0, to znaczy, że odebrany wielomian jest wielokrotnością $g(x)$ (dzieli się przez $g(x)$ bez reszty), czyli jest wielomianem kodowym.

Należy zauważyć, że taka metoda kodowania daje w wyniku kod liniowy. Można to sprawdzić w następujący sposób. Rozważmy dwa wielomiany kodowe $c_1(x)$ i $c_2(x)$. Należy sprawdzić, czy wielomian $c_3(x)$ będący sumą $c_1(x)$ i $c_2(x)$ będzie wielomianem kodowym, czyli czy będzie wielokrotnością $g(x)$.

$$c_3(x) = c_1(x) + c_2(x)$$

Na podstawie (5) można napisać:

$$c_3(x) = m_1(x) \cdot g(x) + m_2(x) \cdot g(x)$$

więc:

$$c_3(x) = (m_1(x) + m_2(x)) \cdot g(x)$$

Ponieważ zarówno stopień wielomianu informacyjnego $m_1(x)$ jak i $m_2(x)$ nie przekracza $k - 1$, to stopień wielomianu $(m_1(x) + m_2(x))$ również nie przekroczy $k - 1$, czyli $(m_1(x) + m_2(x))$ będzie prawidłowym wielomianem informacyjnym. W takim razie, zgodnie z tym co napisano wcześniej, wynik mnożenia wielomianu informacyjnego przez wielomian generujący będzie wielomianem stopnia co najwyżej $n - 1$ oraz będzie wielokrotnością wielomianu generującego czyli będzie poprawnym wielomianem kodowym. Jak więc widać, suma słów kodowych takiego kodu będzie również słowem kodowym tego kodu (kryterium liniowości), co jest warunkiem koniecznym i wystarczającym aby uznać taki kod za kod liniowy.

Sprawdźmy teraz czy można w taki sposób generować kod cykliczny. Z kryterium cykliczności wiadomo, że wielomian $c^{(+i)}(x)$ wynikający z przesunięcia cyklicznego wielomianu kodowego $c(x)$ powinien być również wielomianem kodowym tego samego kodu, do którego należy $c(x)$.

Na podstawie (4) można napisać:

$$c^{(+i)}(x) = (x^i \cdot c(x)) \bmod (x^n + 1)$$

⇓

$$c^{(+i)}(x) = x^i \cdot c(x) + q(x) \cdot (x^n + 1)$$

$ \begin{aligned} r &= y \bmod x \\ &\Downarrow \\ y &= n \cdot x + r \\ &\Downarrow \\ r &= y - n \cdot x \end{aligned} $
--

Korzystając z (5) rozpisujemy wielomian $c(x)$. Dodatkowo zakładamy, że $g(x)$ jest czynnikiem wielomianu $(x^n + 1)$, czyli wielomian $(x^n + 1)$ dzieli się bez reszty przez wielomian $g(x)$:

$$c^{(+i)}(x) = x^i \cdot m(x) \cdot g(x) + q(x) \cdot p(x) \cdot g(x)$$

⇓

$$c^{(+i)}(x) = (x^i \cdot m(x) + q(x) \cdot p(x)) \cdot g(x)$$

gdzie $q(x)$ i $p(x)$ reprezentują pewne wielomiany.

¹ Wynika to z faktu, iż tworząc zbiór wszystkich wielomianów kodowych mnożymy wielomian generujący przez **wszystkie możliwe** wielomiany stopnia nieprzekraczającego $k - 1$, a tym samym otrzymujemy zbiór wszystkich możliwych wielokrotności wielomianu generującego stopnia nieprzekraczającego $n - 1$.

Jak widać, przy założeniu że wielomian $g(x)$ jest czynnikiem wielomianu (x^n+1) , wielomian wynikający z przesunięcia cyklicznego $c^{(+i)}(x)$ jest wielokrotnością wielomianu generującego $g(x)$, co jest warunkiem koniecznym do tego, aby wielomian $c^{(+i)}(x)$ był wielomianem kodowym tego samego kodu co $c(x)$. Dodatkowo, z zależności (4) widać, że wielomian $c^{(+i)}(x)$ jest resztą z dzielenia pewnego wielomianu przez wielomian (x^n+1) , a reszta z dzielenia przez wielomian n -tego stopnia będzie miała stopień co najwyżej $n-1$, co jest drugim warunkiem koniecznym, aby wielomian $c^{(+i)}(x)$ był wielomianem kodowym kodu o długości n . W takim razie można stwierdzić, że $c^{(+i)}(x)$ reprezentuje prawidłowy wielomian kodowy tego samego kodu cyklicznego, do którego należy wielomian $c(x)$. Oczywiście, powyższe rozważania są również prawdziwe dla przesunięcia cyklicznego w prawo, ponieważ przesunięcie cykliczne słowa kodowego o i pozycji w jedną stronę jest równoważne przesunięciu cyklicznemu o $(n-i)$ pozycji w przeciwną stronę.

Podsumowując, taka metoda może służyć do generowania kodu cyklicznego (n, k) , **pod warunkiem, że wielomian generujący $g(x)$ będzie czynnikiem wielomianu (x^n+1)** . Można łatwo zauważyć, że warunkiem koniecznym (ale niewystarczającym) aby $g(x)$ był czynnikiem (x^n+1) jest występowanie w wielomianie $g(x)$ składnika x^0 (czyli „1”) — w innym wypadku, w reszcie z dzielenia wielomianu (x^n+1) przez $g(x)$ zawsze wystąpi składnik x^0 .

Niestety **taka metoda kodowania daje w wyniku kod niesystematyczny**. W celu otrzymania kodu systematycznego należy zmodyfikować sposób kodowania tak, aby k najbardziej znaczących bitów słowa kodowego było równe nadawanej informacji. W tym celu należy przesunąć (niecyklicznie!) słowo informacyjne o $n-k$ pozycji w lewo, uzyskując w ten sposób ostateczną postać części informacyjnej słowa kodowego i jednocześnie robiąc miejsce dla części nadmiarowej. Następnie należy obliczyć $n-k$ bitową część nadmiarową słowa kodowego tak, aby całe słowo kodowe było wielokrotnością wielomianu generującego kod. Najprostszą metodą uzyskania części nadmiarowej jest obliczenie reszty z dzielenia przesuniętego wcześniej o $n-k$ pozycji w lewo słowa informacyjnego przez wielomian generujący. Dodając (odejmując) tak uzyskaną resztę do przesuniętej informacji otrzymamy słowo kodowe. Korzystając z równania (2) można napisać:

$$y \bmod x = r \Leftrightarrow y = n \cdot x + r$$

więc:

$$y - r = n \cdot x$$

czyli:

$$y - (y \bmod x) = n \cdot x$$

Czyli odejmując resztę od dzielnej otrzymamy liczbę będącą wielokrotnością dzielnika. Podstawiając za $y \leftarrow (x^{n-k} \cdot m(x))$ oraz za $x \leftarrow g(x)$ można zapisać:

$$c(x) = (x^{n-k} \cdot m(x)) + (x^{n-k} \cdot m(x)) \bmod g(x) \quad (6)$$

Jak widać, taka metoda kodowania zapewnia, że wielomian $c(x)$ będzie wielokrotnością wielomianu $g(x)$. Można również zauważyć, że ponieważ stopień $g(x)$ jest równy $n-k$, to stopień wyrażenia $(x^{n-k} \cdot m(x)) \bmod g(x)$ będzie mniejszy od $n-k$, czyli nie zmodyfikuje wcześniej przygotowanej części informacyjnej słowa kodowego (zajmie najwyżej $n-k$ bitów). **W takim razie taka metoda kodowania daje systematyczny kod liniowy (lub cykliczny).**

Przykład obliczenia słowa kodowego za pomocą powyższej zależności pokazano w rozwiązaniu zadania nr 2 na stronie 31.

5.2 Kodowanie za pomocą macierzy generującej

Kodowanie informacji metodą macierzową polega na obliczeniu iloczynu c wektora informacyjnego m i macierzy generującej kod G .

$$c = m \cdot G$$

gdzie:

- c – wektor kodowy o wymiarze $[1 \times n]$,
- m – wektor informacyjny o wymiarze $[1 \times k]$,
- G – macierz generująca o wymiarze $[k \times n]$.

Jeżeli potraktujemy wiersze macierzy generującej jako wektory, to wynik mnożenia wektora informacyjnego przez macierz generującą możemy zapisać jako sumę jej wierszy pomnożonych przez odpowiednie współrzędne wektora informacyjnego.

$$\begin{aligned}
 [m_2 \ m_1 \ m_0] \cdot \begin{bmatrix} a_{15} & a_{14} & a_{13} & a_{12} & a_{11} & a_{10} \\ a_{25} & a_{24} & a_{23} & a_{22} & a_{21} & a_{20} \\ a_{35} & a_{34} & a_{33} & a_{32} & a_{31} & a_{30} \end{bmatrix} &= m_2 \cdot [a_{15} \ a_{14} \ a_{13} \ a_{12} \ a_{11} \ a_{10}] \\
 &+ m_1 \cdot [a_{25} \ a_{24} \ a_{23} \ a_{22} \ a_{21} \ a_{20}] \\
 &+ m_0 \cdot [a_{35} \ a_{34} \ a_{33} \ a_{32} \ a_{31} \ a_{30}] \\
 &= [c_5 \ c_4 \ c_3 \ c_2 \ c_1 \ c_0]
 \end{aligned}$$

Jak widać, wektor kodowy powstaje w wyniku sumowania pewnych wektorów pomnożonych przez skalar, czyli stanowi kombinację liniową tych wektorów. W rzeczywistości **wiersze macierzy generującej są wektorami kodowymi kodu liniowego (cyklicznego)**, a w sensie algebraicznym stanowią bazę liniowej przestrzeni wektorowej reprezentującej ten kod. Można zatem zauważyć, że zasada tworzenia wektora kodowego metodą macierzową jest bardzo podobna do tworzenia słów kodowych poprzez sumowanie innych słów kodowych tego samego kodu (kryterium liniowości).

Poniżej przedstawiono przykład obliczenia słowa kodowego systematycznego kodu liniowego za pomocą macierzy generującej.

Przykład 9

Dana jest macierz G generująca systematyczny kod cykliczny (8, 3). Obliczyć słowo kodowe odpowiadające informacji $m(x) = x + 1$.

$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Słowo informacyjne w postaci bitowej ma postać $m = [011]$. Obliczenie słowa kodowego polega na przemnożeniu słowa informacyjnego przez macierz generującą kod.

$$\begin{aligned}
 c &= [011] \cdot \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix} = \\
 &= 0 \cdot [10011001] + 1 \cdot [01010101] + 1 \cdot [00110011] = \\
 &= [01010101] + [00110011] = \\
 &= [01100110]
 \end{aligned}$$

Słowo kodowe odpowiadające informacji $m(x) = x + 1$ ma postać $c(x) = x^6 + x^5 + x^2 + x$.

Macierz generującą kod można obliczyć na podstawie wielomianu generującego. Jedną z metod utworzenia macierzy jest metoda polegająca na wpisywaniu w jej kolejne wiersze przesuwanego wielomianu generującego $g(x)$. Sposób ten pokazano poniżej.

$$G = \begin{bmatrix} x^{k-1} \cdot g(x) \\ x^{k-2} \cdot g(x) \\ \vdots \\ x^1 \cdot g(x) \\ x^0 \cdot g(x) \end{bmatrix}$$

Jak już wspomniano wcześniej, macierz generująca kod liniowy musi posiadać k wierszy, n kolumn oraz wszystkie wiersze muszą reprezentować liniowo niezależne¹ wektory kodowe. Jak widać, taki sposób pozwala utworzyć macierz posiadającą k wierszy, a stopień wielomianu generującego równy $(n-k)$ gwarantuje, że pierwszy wiersz będzie stopnia $(n-k) + (k-1) = n-1$, co w formie bitowej odpowiada n pozycjom znaczącym (n kolumn). Oczywiście wszystkie wiersze stanowią wielokrotności wielomianu generującego, których stopień nie przekracza $(n-1)$, czyli reprezentują poprawne wektory kodowe kodu generowanego przez $g(x)$. Dodatkowo widać, że wszystkie wiersze są liniowo niezależne.

Niestety taka macierz nie generuje kodu systematycznego (podobnie jak zależność 5). Jak już wcześniej wspomniano, w kodzie systematycznym część informacyjna każdego słowa kodowego jest identyczna ze słowem informacyjnym, któremu to słowo kodowe odpowiada. W celu uzyskania takiej postaci słowa kodowego należy przepisać bez zmian słowo informacyjne w część informacyjną słowa kodowego, co można zrealizować poprzez odpowiednie przekształcenie macierzy generującej uzyskanej w wyniku przesuwania wielomianu generującego. Przekształcenie to polega na sumowaniu wierszy macierzy tak, aby jej lewa strona miała postać macierzy jednostkowej. Z uwagi na to, że mamy do czynienia z kodem liniowym, sumowanie wektorów kodowych reprezentowanych przez wiersze macierzy generującej daje w wyniku również wiersze reprezentujące wektory kodowe tego samego kodu. Poniżej przedstawiono przykład utworzenia macierzy generującej kod systematyczny za pomocą przesuwania wielomianu generującego.

Przykład 10

Dany jest wielomian generujący cykliczny kod binarny (14, 6) $g(x) = x^8 + x^6 + x^4 + 1$. Utworzyć macierz generującą kod systematyczny oparty na wielomianie $g(x)$.

$$G' = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} \text{I} \\ \text{II} \\ \text{III} \\ \text{IV} \\ \text{V} \\ \text{VI} \end{matrix}$$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{matrix} \text{I+III} \\ \text{II+IV} \\ \text{III+V} \\ \text{IV+VI} \\ \text{V} \\ \text{VI} \end{matrix}$$

¹ Wektory a , b , c są liniowo niezależne gdy ich kombinacja liniowa $\alpha \cdot a + \beta \cdot b + \gamma \cdot c = 0$ tylko dla $\alpha = \beta = \gamma = 0$

Inną metodą uzyskania macierzy generującej kod systematyczny jest metoda wykorzystująca zależność 6. Analizując poprzedni przykład można zauważyć, że kolejne wiersze macierzy generującej G reprezentują słowa kodowe kodu systematycznego odpowiadające wielomianom informacyjnym $x^{k-1}, x^{k-2}, \dots, x^1, x^0$. W takim razie pierwszy wiersz macierzy generującej kod systematyczny można obliczyć z zależności 6 przyjmując wielomian informacyjny $m(x) = x^{k-1}$. Poniżej przedstawiono sposób obliczenia pierwszego wiersza macierzy generującej kod systematyczny z poprzedniego przykładu.

Przykład 11

Dany jest wielomian generujący cykliczny kod binarny (14, 6) $g(x) = x^8 + x^6 + x^4 + 1$. Obliczyć pierwszy wiersz macierzy generującej kod systematyczny oparty na wielomianie $g(x)$.

Pierwszy wiersz macierzy generującej kod systematyczny ma postać słowa kodowego odpowiadającego informacji $m(x) = x^{k-1}$. Słowo kodowe można obliczyć korzystając z zależności 6:

$$c(x) = (x^{n-k} \cdot m(x)) + (x^{n-k} \cdot m(x)) \bmod g(x)$$

podstawiając parametry rozpatrywanego kodu otrzymujemy

$$c(x) = (x^8 \cdot x^5) + (x^8 \cdot x^5) \bmod (x^8 + x^6 + x^4 + 1)$$

$$c(x) = (x^{13}) + (x^{13}) \bmod (x^8 + x^6 + x^4 + 1)$$

Obliczamy resztę z dzielenia x^{13} przez wielomian generujący kod $g(x) = x^8 + x^6 + x^4 + 1$:

$$\begin{array}{r} 101010001 \overline{) 10000000000000} \\ \underline{101010001} \\ 010100010 \\ \underline{000000000} \\ 101000100 \\ \underline{101010001} \\ 000101010 \\ \underline{000000000} \\ 001010100 \\ \underline{000000000} \\ 010101000 \\ \underline{000000000} \\ 10101000 \end{array}$$

Jak widać reszta z dzielenia ma postać 10101000, więc pierwszy wiersz macierzy generującej ma postać 10000010101000.

W taki sposób można obliczyć wszystkie wiersze macierzy generującej kod systematyczny oparty na zadanym wielomianie generującym, jednak wymagałoby to przeprowadzenia $k-1$ dzielen¹. Analizując powyższy przykład można zauważyć, że kolejno otrzymywane reszty pośrednie są resztami z dzielenia wielomianu x^{n-k} przesuwanego kolejno o jedną pozycję w lewo, co odpowiada kolejnym częściom informacyjnym wierszy macierzy generującej. W takim razie obliczenia można znacząco uprościć wykorzystując wszystkie reszty pośrednie zapisywane w procesie dzielenia. Sposób ten ilustruje poniższy przykład. Pogrubioną czcionką zaznaczono reszty pośrednie wpisywane w kolejne wiersze części kontrolnej macierzy generującej² od dołu do góry.

¹ Ostatni wiersz macierzy ma postać wielomianu generującego, więc dzielenie nie jest potrzebne.

² Część kontrolną macierzy generującej stanowi $n-k$ kolumn po prawej stronie macierzy, a część jednostkową stanowi k kolumn po lewej stronie macierzy.

Przykład 12

Dany jest wielomian generujący cykliczny kod binarny (14, 6) $g(x) = x^8 + x^6 + x^4 + 1$. Obliczyć macierz generującą kod systematyczny oparty na wielomianie $g(x)$.

W pierwszym etapie konstruujemy macierz jednostkową $k \times k$ oraz pozostawiamy miejsce dla $n-k$ kolumn po prawej stronie.

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & - & - & - & - & - & - & - & - \\ 0 & 1 & 0 & 0 & 0 & 0 & - & - & - & - & - & - & - & - \\ 0 & 0 & 1 & 0 & 0 & 0 & - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 1 & 0 & 0 & - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & 1 & 0 & - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & 0 & 1 & - & - & - & - & - & - & - & - \\ 0 & 0 & 0 & 0 & 0 & 1 & - & - & - & - & - & - & - & - \end{bmatrix}$$

Następnie przeprowadzamy dzielenie ciągu n -bitowego z jedynką na najbardziej znaczącej pozycji i zerami na pozostałych $n-1$ pozycjach. Ciąg taki odpowiada wielomianowi x^{n-1} pomnożonemu przez wielomian x^{n-k} (patrz zależność 6).

$$\begin{array}{r} 101010001 \overline{) 10000000000000} \\ \underline{101010001} \\ \text{wiersz VI} \quad \mathbf{010100010} \\ \underline{000000000} \\ \text{wiersz V} \quad \mathbf{101000100} \\ \underline{101010001} \\ \text{wiersz IV} \quad \mathbf{000101010} \\ \underline{000000000} \\ \text{wiersz III} \quad \mathbf{001010100} \\ \underline{000000000} \\ \text{wiersz II} \quad \mathbf{010101000} \\ \underline{000000000} \\ \text{wiersz I} \quad \mathbf{10101000} \end{array}$$

Kolejno otrzymywane reszty (bez bitów przepisywanych z dzielnej) wpisujemy w kolejne wiersze części kontrolnej macierzy generującej zaczynając od ostatniego wiersza.

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Jak widać, otrzymana macierz jest zgodna z macierzą generującą otrzymaną w przykładzie 10. Sposób obliczenia macierzy generującej kod systematyczny z wykorzystaniem dzielenia jest najczęściej mniej pracochłonny niż sposób wykorzystujący dodawanie wierszy.

6 Test nr 2 — zadania

1. Poniżej podano sześć wielomianów.

$$g_1(x) = x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$$

$$g_2(x) = x^5 + x^4 + x^3 + x^2 + x$$

$$g_3(x) = x^5 + x^2 + x + 1$$

$$g_4(x) = x^6 + x^4 + x^3 + x + 1$$

$$g_5(x) = x^6 + x^5 + x^3 + x$$

$$g_6(x) = x^5 + x^3 + 1$$

Odpowiedz na następujące pytania.

- 1) Które wielomiany mogą generować kod liniowy?
- 2) Które wielomiany mogą generować kod cykliczny?
- 3) Które wielomiany nie mogą generować kodu liniowego (14, 8)?
- 4) Które wielomiany nie mogą generować kodu cyklicznego (14, 8)?
- 5) Które wielomiany generują kod liniowy (14, 8)?
- 6) Które wielomiany generują kod cykliczny (14, 8)?

Rozwiązanie:

Ad. 1) Wprowadzcie kody generowane przez wielomiany nie posiadające składnika x^0 nie są kodami dobrej jakości (najmniej znaczący bit w każdym słowie kodowym jest równy 0) jednak spełniają one kryterium liniowości. W takim razie kod liniowy może generować każdy wielomian z podanych wyżej.

Ad. 2) Jak już wcześniej wykazano, wszystkie wielomiany będące czynnikami wielomianu $x^n + 1$ generują kod cykliczny. Można łatwo zauważyć, że dla każdego wielomianu posiadającego składnik x^0 można znaleźć wielomian postaci $x^n + 1$, którego jest on czynnikiem. W takim razie kod cykliczny może generować każdy wielomian posiadający składnik x^0 , czyli wielomiany $g_1(x)$, $g_3(x)$, $g_4(x)$ i $g_6(x)$.

Ad. 3) Wielomian generujący kod liniowy o parametrach (n, k) musi mieć stopień równy $(n - k)$, czyli w przypadku kodu (14, 8) stopień wielomianu generującego musi wynosić 6. W zadaniu podano trzy wielomiany stopnia różnego od 6: $g_2(x)$, $g_3(x)$ i $g_6(x)$.

Ad. 4) Wielomian generujący kod cykliczny o parametrach (n, k) musi mieć stopień równy $(n - k)$ oraz musi być czynnikiem wielomianu $x^n + 1$, czyli w przypadku kodu (14, 8) stopień wielomianu generującego musi wynosić 6 i musi on być czynnikiem wielomianu $x^{14} + 1$. Warunkiem koniecznym do tego żeby dany wielomian był czynnikiem wielomianu posiadającego składnik x^0 jest obecność składnika x^0 w tym wielomianie, w takim razie wielomiany $g_2(x)$, $g_3(x)$, $g_5(x)$ i $g_6(x)$ nie mogą generować kodu cyklicznego (14, 8).

Ad. 5) Kod liniowy (14, 8) może generować każdy wielomian stopnia równego 6, czyli są to wielomiany $g_1(x)$, $g_4(x)$ i $g_5(x)$.

Ad. 6) Kod cykliczny (14, 8) może generować każdy wielomian stopnia równego 6, który jest czynnikiem wielomianu $x^{14} + 1$. W treści zadania podano dwa wielomiany stopnia szóstego posiadające składnik x^0 ($g_1(x)$ i $g_4(x)$), czyli tylko one mają szanse być czynnikiem wielomianu $x^{14} + 1$. W celu sprawdzenia tego faktu należy wykonać dzielenie wielomianu $(x^{14} + 1)$ przez $g_1(x)$ oraz dzielenie wielomianu $(x^{14} + 1)$ przez $g_4(x)$ i sprawdzić reszty z tych dzieleni. Po wykonaniu dzieleni okazuje się, że reszta z dzielenia $(x^{14} + 1)$ przez $g_1(x)$ wynosi 0, a reszta z dzielenia $(x^{14} + 1)$ przez $g_4(x)$ wynosi $(x^4 + x + 1)$, więc kod cykliczny (14, 8) generuje tylko wielomian $g_1(x)$.

Za każdą prawidłową odpowiedź będzie dodany 1 pkt., a za każdą nieprawidłową odpowiedź lub brak odpowiedzi będzie odjęty 1 pkt. W przypadku uzyskania za pytanie punktów ujemnych nie będą one uwzględniane w ogólnej punktacji z testu.

2. Dany jest wielomian generujący kod liniowy $g(x) = x^{12} + x^{10} + x^2 + x + 1$ oraz wielomian informacyjny $m(x) = x^3 + x$. Obliczyć metodą wielomianową słowo kodowe kodu systematycznego opartego na wielomianie $g(x)$ odpowiadające informacji $m(x)$.

Rozwiązanie:

W celu rozwiązania zadania należy skorzystać z zależności 6:

$$c(x) = (x^{n-k} \cdot m(x)) + (x^{n-k} \cdot m(x)) \bmod g(x)$$

Zarówno wielomian $m(x)$ jak i $g(x)$ są podane w treści zadania, a na podstawie stopnia wielomianu generującego możemy określić długość części nadmiarowej słowa kodowego ($n - k$).

W pierwszym etapie należy obliczyć iloczyn $(x^{n-k} \cdot m(x))$ odpowiadający części informacyjnej słowa kodowego:

$$x^{12} \cdot (x^3 + x) = x^{15} + x^{13}$$

Następnie należy obliczyć resztę z dzielenia otrzymanego w ten sposób wielomianu przez wielomian generujący, która stanowi część nadmiarową słowa kodowego:

$$\begin{array}{r} x^{12} + x^{10} + x^2 + x + 1 \overline{) x^{15} + x^{13}} \\ \underline{x^{15} + x^{13} + x^5 + x^4 + x^3} \\ \phantom{x^{12} + x^{10} + x^2 + x + 1} \phantom{x^{15} + x^{13}} \end{array}$$

Po dodaniu wielomianu reprezentującego część informacyjną do wielomianu reprezentującego część nadmiarową słowa kodowego można podać szukane słowo kodowe:

$$c(x) = x^{15} + x^{13} + x^5 + x^4 + x^3$$

W teście należy podać:

- wielomian reprezentujący część informacyjną słowa kodowego,
- wielomian reprezentujący szukane słowo kodowe.

Jeżeli odpowiedź a) jest niepoprawna to za zadanie zostanie przyznane 0 pkt., w przeciwnym wypadku, jeżeli odpowiedź b) jest niepoprawna to za zadanie zostanie przyznany 1 pkt, w przeciwnym wypadku za zadanie zostaną przyznane 4 pkt.

3. Dany jest wielomian generujący kod liniowy o długości 14: $g(x) = x^8 + x^6 + x^4 + 1$. Obliczyć macierz generującą systematyczny kod liniowy oparty na tym wielomianie.

Rozwiązanie zadania podano w przykładach 10 ÷ 12.

Za prawidłową i kompletną odpowiedź zostaną przyznane 4 pkt. Za każdy błędny bit w macierzy zostanie odjęty 1 pkt. W przypadku uzyskania za pytanie punktów ujemnych nie będą one uwzględniane w ogólnej punktacji z testu.

4. Dana jest macierz G generująca systematyczny kod liniowy. Obliczyć metodą macierzową słowo kodowe tego kodu odpowiadające informacji $m(x) = x^3 + x$.

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Rozwiązanie:

W celu rozwiązania zadania należy pomnożyć podaną macierz generującą kod przez podany wektor informacyjny.

W pierwszym etapie przekształcamy wektor informacyjny na postać bitową

$$x^3 + x \leftrightarrow [1010]$$

Następnie skreślamy te wiersze macierzy, które są mnożone przez współrzędne wektora informacyjnego mające wartość równą 0:

$$c = [1010] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{bmatrix}$$

Dodajemy wiersze, które pozostały i otrzymujemy:

$$c = [1 \ 0 \ 1 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 0 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0]$$

Po przekształceniu na postać wielomianową zapisujemy szukane słowo kodowe:

$$c(x) = x^{15} + x^{13} + x^5 + x^4 + x^3$$

Za prawidłową i kompletną odpowiedź, za zadanie zostaną przyznane 4 pkt. W przypadku błędnej odpowiedzi, jeżeli zostaną skreślone odpowiednie wiersze macierzy – za zadanie zostanie przyznany 1 pkt.

Tabela 4 Punktacja zadań testu nr 2

Numer zadania	Maksymalna liczba punktów
1	6
2	4
3	4
4	4
Suma	18

7 Dekodowanie korekcyjne

W poprzednim rozdziale pokazano sposób tworzenia słów kodowych na podstawie słów informacyjnych. Tak przygotowane słowa kodowe wysyła się do odbiorcy wykorzystując różne media transmisyjne (np. przewody miedziane, fale elektromagnetyczne, nośniki danych itp.). Niestety, słowa kodowe w trakcie transmisji są narażone na przekłamania, które mogą wynikać zarówno z niedoskonałości medium transmisyjnego jak i zakłóceń zewnętrznych i wewnętrznych. W rezultacie, do odbiornika mogą dotrzeć słowa różniące się od wysłanych słów kodowych, co może spowodować otrzymanie błędnej informacji.

W celu zminimalizowania wpływu zakłóceń na przesyłane lub przechowywane informacje stosuje się dekodowanie z korekcją błędów, które jest realizowane w układach dekodatorów. Zadaniem dekodera jest przede wszystkim wykrycie błędów, a w przypadku kodów umożliwiających korekcję błędów, również ich skorygowanie. Dekodowanie korekcyjne można przeprowadzić różnymi metodami w zależności od rodzaju wykorzystywanego kodu (kod liniowy, kod cykliczny czy konkretny rodzaj kodu cyklicznego np. kod Reeda-Solomona).

W przypadku wykorzystywania kodów liniowych, dekodowanie i korekcję błędów można zrealizować poprzez zastosowanie metody wykorzystującej tablicę standardową kodu liniowego. Niestety jest to metoda nieefektywna, ponieważ wymaga użycia pamięci w dekodерze, co w wypadku długich kodów jest niepraktyczne zarówno ze względu na rozmiar pamięci jak i złożoność obliczeniową przeszukiwania tablicy. Zwiększenie wydajności tej metody można uzyskać poprzez zastąpienie tablicy standardowej tablicą zawierającą syndromy i przyporządkowane im wektory błędów. W takim wypadku dekodowanie i korekcja błędów polega na obliczeniu syndromu słowa odebranego, odszukaniu syndromu w tablicy, a następnie dodaniu odpowiadającego mu wektora błędów do słowa odebranego. Niestety taka metoda również wymaga użycia pamięci oraz jednostki arytmetyczno-logicznej w urządzeniu odbiorczym, co nie pozostaje bez wpływu na koszt dekodera.

W praktyce najczęściej stosowaną klasą kodów blokowych są kody cykliczne, których właściwości pozwalają zwiększyć wydajność i obniżyć koszty urządzeń dekodujących. Z uwagi na specyficzne właściwości różnych rodzajów kodów cyklicznych stosuje się wiele różnych algorytmów dekodowania i korekcji błędów. W niniejszym rozdziale zostanie opisany uproszczony algorytm dekodowania korekcyjnego, który może być zastosowany dla wszystkich kodów cyklicznych.

7.1 Syndrom

Jak wiadomo, przy kodowaniu z wykorzystaniem wielomianu generującego wszystkie wielomiany kodowe są wielokrotnościami wielomianu generującego, czyli dzielą się przez wielomian generujący bez reszty. W celu sprawdzenia, po stronie odbiorczej, czy słowo odebrane jest słowem kodowym, wystarczy sprawdzić resztę z dzielenia wielomianu odebranego $u(x)$ przez wielomian generujący $g(x)$.

$$s(x) = u(x) \bmod g(x) \quad (7)$$

Reszta $s(x)$ z tego dzielenia zwana jest syndromem i ma stopień mniejszy niż $n - k$, czyli ma długość równą części nadmiarowej słowa kodowego.

Przekształcając powyższe równanie zgodnie z zależnością (2) można napisać:

$$u(x) = m'(x) \cdot g(x) + s(x)$$

czyli

$$u(x) + s(x) = m'(x) \cdot g(x)$$

gdzie $m'(x)$ jest pewnym wielomianem stopnia mniejszego niż k .

Widać więc, że **suma słowa odebranego $u(x)$ i syndromu $s(x)$ będzie wielokrotnością wielomianu generującego, czyli da prawidłowe słowo kodowe**. Niestety słowo kodowe otrzymane w ten sposób może się różnić od nadanego słowa kodowego. Wynika to z faktu, że syndrom ma długość równą części nadmiarowej słowa kodowego, więc k najbardziej znaczących bitów (część informacyjna) słowa odebranego nie ulegnie zmianie po dodaniu do niego syndromu.

W praktyce, syndrom reprezentuje różnicę między odebraną częścią nadmiarową ($n - k$ najmniej znaczących bitów słowa odebranego), a częścią nadmiarową prawidłowego słowa kodowego odpowiadającą odebranej części informacyjnej (k najbardziej znaczących bitów słowa odebranego).

Przykład 13

Po stronie nadawczej wysłano słowo kodowe $c_t(x)$ należące do systematycznego kodu cyklicznego $(12, 3)$ opartego na wielomianie generującym $g(x)$. Słowo kodowe odpowiada informacji $m_t(x)$. W trakcie transmisji zostały przekłamanne dwa najbardziej znaczące bity wysłanego słowa kodowego, co spowodowało odebranie słowa $u(x)$. Obliczyć metodą wielomianową syndrom błędu $s(x)$.

$$g(x) = x^9 + x^8 + x^5 + x^4 + x + 1$$

$$m_t(x) = x + 1$$

$$c_t(x) = x^{10} + x^9 + x^6 + x^5 + x^2 + x$$

$$e(x) = x^{11} + x^{10}$$

$$u(x) = x^{11} + x^9 + x^6 + x^5 + x^2 + x$$

Oczywiście, ani wektor błędu $e(x)$ ani wysłane słowo kodowe $c_t(x)$ nie są znane po stronie odbiorczej.

Obliczamy syndrom błędu $s(x)$ wykorzystując zależność (7):

$$\begin{array}{r} x^9 + x^8 + x^5 + x^4 + x + 1 \overline{) x^{11} + x^9 + x^6 + x^5 + x^2 + x} \\ \underline{x^{11} + x^{10} + x^7 + x^6 + x^3 + x^2} \\ x^{10} + x^9 + x^7 + x^5 + x^3 + x \\ \underline{x^{10} + x^9 + x^6 + x^5 + x^2 + x} \\ x^7 + x^6 + x^3 + x^2 \end{array}$$

Jak widać, syndrom błędu $s(x) = x^7 + x^6 + x^3 + x^2$.

Jeżeli dodamy otrzymany syndrom $s(x)$ do słowa odebranego $u(x)$ otrzymamy poprawne słowo kodowe $c_r(x)$.

$$c_r(x) = x^{11} + x^9 + x^7 + x^5 + x^3 + x$$

Niestety, słowo kodowe $c_r(x)$ odpowiada informacji $m_r(x) = x^2 + 1$, która różni się od informacji nadanej $m_t(x)$.

Syndrom błędów zawiera w sobie informację o błędach, które wystąpiły w trakcie transmisji. W celu zbadania tej właściwości syndromu zauważmy, że wielomian odebrany można zapisać jako sumę nadanego wielomianu kodowego $c_t(x)$ i wielomianu (wektora) błędu $e(x)$:

$$u(x) = c_t(x) + e(x)$$

podstawiając powyższą zależność do (7) otrzymamy:

$$s(x) = (c_t(x) + e(x)) \bmod g(x)$$

ponieważ $c(x) \bmod g(x) = 0$ to można napisać¹:

$$s(x) = e(x) \bmod g(x)$$

¹ Wynika to z właściwości dodawania stronami kongruencji: jeżeli $a \equiv b \bmod g$ oraz $z \equiv y \bmod g$ to $a + z \equiv (b + y) \bmod g$.

Przykład 14

Obliczmy syndrom $s(x)$ dzieląc wektor błędu $e(x)$ przez wielomian generujący $g(x)$. Wszystkie dane takie jak w poprzednim przykładzie.

$$x^9 + x^8 + x^5 + x^4 + x + 1 \overline{) x^{11} + x^{10} + x^7 + x^6 + x^3 + x^2}$$

Syndrom błędu $s(x) = x^7 + x^6 + x^3 + x^2$ — czyli jest taki sam jak syndrom obliczony w poprzednim przykładzie jako reszta z dzielenia wielomianu odebranego $u(x)$ przez wielomian generujący $g(x)$.

Jak widać **syndrom zależy tylko od wektora błędu (nie zależy od nadanego słowa kodowego)**. Dodatkowo, jeżeli stopień wielomianu $e(x)$ jest mniejszy niż $(n - k)$, czyli błędy nie występują w części informacyjnej słowa kodowego, to reszta z dzielenia $e(x)$ przez wielomian wyższego stopnia (czyli $n - k$) będzie równa $e(x)$, więc mamy:

$$s(x) = e(x) \quad (8)$$

Z tego wynika, że **jeżeli przekłamanie nie wystąpiło w części informacyjnej nadanego słowa kodowego to syndrom jest równy wektorowi błędu**. W takim wypadku korekcja błędów polega na dodaniu syndromu do wektora odebranego — wtedy otrzymujemy nadane słowo kodowe.

Przykład 15

Po stronie nadawczej wysłano słowo kodowe $c_t(x)$ należące do systematycznego kodu cyklicznego $(12, 3)$ opartego na wielomianie generującym $g(x)$. Słowo kodowe odpowiada informacji $m_t(x)$. W trakcie transmisji zostały przekłamane dwa najmniej znaczące bity wysłanego słowa kodowego, co spowodowało odebranie słowa $u(x)$. Obliczyć metodą wielomianową syndrom błędu $s(x)$.

$$g(x) = x^9 + x^8 + x^5 + x^4 + x + 1$$

$$m_t(x) = x + 1$$

$$c_t(x) = x^{10} + x^9 + x^6 + x^5 + x^2 + x$$

$$e(x) = x + 1$$

$$u(x) = x^{10} + x^9 + x^6 + x^5 + x^2 + 1$$

Obliczamy syndrom błędu $s(x)$ wykorzystując zależność (7):

$$x^9 + x^8 + x^5 + x^4 + x + 1 \overline{) x^{10} + x^9 + x^6 + x^5 + x^2 + 1}$$

$$\underline{x^{10} + x^9 + x^6 + x^5 + x^2 + x}$$

$$x + 1$$

Jak widać, syndrom błędu $s(x) = x + 1$, więc jest równy wektorowi błędów $e(x)$.

Jeżeli dodamy otrzymany syndrom $s(x)$ do słowa odebranego $u(x)$ otrzymamy poprawne słowo kodowe $c_r(x)$.

$$c_r(x) = x^{10} + x^9 + x^6 + x^5 + x^2 + x$$

Ponieważ nie wystąpiły błędy w części informacyjnej słowa kodowego to słowo kodowe $c_r(x) = c_t(x)$.

Powyższy przykład pokazuje sposób przeprowadzenia korekcji błędów w przypadku, gdy nie wystąpiły przekłamanie w części informacyjnej wysłanego słowa kodowego. Niestety, jak to pokazano wcześniej, w przypadku wystąpienia błędów w części informacyjnej nie można ich skorygować za pomocą tej metody, jednak z uwagi na jej prostotę i dużą efektywność warto zastosować taki algorytm w praktyce. W tym celu urządzenie dekodujące musi sprawdzić czy wystąpiły błędy w części informacyjnej nadanego słowa kodowego.

Jak już wcześniej wspomniano, syndrom zawiera informację o rozkładzie błędów, które wystąpiły w trakcie transmisji słowa kodowego $c_t(x)$. Przyjmijmy, że słowo odebrane $u(x)$ zawiera t przekłamań, oraz chociaż jedno z nich leży w części informacyjnej. W takim wypadku, poprzez dodanie syndromu $s(x)$ do słowa odebranego $u(x)$, otrzymamy słowo kodowe $c_r(x)$, które oczywiście będzie różne od $c_t(x)$:

$$c_r(x) = u(x) + s(x) \text{ oraz } u(x) = c_t(x) + e(x)$$

czyli:

$$c_r(x) = c_t(x) + e(x) + s(x)$$

Wiemy, że różne słowa kodowe kodu liniowego leżą od siebie w odległości nie mniejszej niż odległość minimalna kodu d , czyli:

$$d_H(c_t(x), c_r(x)) \geq d$$

więc:

$$w_H(c_t(x) + c_r(x)) \geq d$$

czyli:

$$w_H(c_t(x) + c_t(x) + e(x) + s(x)) \geq d$$

zatem:

$$w_H(e(x) + s(x)) \geq d \quad (9)$$

Z powyższej zależności wynika, że w przypadku wystąpienia przekłamań w części informacyjnej, waga Hamminga sumy wektora błędów i syndromu będzie większa lub równa odległości minimalnej kodu. Korzystając z tego faktu można określić wagę Hamminga syndromu. W tym celu zauważmy, że suma wag Hamminga dwóch wektorów u i v jest większa lub równa wadze Hamminga sumy tych wektorów:

$$w_H(u) + w_H(v) \geq w_H(u + v)$$

uwzględniając zależność (9) można napisać:

$$w_H(e(x)) + w_H(s(x)) \geq w_H(e(x) + s(x)) \geq d$$

czyli:

$$w_H(e(x)) + w_H(s(x)) \geq d$$

więc:

$$w_H(s(x)) \geq d - w_H(e(x)) \quad (10)$$

Waga Hamminga wektora błędów $e(x)$ jest równa liczbie przekłamań, które wystąpiły w trakcie transmisji, więc zgodnie z wcześniejszym założeniem:

$$w_H(e(x)) = t$$

Nawiązując do wiadomości podanych w rozdziale 3.4 wiemy, że odległość minimalna kodu liniowego wynosi co najmniej:

$$d = 2 \cdot t + 1$$

W takim razie, podstawiając powyższe zależności do (10) można napisać:

$$w_H(s(x)) \geq (2 \cdot t + 1) - t$$

czyli:

$$w_H(s(x)) \geq t + 1 \quad (11)$$

Z powyższej zależności wynika, że **jeżeli nastąpiły przekłamania w części informacyjnej wektora kodowego, to waga Hamminga syndromu będzie większa od zdolności korekcyjnej kodu**, a z zależności (8) widać, że w przeciwnym wypadku **waga Hamminga syndromu nie przekroczy zdolności korekcyjnej kodu**.

Powyższe rozważania dotyczą tylko takich przypadków, w których liczba przekłamań nie przekracza zdolności korekcyjnej kodu t . Jeżeli liczba przekłamań będzie większa, to mogą wystąpić błędy korekcji lub nawet brak możliwości ich wykrycia.

7.2 Macierz korekcyjna

W poprzednim rozdziale pokazano dwie metody kodowania informacji: metodę opartą o wielomian generujący oraz metodę wykorzystującą macierz generującą kod G . Przy kodowaniu informacji metodą macierzową, wektor kodowy c oblicza się jako iloczyn wektora informacyjnego m oraz macierzy generującej G . Podobnie jak w przypadku kodowania informacji, również dekodowanie może być przeprowadzone metodą macierzową.

W macierzowej metodzie dekodowania także wykorzystuje się operację mnożenia macierzy, przy czym tutaj czynnikami mnożenia są słowo odebrane u oraz macierz korekcyjna transponowana H^T , a wynikiem jest wektor reprezentujący syndrom błędów s :

$$s = u \cdot H^T \quad (12)$$

gdzie:

- s – wektor syndromu błędów o wymiarze $[1 \times n-k]$,
- u – wektor odebrany o wymiarze $[1 \times n]$,
- H^T – macierz korekcyjna transponowana o wymiarze $[n \times n-k]$.

Na podstawie powyższej zależności, tak jak w metodzie wielomianowej, można pokazać, że syndrom nie zależy od nadanego słowa kodowego:

$$s = u \cdot H^T$$

podstawiając za słowo odebrane u sumę nadanego słowa kodowego c_r i wektora błędów e otrzymujemy:

$$\begin{aligned} s &= (c_r + e) \cdot H^T \\ s &= c_r \cdot H^T + e \cdot H^T \end{aligned}$$

ponieważ w przypadku braku błędów syndrom jest równy zero, czyli $c_r \cdot H^T = 0$, to

$$s = e \cdot H^T$$

Poniżej przedstawiono przykład obliczenia syndromu błędów metodą macierzową. Wartości wszystkich danych są takie same jak w przykładzie 13.

Przykład 16

Po stronie nadawczej wysłano słowo kodowe c_t należące do systematycznego kodu cyklicznego (12, 3). Słowo kodowe odpowiada informacji m_t . W trakcie transmisji zostały przekłamanie dwa najbardziej znaczące bity wysłanego słowa kodowego, co spowodowało odebranie słowa u . Obliczyć metodą macierzową syndrom błędu s . Macierz korekcyjna transponowana H^T jest podana poniżej.

$$H^T = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad \begin{array}{l} m_t = 011 \\ c_t = 011001100110 \\ e = 110000000000 \\ u = 101001100110 \end{array}$$

Oczywiście, ani wektor błędu e ani wysłane słowo kodowe c_t nie są znane po stronie odbiorczej.

Obliczamy syndrom błędu s wykorzystując zależność (12):

$$s = [101001100110] \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = [011001100]$$

Syndrom błędu $s = 011001100$.

Jak już wcześniej wspomniano, syndrom błędów reprezentuje różnicę między odebraną częścią nadmiarową ($n - k$ najmniej znaczących bitów słowa odebranego), a częścią nadmiarową prawidłowego słowa kodowego odpowiadającego odebranej części informacyjnej (k najbardziej znaczących bitów słowa odebranego). W takim razie obliczenie syndromu można podzielić na dwie operacje:

1. obliczenie prawidłowej części nadmiarowej na podstawie odebranej części informacyjnej,
2. dodanie (odjęcie) tak obliczonej części nadmiarowej od odebranej części nadmiarowej.

Na takiej zasadzie opiera się działanie macierzy korekcyjnej transponowanej. Część górna macierzy (k górnych wierszy) oblicza część nadmiarową słowa odebranego bazując na jego k bitach informacyjnych, a część jednostkowa ($n - k$ dolnych wierszy) przenosi niezmienną część nadmiarową słowa odebranego. Oczywiście, z uwagi na to, że jest to jedna macierz, wyniki obydwu części się dodają i otrzymujemy syndrom.

Do obliczenia prawidłowej części nadmiarowej odpowiadającej odebranej części informacyjnej wykorzystuje się część kontrolną macierzy generującej (jej $n - k$ kolumn po prawej stronie), mnożąc ją przez część informacyjną słowa odebranego. Fakt ten można wykorzystać do utworzenia macierzy H^T na podstawie macierzy G — wystarczy pod częścią kontrolną macierzy generującej G dopisać macierz jednostkową.

Przykład 17

Dana jest macierz \mathbf{G} generująca systematyczny kod liniowy (12, 3). Obliczyć macierz korekcyjną transponowaną \mathbf{H}^T dla tego kodu.

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

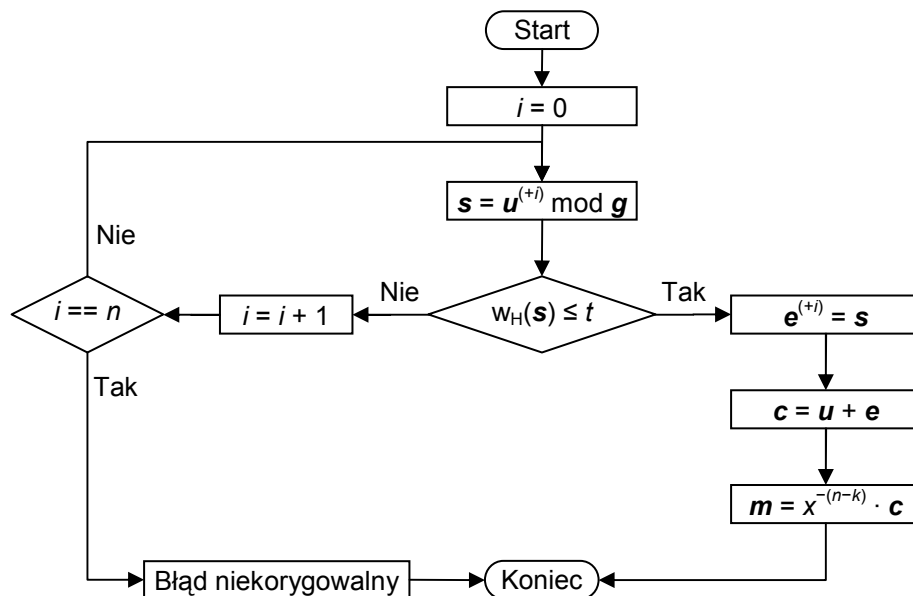
Przepisujemy część kontrolną macierzy generującej (zaznaczoną pogrubioną czcionką) i dopisujemy pod nią macierz jednostkową.

$$\mathbf{H}^T = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} & \mathbf{0} & \mathbf{1} \\ \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} & \mathbf{0} & \mathbf{0} & \mathbf{1} & \mathbf{1} \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

W taki sposób otrzymujemy macierz korekcyjną transponowaną \mathbf{H}^T .

7.3 Uproszczony algorytm dekodowania dla kodów cyklicznych

Podany poniżej algorytm opiera się na właściwościach syndromu omówionych poprzednio oraz na właściwości cykliczności kodu. Umożliwia on korekcję maksymalnie t przekłamań w słowie odebranym i może być stosowany dla wszystkich kodów cyklicznych.



Rys. 7 Algorytm dekodowania korekcyjnego dla systematycznych kodów cyklicznych

Działanie pokazanego algorytmu dekodowania polega na przesuwaniu cyklicznym słowa odebranego u do momentu, gdy wszystkie przekłamane bity znajdują się w części nadmiarowej. Wtedy waga Hamminga syndromu s nie przekroczy zdolności korekcyjnej kodu t i można przyjąć, że wektor błędów $e^{(i)}$ przesuniętego cyklicznie słowa odebranego $u^{(i)}$ jest równy bieżącemu syndromowi. Następnie należy przesunąć cyklicznie wektor błędów o i pozycji w prawo uzyskując wektor błędów e reprezentujący przekłamania, które wystąpiły podczas transmisji słowa kodowego. Obliczony w ten sposób wektor błędów dodaje się do słowa odebranego u , w wyniku czego otrzymuje się skorygowane słowo kodowe c . Teraz wystarczy zdekodować informację m poprzez przesunięcie (niecykliczne) słowa kodowego c o $n - k$ pozycji w prawo.

Jeżeli po $n - 1$ przesunięciach cyklicznych słowa odebranego u nie uzyskamy syndromu, którego waga Hamminga nie przekracza zdolności korekcyjnej kodu to znaczy, że wystąpiło zbyt dużo błędów lub ich rozłożenie uniemożliwia korekcję błędów za pomocą takiego algorytmu. W takim wypadku urządzenie dekodujące sygnalizuje błąd niekorygowalny.

Na rysunku 7 przedstawiono algorytm przystosowany do kodów systematycznych, jednak może on być również zastosowany do niesystematycznych kodów cyklicznych, pod warunkiem, że zostanie zmieniony sposób obliczenia informacji m . W przypadku kodów niesystematycznych wystarczy obliczyć wynik dzielenia wektora c przez wektor generujący g .

Poniżej przedstawiono przykład dekodowania korekcyjnego dla błędów wprowadzonych w przykładzie 13.

Przykład 18

Dekoder systematycznego kodu cyklicznego $(12, 3)$ opartego na wielomianie generującym $g(x) = x^9 + x^8 + x^5 + x^4 + x + 1$ odebrał słowo $u(x) = x^{11} + x^9 + x^6 + x^5 + x^2 + x$.

Obliczyć metodą wielomianową wektor błędu $e(x)$, nadane słowo kodowe $c(x)$ oraz nadany wielomian informacyjny $m(x)$. Odległość minimalna kodu $d = 6$.

Obliczamy zdolność korekcyjną kodu $t = \text{int}\left(\frac{6-1}{2}\right) = 2$

Obliczamy syndrom błędu $s_0(x)$ odpowiadający wielomianowi odebranemu $u(x)$:

$$\begin{array}{r} x^9 + x^8 + x^5 + x^4 + x + 1 \overline{) x^{11} + x^9 + x^6 + x^5 + x^2 + x} \\ \underline{x^{11} + x^{10} + x^7 + x^6 + x^3 + x^2} \\ x^{10} + x^9 + x^7 + x^5 + x^3 + x \\ \underline{x^{10} + x^9 + x^6 + x^5 + x^2 + x} \\ x^7 + x^6 + x^3 + x^2 \end{array}$$

Jak widać, waga Hamminga syndromu błędu $s_0(x) = x^7 + x^6 + x^3 + x^2$ jest równa 4, więc przekracza zdolność korekcyjną kodu. W takim wypadku nie można założyć, że syndrom $s_0(x)$ reprezentuje wektor błędu słowa odebranego $u(x)$.

Przesuwamy cyklicznie słowo odebrane o jedną pozycję w lewo: $u^{(+1)}(x) = x^{10} + x^7 + x^6 + x^3 + x^2 + 1$, a następnie obliczamy odpowiadający mu syndrom $s_1(x)$.

$$\begin{array}{r} x^9 + x^8 + x^5 + x^4 + x + 1 \overline{) x^{10} + x^7 + x^6 + x^3 + x^2 + 1} \\ \underline{x^{10} + x^9 + x^6 + x^5 + x^2 + x} \\ x^9 + x^7 + x^5 + x^3 + x + 1 \\ \underline{x^9 + x^8 + x^5 + x^4 + x + 1} \\ x^8 + x^7 + x^4 + x^3 \end{array}$$

Jak widać, waga Hamminga syndromu błędu $s_1(x) = x^8 + x^7 + x^4 + x^3$ jest równa 4, więc przekracza zdolność korekcyjną kodu. W takim wypadku nie można założyć, że syndrom $s_1(x)$ reprezentuje wektor błędu przesuniętego cyklicznie słowa odebranego $u^{(+1)}(x)$.

Przesuwamy cyklicznie słowo odebrane o dwie pozycje w lewo: $u^{(+2)}(x) = x^{11} + x^8 + x^7 + x^4 + x^3 + x$, a następnie obliczamy odpowiadający mu syndrom $s_2(x)$.

$$\begin{array}{r} x^9 + x^8 + x^5 + x^4 + x + 1 \overline{) x^{11} + x^8 + x^7 + x^4 + x^3 + x} \\ \underline{x^{11} + x^{10} + x^7 + x^6 + x^3 + x^2} \\ x^{10} + x^8 + x^6 + x^4 + x^2 + x \\ \underline{x^{10} + x^9 + x^6 + x^5 + x^2 + x} \\ x^9 + x^8 + x^5 + x^4 \\ \underline{x^9 + x^8 + x^5 + x^4 + x + 1} \\ x + 1 \end{array}$$

Jak widać, waga Hamminga syndromu błędu $s_2(x) = x + 1$ jest równa 2, więc nie przekracza zdolności korekcyjnej kodu. W takim wypadku można założyć, że syndrom $s_2(x)$ reprezentuje wektor błędu przesuniętego cyklicznie słowa odebranego $u^{(+2)}(x)$, czyli $e^{(+2)}(x) = x + 1$.

Obliczamy wektor błędu $e(x)$ poprzez przesunięcie wektora $e^{(+2)}(x)$ o dwie pozycje w prawo:

$$e(x) = x^{11} + x^{10}$$

Obliczamy skorygowane słowo kodowe poprzez dodanie wektora błędów $e(x)$ do słowa odebranego $u(x)$

$$c(x) = (x^{11} + x^9 + x^6 + x^5 + x^2 + x) + (x^{11} + x^{10}) = x^{10} + x^9 + x^6 + x^5 + x^2 + x$$

Obliczamy nadany wielomian informacyjny poprzez przesunięcie (niecykliczne) wielomianu $c(x)$ o $n - k$ ($12 - 3 = 9$) pozycji w prawo:

$$m(x) = x + 1$$

Jak widać, wektor błędów $e(x)$, słowo kodowe $c(x)$, oraz wielomian informacyjny $m(x)$ odpowiadają odpowiednim danym z przykładu 13.

W powyższym przykładzie obliczano syndrom wykorzystując rachunek wielomianów. Jak to pokazano wcześniej, syndrom można obliczyć używając transponowanej macierzy korekcyjnej. Poniżej przedstawiono przykład dekodowania korekcyjnego dla danych z przykładu 16.

Przykład 19

Dekoder systematycznego kodu cyklicznego odebrał słowo $u = 101001100110$.

Obliczyć metodą macierzową wektor błędu e , nadane słowo kodowe c oraz nadany wektor informacyjny m . Odległość minimalna kodu $d = 6$, macierz korekcyjna transponowana H^T jest podana poniżej.

$$H^T = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Obliczamy zdolność korekcyjną kodu $t = \text{int}\left(\frac{6-1}{2}\right) = 2$

Na podstawie liczby wierszy podanej macierzy widać, że $n = 12$.

Na podstawie liczby kolumn podanej macierzy widać, że $k = 12 - 9 = 3$.

Obliczamy syndrom błędu \mathbf{s}_0 odpowiadający wektorowi odebranemu \mathbf{u} :

$$\mathbf{s}_0 = [101001100110] \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = [011001100]$$

Jak widać, waga Hamminga syndromu błędu $\mathbf{s}_0 = 011001100$ jest równa 4, więc przekracza zdolność korekcyjną kodu. W takim wypadku nie można założyć, że syndrom \mathbf{s}_0 reprezentuje wektor błędu słowa odebranego \mathbf{u} .

Przesuwamy cyklicznie słowo odebrane o jedną pozycję w lewo: $\mathbf{u}^{(+1)} = 010011001101$, a następnie obliczamy odpowiadający mu syndrom \mathbf{s}_1 .

$$\mathbf{s}_1 = [010011001101] \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = [110011000]$$

Jak widać, waga Hamminga syndromu błędu $\mathbf{s}_1 = 110011000$ jest równa 4, więc przekracza zdolność korekcyjną kodu. W takim wypadku nie można założyć, że syndrom \mathbf{s}_1 reprezentuje wektor błędu przesuniętego cyklicznie słowa odebranego $\mathbf{u}^{(+1)}$.

Przesuwamy cyklicznie słowo odebrane o dwie pozycje w lewo: $\mathbf{u}^{(+2)} = 100110011010$, a następnie obliczamy odpowiadający mu syndrom \mathbf{s}_2 .

$$\mathbf{s}_2 = [100110011010] \cdot \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} = [000000011]$$

Jak widać, waga Hamminga syndromu błędu $\mathbf{s}_2 = 000000011$ jest równa 2, więc nie przekracza zdolności korekcyjnej kodu. W takim wypadku można założyć, że syndrom \mathbf{s}_2 reprezentuje wektor błędu przesuniętego cyklicznie słowa odebranego $\mathbf{u}^{(+2)}$, czyli $\mathbf{e}^{(+2)} = 00000000011$.

Obliczamy wektor błędu \mathbf{e} poprzez przesunięcie wektora $\mathbf{e}^{(+2)}$ o dwie pozycje w prawo:

$$\mathbf{e} = 110000000000$$

Obliczamy skorygowane słowo kodowe poprzez dodanie wektora błędów \mathbf{e} do słowa odebranego \mathbf{u}

$$\begin{array}{r} \mathbf{c} = 110000000000 \\ + 101001100110 \\ \hline = 011001100110 \end{array}$$

Obliczamy nadany wektor informacyjny \mathbf{m} poprzez przesunięcie (niecykliczne) wektora kodowego \mathbf{c} o $n - k$ ($12 - 3 = 9$) pozycji w prawo:

$$\mathbf{m} = 011$$

Jak widać, wektor błędów \mathbf{e} , słowo kodowe \mathbf{c} , oraz wielomian informacyjny \mathbf{m} odpowiadają odpowiednim danym z przykładu 16.

8 Test nr 3 — zadania

1. Dana jest macierz generująca G pewien systematyczny kod liniowy. Oblicz macierz korekcyjną transponowaną dla tego kodu, podaj długość słowa kodowego oraz długość słowa informacyjnego.

Rozwiązanie zadania podano w przykładzie 17.

Za prawidłową i kompletną odpowiedź, za zadanie zostaną przyznane 2 pkt.

2. Dane jest słowo odebrane $u(x) = x^{11} + x^9 + x^6 + x^5 + x^2 + x$ przez dekodery systematycznego kodu cyklicznego o długości 12. Odległość minimalna kodu wynosi 6, wielomian generujący kod $g(x) = x^9 + x^8 + x^5 + x^4 + x + 1$. Oblicz wektor błędu, nadane słowo kodowe oraz nadaną informację. Wyniki podać w postaci wielomianowej.

Rozwiązanie zadania podano w przykładzie 18.

W teście należy podać wszystkie syndromy użyte w trakcie obliczeń, wektor błędu, nadane słowo kodowe, oraz nadaną informację. Syndromy będą uwzględniane w punktacji pod warunkiem, że co najmniej jeden z nich będzie miał wagę Hamminga nieprzekraczającą zdolności korekcyjnej kodu

Za prawidłową i kompletną odpowiedź, za zadanie zostanie przyznanych 7 pkt.,

za prawidłowe: wszystkie syndromy użyte w trakcie obliczeń, wektor błędu i nadane słowo kodowe,

za zadanie zostanie przyznanych 5 pkt.,

za prawidłowe: wszystkie syndromy użyte w trakcie obliczeń i wektor błędu, za zadanie zostaną przyznane 4 pkt.,

za prawidłowe: wszystkie syndromy użyte w trakcie obliczeń, za zadanie zostaną przyznane 3 pkt.

3. Dane jest słowo odebrane $u(x) = x^{11} + x^9 + x^6 + x^5 + x^2 + x$ przez dekodery systematycznego kodu cyklicznego. Macierz korekcyjna transponowana H^T jest podana poniżej, odległość minimalna kodu wynosi $d = 6$. Oblicz wektor błędu, nadane słowo kodowe oraz nadaną informację. Wyniki podać w postaci wielomianowej.

Rozwiązanie zadania podano w przykładzie 19.

W teście należy podać wszystkie syndromy użyte w trakcie obliczeń, wektor błędu, nadane słowo kodowe, oraz nadaną informację. Syndromy będą uwzględniane w punktacji pod warunkiem, że co najmniej jeden z nich będzie miał wagę Hamminga nieprzekraczającą zdolności korekcyjnej kodu

Za prawidłową i kompletną odpowiedź, za zadanie zostanie przyznanych 7 pkt.,

za prawidłowe: wszystkie syndromy użyte w trakcie obliczeń, wektor błędu i nadane słowo kodowe,

za zadanie zostanie przyznanych 5 pkt.,

za prawidłowe: wszystkie syndromy użyte w trakcie obliczeń i wektor błędu, za zadanie zostaną przyznane 4 pkt.,

za prawidłowe: wszystkie syndromy użyte w trakcie obliczeń, za zadanie zostaną przyznane 3 pkt.

Tabela 5 Punktacja zadań testu nr 3

Numer zadania	Maksymalna liczba punktów
1	2
2	7
3	7
Suma	16