Polska-Brazylia 5:0, czyli o poprawianiu błędów w przekazywanych informacjach

Witold Tomaszewski

Instytut Matematyki
Politechniki Śląskiej
e-mail: Witold.Tomaszewski@polsl.pl

KATOWICE
2,00 zł cena gazety
5,99 zł cena gazety i płyty
ŚRODA
12 czerwca 2030
www.wyborcza.pl

gazeta
WYBORCZA.PL

Sensacja na Mundialu Tonga 2030. Droga do mistrzostwa otwarta!!!

POLSKA-BRAZYLIA 5:0

Polska reprezentacja pod wodzą Franciszka Smudy (82 l.)

Przykłady kodów wykrywających błędy

Kod ISBN

RUDOLF KIPPENHAHN

Kod ISBN

RUDOLF KIPPENHAHN

TAJEMNE PRZEKAZY
SZYFRY, ENIGMA I KARTY CHIPOWE

Kod ISBN

RUDOLF KIPPENHAHN

TAJEMNE PRZEKAZY
SZYFRY, ENIGMA I KARTY CHIPOWE

Na ścieżkach nauki
Wydawnictwo Prószyński i S-ka
Warszawa 2000

Kod ISBN

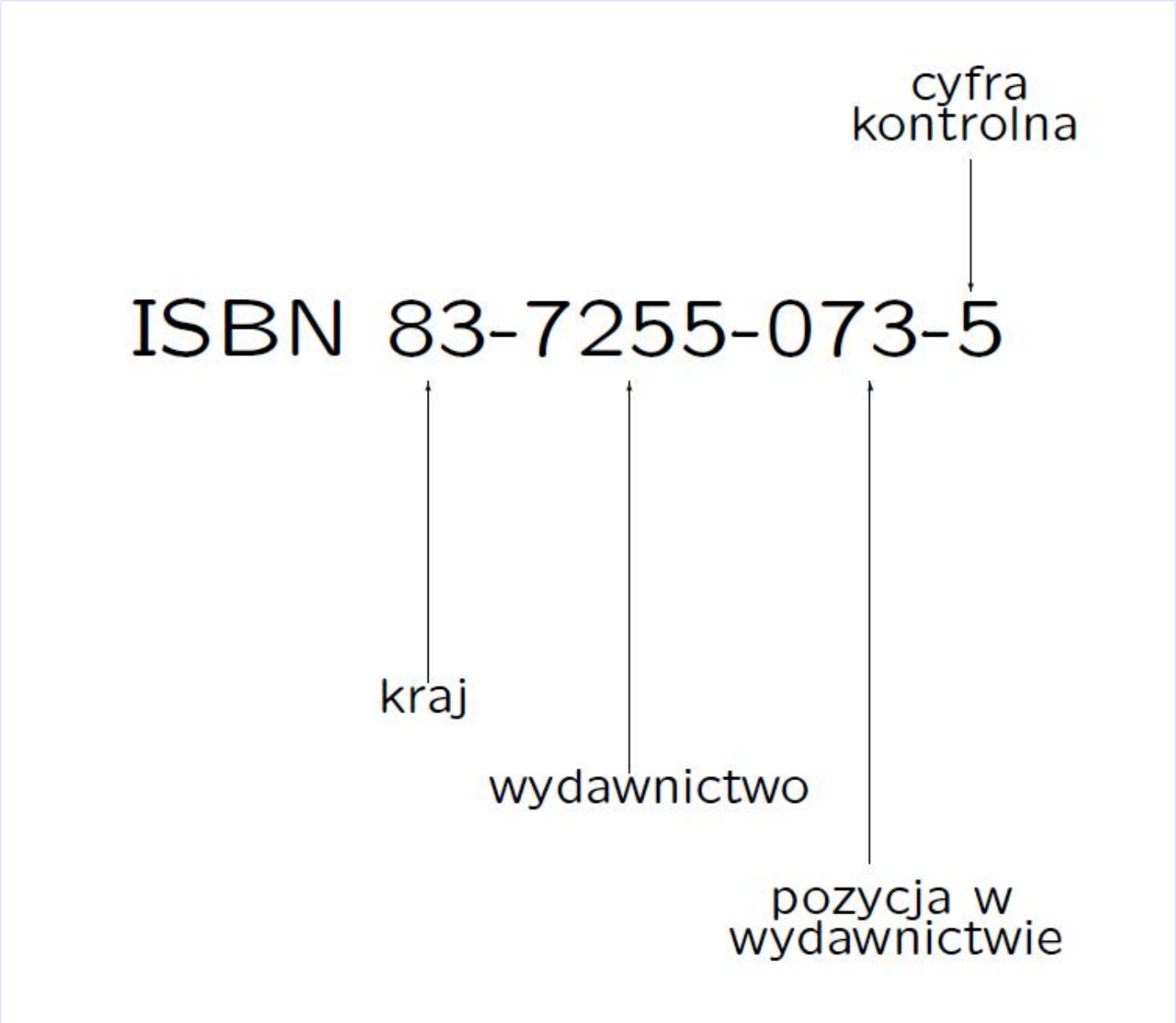
RUDOLF KIPPENHAHN

TAJEMNE PRZEKAZY
SZYFRY, ENIGMA I KARTY CHIPOWE

Na ścieżkach nauki
Wydawnictwo Prószyński i S-ka
Warszawa 2000

ISBN 83-7255-073-5

Kod ISBN


cyfra
kontrolna
ISBN 83-7255-073-5
kraj
wydawnictwo
pozycja w
wydawnictwie

Kontrola poprawności kodu ISBN

ISBN 83-7255-073-5

Kontrola poprawności kodu ISBN

ISBN 83-7255-073-5

$$10 \cdot \mathbf{8} + 9 \cdot \mathbf{3} + 8 \cdot \mathbf{7} + 7 \cdot \mathbf{2} + 6 \cdot \mathbf{5} + 5 \cdot \mathbf{5} + 4 \cdot \mathbf{0} + 3 \cdot \mathbf{7} + 2 \cdot \mathbf{3} + 1 \cdot \mathbf{5}$$

$$= 264 = 11 \cdot 24$$

Kontrola poprawności kodu ISBN

ISBN 83-7255-073-5

$$10 \cdot \mathbf{8} + 9 \cdot \mathbf{3} + 8 \cdot \mathbf{7} + 7 \cdot \mathbf{2} + 6 \cdot \mathbf{5} + 5 \cdot \mathbf{5} + 4 \cdot \mathbf{0} + 3 \cdot \mathbf{7} + 2 \cdot \mathbf{3} + 1 \cdot \mathbf{5}$$

$$= 264 = 11 \cdot 24$$

Każdy numer ISBN po zastosowaniu powyższej procedury daje wynik podzielny przez 11.

Algorytm podzielności przez 11

Kiedy liczba $x_1x_2x_3x_4 \ldots x_n$ jest podzielna przez 11?

Algorytm podzielności przez 11

Kiedy liczba $x_1x_2x_3x_4\ldots x_n$ jest podzielna przez 11?

Kryterium

Liczba naturalna $x_1x_2x_3x_4\ldots x_n$ (gdzie $x_1, x_2, x_3, x_4\ldots, x_n$ są jej cyframi przy zapisie dziesiętnym) jest podzielna przez 11 *dokładnie wtedy gdy wartość wyrażenia $x_1 - x_2 + x_3 - x_4 + \ldots$ (stawiamy między cyframi raz plus, raz minus) jest podzielna przez* 11*.*

Algorytm podzielności przez 11

Kiedy liczba $x_1x_2x_3x_4\ldots x_n$ jest podzielna przez 11?

Kryterium

Liczba naturalna $x_1x_2x_3x_4\ldots x_n$ (gdzie $x_1, x_2, x_3, x_4\ldots, x_n$ są jej cyframi przy zapisie dziesiętnym) jest podzielna przez 11 *dokładnie wtedy gdy wartość wyrażenia $x_1 - x_2 + x_3 - x_4 + \ldots$ (stawiamy między cyframi raz plus, raz minus) jest podzielna przez* 11*.*

Przykład Liczba 7256436 jest podzielna przez 11 bo $7 - 2 + 5 - 6 + 4 - 3 + 6 = 11$ jest podzielne przez 11. Rzeczywiście $7256436 : 11 = 659676$

Karty Visa

Każdy numer karty Visa złożony jest z 12 cyfr.

Karty Visa

Każdy numer karty Visa złożony jest z 12 cyfr.

Weźmy kartę Visa o numerze:

0 6 9 9 0 0 4 3 1 3 1 3 9 6 4 2

Karty Visa

Każdy numer karty Visa złożony jest z 12 cyfr.

Weźmy kartę Visa o numerze:

0	6	9	9	0	0	4	3	1	3	1	3	9	6	4	2

Pod spodem wypisujemy cyfry z nieparzystych pozycji:

0	6	9	9	0	0	4	3	1	3	1	3	9	6	4	2
	6		9		0		3		3		3		6		2

Kontrola poprawności numeru karty Visa

Cyfry z pozycji parzystych mnożymy przez dwa i wpisujemy kolejno w puste miejsca (jeśli przy mnożeniu otrzymamy liczbę większą od 9 to odejmujemy od niej 9).

Kontrola poprawności numeru karty Visa

Cyfry z pozycji parzystych mnożymy przez dwa i wpisujemy kolejno w puste miejsca (jeśli przy mnożeniu otrzymamy liczbę większą od 9 to odejmujemy od niej 9).

0	6	9	9	0	0	4	3	1	3	1	3	9	6	4	2
	6		9		0		3		3		3		6		2
0	6	9	9	0	0	8	3	2	3	2	3	9	6	8	2

Kontrola poprawności numeru karty Visa

Cyfry z pozycji parzystych mnożymy przez dwa i wpisujemy kolejno w puste miejsca (jeśli przy mnożeniu otrzymamy liczbę większą od 9 to odejmujemy od niej 9).

0	6	9	9	0	0	4	3	1	3	1	3	9	6	4	2
	6		9		0		3		3		3		6		2
0	6	9	9	0	0	8	3	2	3	2	3	9	6	8	2

Tak otrzymane cyfry dodajemy do siebie:

$$0+6+9+9+0+0+8+3+2+3+2+3+9+6+8+2$$

I sprawdzamy czy wynik jest podzielny przez 10. Jeśli tak jest to numer karty jest poprawny. W naszym przypadku wynik jest równy 70.

Kod kontroli parzystości

Rozważmy wszystkie binarne ciągi ośmiobitowe. To znaczy wszystkie ciagi $x_1x_2x_3x_4x_5x_6x_7x_8$, dla których każda pozycja x_i jest równa 1 lub 0 (jest ich $2^8 = 256$).

Kod kontroli parzystości

Rozważmy wszystkie binarne ciągi ośmiobitowe. To znaczy wszystkie ciagi $x_1x_2x_3x_4x_5x_6x_7x_8$, dla których każda pozycja x_i jest równa 1 lub 0 (jest ich $2^8 = 256$).

Niech C będzie zbiorem tych wszystkich ciągów, które mają w swoim zapisie parzystą liczbę 1. Na przykład 01111111 nie należy do C, a 11111111 należy.

Kod kontroli parzystości

Rozważmy wszystkie binarne ciągi ośmiobitowe. To znaczy wszystkie ciagi $x_1x_2x_3x_4x_5x_6x_7x_8$, dla których każda pozycja x_i jest równa 1 lub 0 (jest ich $2^8 = 256$).

Niech C będzie zbiorem tych wszystkich ciągów, które mają w swoim zapisie parzystą liczbę 1. Na przykład 01111111 nie należy do C, a 11111111 należy.

Uznajemy ciągi należące do C za prawidłowe, a te które do C nie należą za nieprawidłowe.

Kod kontroli parzystości

Rozważmy wszystkie binarne ciągi ośmiobitowe. To znaczy wszystkie ciagi $x_1x_2x_3x_4x_5x_6x_7x_8$, dla których każda pozycja x_i jest równa 1 lub 0 (jest ich $2^8 = 256$).

Niech C będzie zbiorem tych wszystkich ciągów, które mają w swoim zapisie parzystą liczbę 1. Na przykład 01111111 nie należy do C, a 11111111 należy.

Uznajemy ciągi należące do C za prawidłowe, a te które do C nie należą za nieprawidłowe.

C nazywamy kodem kontroli parzystości. Kod ten jest wykorzystywany do testowania pamięci komputera.

Kod kontroli parzystości

Rozważmy wszystkie binarne ciągi ośmiobitowe. To znaczy wszystkie ciagi $x_1x_2x_3x_4x_5x_6x_7x_8$, dla których każda pozycja x_i jest równa 1 lub 0 (jest ich $2^8 = 256$).

Niech C będzie zbiorem tych wszystkich ciągów, które mają w swoim zapisie parzystą liczbę 1. Na przykład 01111111 nie należy do C, a 11111111 należy.

Uznajemy ciągi należące do C za prawidłowe, a te które do C nie należą za nieprawidłowe.

C nazywamy kodem kontroli parzystości. Kod ten jest wykorzystywany do testowania pamięci komputera.

Kod ten wykrywa błędy pojedyńcze.

Przykłady kodów korekcyjnych

Kod powtórzeniowy

Rozważmy wszystkie trzybitowe ciągi binarne
$000, 001, 010, 011, 100, 101, 110, 111$

Kod powtórzeniowy

Rozważmy wszystkie trzybitowe ciągi binarne
$000, 001, 010, 011, 100, 101, 110, 111$

Umówmy się teraz, że nadawca wysyła tylko dwa ciągi 000 i 111.

Kod powtórzeniowy

Rozważmy wszystkie trzybitowe ciągi binarne
$000, 001, 010, 011, 100, 101, 110, 111$

Umówmy się teraz, że nadawca wysyła tylko dwa ciągi 000 i 111.

Możemy to rozumieć tak, że podstawowymi znakami wysyłanymi są 0 i 1. Aby poprawić niezawodność przekazu nadawca zamiast wysyłać do odbiorcy 0 to wysyła 000, a zamiast wysyłać 1 wysyła 111.

Kod powtórzeniowy

Rozważmy wszystkie trzybitowe ciągi binarne
$000, 001, 010, 011, 100, 101, 110, 111$

Umówmy się teraz, że nadawca wysyła tylko dwa ciągi 000 i 111.

Możemy to rozumieć tak, że podstawowymi znakami wysyłanymi są 0 i 1. Aby poprawić niezawodność przekazu nadawca zamiast wysyłać do odbiorcy 0 to wysyła 000, a zamiast wysyłać 1 wysyła 111.

Ponieważ w trakcie transmisji mogło dojść do przekłamań na każdej pozycji to odbiorca może otrzymać każdy trzybitowy ciąg.

Przykład korekcji

$C = \{000, 111\}$.

Przykład korekcji

$C = \{000, 111\}$.

Przypuśćmy, że odborca otrzymał wiadomość 101.

Przykład korekcji

$C = \{000, 111\}$.

Przypuśćmy, że odborca otrzymał wiadomość 101.

Wie, że ona jest błędna ale zauważa, że ciąg ten różni się od 000 dwoma pozycjami, a od 111 tylko jedną.

Przykład korekcji

$C = \{000, 111\}$.

Przypuśćmy, że odborca otrzymał wiadomość 101.

Wie, że ona jest błędna ale zauważa, że ciąg ten różni się od 000 dwoma pozycjami, a od 111 tylko jedną.

Zakładając, że rzadziej popełnia się dwa błędy niż jeden może stwierdzić (z przekonaniem graniczącym z pewnością), że nadanym ciągiem był 111.

Ogólna zasada korekcji

Możemy teraz trzybitowe ciągi binarne podzielić na dwa „obozy”:

000	111
001	011
010	101
100	110

Ogólna zasada korekcji

Możemy teraz trzybitowe ciągi binarne podzielić na dwa „obozy”:

000	111
001	011
010	101
100	110

Obóz czerwony to ciągi „bliżej” położone do 000, a niebieski ciągi „bliżej” położone do 111. Zatem ciągi czerwone będziemy odczytywać jako 000, a niebieskie jako 111.

Ogólna zasada korekcji

Możemy teraz trzybitowe ciągi binarne podzielić na dwa „obozy”:

000	111
001	011
010	101
100	110

Obóz czerwony to ciągi „bliżej” położone do 000, a niebieski ciągi „bliżej” położone do 111. Zatem ciągi czerwone będziemy odczytywać jako 000, a niebieskie jako 111.

Tak przyjęta zasada nazywana jest **Zasadą Największej Wiarygodności**

Ogólna zasada korekcji

Możemy teraz trzybitowe ciągi binarne podzielić na dwa „obozy”:

000	111
001	011
010	101
100	110

Obóz czerwony to ciągi „bliżej” położone do 000, a niebieski ciągi „bliżej” położone do 111. Zatem ciągi czerwone będziemy odczytywać jako 000, a niebieskie jako 111.

Tak przyjęta zasada nazywana jest **Zasadą Największej Wiarygodności**

Kod ten wykrywa błędy podwójne, koryguje błędy pojedyńcze.

Zasada Największej Wiarygodności

Ustalmy liczbę n. Rozważmy zbiór wszystkich n-bitowych ciągów binarnych. Oznaczymy go przez $\{0,1\}^n$.

Zasada Największej Wiarygodności

Ustalmy liczbę n. Rozważmy zbiór wszystkich n-bitowych ciągów binarnych. Oznaczymy go przez $\{0,1\}^n$.

Niech C będzie niepustym podzbiorem tego zbioru.

Zasada Największej Wiarygodności

Ustalmy liczbę n. Rozważmy zbiór wszystkich n-bitowych ciągów binarnych. Oznaczymy go przez $\{0,1\}^n$.

Niech C będzie niepustym podzbiorem tego zbioru.

Nadawca nadaje do odbiorcy tylko ciągi ze zbioru C.

Zasada Największej Wiarygodności

Ustalmy liczbę n. Rozważmy zbiór wszystkich n-bitowych ciągów binarnych. Oznaczymy go przez $\{0,1\}^n$.

Niech C będzie niepustym podzbiorem tego zbioru.

Nadawca nadaje do odbiorcy tylko ciągi ze zbioru C.

Odbiorca może otrzymać dowolny ciąg n-bitowy.

Zasada Największej Wiarygodności

Ustalmy liczbę n. Rozważmy zbiór wszystkich n-bitowych ciągów binarnych. Oznaczymy go przez $\{0,1\}^n$.

Niech C będzie niepustym podzbiorem tego zbioru.

Nadawca nadaje do odbiorcy tylko ciągi ze zbioru C.

Odbiorca może otrzymać dowolny ciąg n-bitowy.

Jeśli otrzyma ciąg y i ten ciąg należy do zbioru C to przyjmuje, że y został prawidłowo odebrany.

Zasada Największej Wiarygodności

Ustalmy liczbę n. Rozważmy zbiór wszystkich n-bitowych ciągów binarnych. Oznaczymy go przez $\{0,1\}^n$.

Niech C będzie niepustym podzbiorem tego zbioru.

Nadawca nadaje do odbiorcy tylko ciągi ze zbioru C.

Odbiorca może otrzymać dowolny ciąg n-bitowy.

Jeśli otrzyma ciąg y i ten ciąg należy do zbioru C to przyjmuje, że y został prawidłowo odebrany.

Jeśli y nie należy do C to (odbiorca) szuka element c ze zbioru C, który różni się o najmniejszą liczbę pozycji od y.

Zasada Największej Wiarygodności

Ustalmy liczbę n. Rozważmy zbiór wszystkich n-bitowych ciągów binarnych. Oznaczymy go przez $\{0,1\}^n$.

Niech C będzie niepustym podzbiorem tego zbioru.

Nadawca nadaje do odbiorcy tylko ciągi ze zbioru C.

Odbiorca może otrzymać dowolny ciąg n-bitowy.

Jeśli otrzyma ciąg y i ten ciąg należy do zbioru C to przyjmuje, że y został prawidłowo odebrany.

Jeśli y nie należy do C to (odbiorca) szuka element c ze zbioru C, który różni się o najmniejszą liczbę pozycji od y.

Jeśli takie c jest jedyne to odbiorca uznaje, że c zostało nadane (koryguje y na c).

Przykład

Rozważmy zbiór czterobitowych ciągów binarnych.

Przykład

Rozważmy zbiór czterobitowych ciągów binarnych.

Niech $C = \{0000, 1111\}$

Przykład

Rozważmy zbiór czterobitowych ciągów binarnych.

Niech $C = \{0000, 1111\}$

Jeśli odbiorca otrzymał 0001 to może przyjąć, że nadany został 0000 bo 0001 różni się o jedną pozycję od 0000 i aż o trzy pozycje od 1111.

Przykład

Rozważmy zbiór czterobitowych ciągów binarnych.

Niech $C = \{0000, 1111\}$

Jeśli odbiorca otrzymał 0001 to może przyjąć, że nadany został 0000 bo 0001 różni się o jedną pozycję od 0000 i aż o trzy pozycje od 1111.

Jeśli otrzymał jednak 0011 to ma problem bo ciąg ten różni się od dwie pozycje i od 0000 i od 1111. Musi więc zrezygnować (lub przyjąć jakąś inną zasadę).

Kody Hamminga

Działania modulo 2

W zbiorze $\{0, 1\}$ możemy zdefiniować działania dodawania i mnożenia modulo 2:

$$0 \underset{2}{+} 0 = 0 \quad 0 \underset{2}{+} 1 = 1$$
$$1 \underset{2}{+} 0 = 1 \quad 1 \underset{2}{+} 1 = 0$$

$$0 \underset{2}{\cdot} 0 = 0 \quad 0 \underset{2}{\cdot} 1 = 1$$
$$1 \underset{2}{\cdot} 0 = 1 \quad 1 \underset{2}{\cdot} 1 = 0$$

Macierz

Wypiszmy w kolumnach wszystkie niezerowe trzybitowe ciągi binarne:

Macierz

Wypiszmy w kolumnach wszystkie niezerowe trzybitowe ciągi binarne:

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

Mnożenie macierzy

Definiujemy mnożenie:

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} =$$

$$\begin{bmatrix} 0 \cdot x_1 + 0 \cdot x_2 + 0 \cdot x_3 + 1 \cdot x_4 + 1 \cdot x_5 + 1 \cdot x_6 + 1 \cdot x_7 \\ 0 \cdot x_1 + 1 \cdot x_2 + 1 \cdot x_3 + 0 \cdot x_4 + 0 \cdot x_5 + 1 \cdot x_6 + 1 \cdot x_7 \\ 1 \cdot x_1 + 0 \cdot x_2 + 1 \cdot x_3 + 0 \cdot x_4 + 1 \cdot x_5 + 0 \cdot x_6 + 1 \cdot x_7 \end{bmatrix} =$$

$$\begin{bmatrix} x_4 + x_5 + x_6 + x_7 \\ x_2 + x_3 + x_6 + x_7 \\ x_1 + x_3 + x_5 + x_6 + x_7 \end{bmatrix}$$

Mnożenie macierzy

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} x_4 + x_5 + x_6 + x_7 \\ x_2 + x_3 + x_6 + x_7 \\ x_1 + x_3 + x_5 + x_6 + x_7 \end{bmatrix}$$

Mnożenie macierzy

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} x_4 + x_5 + x_6 + x_7 \\ x_2 + x_3 + x_6 + x_7 \\ x_1 + x_3 + x_5 + x_6 + x_7 \end{bmatrix}$$

Działania wykonujemy oczywiście modulo 2.

Kod Hamminga

Za prawidłowe (wysyłane) uznajemy te ciągi siedmiobitowe $x_1x_2x_3x_4x_5x_6x_7$, dla których

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Kod Hamminga

Za prawidłowe (wysyłane) uznajemy te ciągi siedmiobitowe $x_1x_2x_3x_4x_5x_6x_7$, dla których

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

Na przykład 0000000, 1111111 są prawidłowe, a 1111110 nie

Algorytm korekcji

1. Otrzymujemy ciąg $y = y_1 y_2 y_3 y_4 y_5 y_6 y_7$.

Algorytm korekcji

1. Otrzymujemy ciąg $y = y_1y_2y_3y_4y_5y_6y_7$.
2. Obliczamy

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

Algorytm korekcji

1. Otrzymujemy ciąg $y = y_1y_2y_3y_4y_5y_6y_7$.
2. Obliczamy

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

3. Jeśli $s_1s_2s_3 = 000$ to y został prawidłowo odebrany.

Algorytm korekcji

1. Otrzymujemy ciąg $y = y_1y_2y_3y_4y_5y_6y_7$.
2. Obliczamy

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \\ y_5 \\ y_6 \\ y_7 \end{bmatrix} = \begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$$

3. Jeśli $s_1s_2s_3 = 000$ to y został prawidłowo odebrany.
4. Jeśli $s_1s_2s_3 \neq 000$ to $\begin{bmatrix} s_1 \\ s_2 \\ s_3 \end{bmatrix}$ jest jedną z kolumn macierzy H.

   Szukamy numeru tej kolumny i y korygujemy zmieniając znak na pozycji odpowiadającej temu numerowi (to znaczy zmieniamy 0 na 1 i 1 na 0).

Algorytm korekcji

1. Otrzymujemy ciąg $y = 1111000$.

Algorytm korekcji

1. Otrzymujemy ciąg $y = 1111000$.
2. Obliczamy

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

Algorytm korekcji

1. Otrzymujemy ciąg $y = 1111000$.
2. Obliczamy

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

3. Ponieważ to co otrzymaliśmy jest czwartą kolumną macierzy H to w wektorze odebranym 1111000 zmieniamy czwartą pozycję otrzymując 1110000.