Rozdział 1. Kody i kodowanie w systemach cyfrowych - Zadania

Zadanie 1.1

Przedstawić liczby naturalne 0, 5, 16, 121, 250 (zapisane w naturalnym zapisie dziesiętnym) w naturalnych zapisach wagowych o innych podstawach.

a) binarnym
b) trójkowym
c) oktalnym
d) szesnastkowym (hexadecymalnym)
e) przy podstawie 28
f) minusdwójkowym

Rozwiązanie: *przypadek a)*

```
0 (D) =  0
5(D) = 101
16(D) =10000
121(D) = 1111001
250 (D) = 11111010
```

itd.

Zadanie 1.2

Przedstawić liczbę

a) ABCDEF (zapisaną w zapisie hexadecymalnym) w zwykłym zapisie dziesiętnym
b) 10100111 (zapisaną w zapisie binarnym) w zwykłym zapisie dziesiętnym

Zadanie 1.3

Rozważmy zapis wagowy z wagą W=7. Znaleźć uzupełnienie do W i W-1 liczby 151 zapisanej w zapisie siódemkowym.

Zadanie 1.4

Podać uzupełnienie do 2 i uzupełnienie do 1 następujących liczb 8 bitowych:

```
a)  10101010
b)  11111111
c)  00000000
d)  11110000
e)  00001111
```

Zadanie 1.5

Zapisać liczby 128, 127,126,125, -128,-127,-126, 125 za pomocą 8 bitowego kodu a) NKB, b) U2, c) U1 d) moduł znak. Gdyby nie było to możliwe, odpowiedź uzasadnić.

Rozwiązanie: *przypadek a) - kod NKB*

```
128 (D) = 10000000
127 (D) = 01111111
126 (D) = 01111110
125 (D) = 01111101
```

Liczb całkowitych ujemnych z samej definicji kodu NKB nie zapisujemy.

przypadek b) kod U2

`128 (D)` nie można zapisać w 8 bitowym kodzie U2 (liczba 128 znajduje się poza zakresem liczb reprezentowanych)

```
 127 (D) = 01111111
 126 (D) = 01111110
 125 (D) = 01111101
-128 (D) = 10000000
-127 (D) = 10000001
-126 (D) = 10000010
-125 (D) = 10000011
```

Zadanie 1.6

Podać zakresy następujących zapisów liczbowych

a) 8 bitowego NKB, 16 bitowego NKB, 32 bitowego NKB, 64 bitowego NKB
b) 8 bitowego U2, 16 bitowego U2, 32 bitowego U2, 64 bitowego U2
c) 8 bitowego U1, 16 bitowego U1, 32 bitowego U1, 64 bitowego U1

Zadanie 1.7

Ilu musimy użyć bitów (czyli cyfr dwójkowych) do zapisania liczb całkowitych ze zbioru

a) $< 0, 2^{1024} >$
b) $< 0, 2^{1024} - 1 >$
c) $< 0, 10^{1024} >$
d) $< 0, 10^{1024} - 1 >$

(takich dużych liczb używamy np. w kryptografii).

Zadanie 1.8

Ilu musimy użyć bitów (czyli cyfr dwójkowych) do zapisania w kodzie U1 liczb

a) 2^{1024}
b) 10^{1024}

Zadanie 1.9

Ilu musimy użyć bitów (czyli cyfr dwójkowych) do zapisania liczb a) 2^{1024}, b) 10^{1024} w kodzie U2.

Zadanie 1.10

Kodujemy elementy zbioru n-elementowego A za pomocą ciągów k-bitowych . Znaleźć najmniejsze $k \in N$ takie, że istnieje odwzorowanie różnowartościowe $f : A \to \{0,1\}^k$ będące kodem.

Zadanie 1.11

Jak dodawanie dwóch 8 bitowych liczb ustawia znaczniki (CF – Carry Flag, OF - Overflow flag ,AF – Auxiliary Flag) w typowym 8 bitowym mikroprocesorze

```
a)    a = 11111111     i     b = 11111111
b)    a = 00001111     i     b = 00000001
c)    a = 11110000     i     b = 00001111
d)    a = 01111111     i     b = 01111111
e)    a = 10011111     i     b = 11110111
f)    a = 01111111     i     b = 11110111
g)    a = 11000011     i     b = 11100001
h)    a = 10000000     i     b = 11111111
i)    a = 01110111     i     b = 01111111
j)    a = 11111111     i     b = 00000001
```

Odp. a) CF=1, OF=0, AF=1,
b) CF=0, OF=0, AF=1

```
c) CF=0, OF=0, AF=0,
d) CF=0, OF=1, AF=1 itd.
```

Zadanie 1.12

Znaleźć takie 2 słowa binarne (bajty) a i b ,które dodane ustawiają w 8-bitowym mikroprocesorze

```
a)     CF=1, OF=0, AF=0
b)     CF=0, OF=0, AF=1
c)     CF=0, OF=1, AF=0
d)     CF=0, OF=1, AF=1
e)     CF=1, OF=0, AF=0
f)     CF=1, OF=0, AF=1
g)     CF=1, OF=1, AF=0
h)     CF=1, OF=1, AF=1
```

Odp.

```
a)    a = 00000000     i     b = 11111111
b)    a = 00001000     i     b = 11111111
c)    a = 01000000     i     b = 01000000
d)    a = 01001000     i     b = 01001000
e)    a = 11110000     i     b = 11110000
f)    a = 11111111     i     b = 11111111
g )   a = 10000000     i     b = 10000000
h)    a = 10001000     i     b = 10001000
```

Zadanie 1.13

Niech $W \in N, W \geq 2$. Pokazać, że liczba wymierna $\frac{m}{n}$, gdzie $m,n \in N$, $n \geq 2$ oraz $GCD(n,W) = 1$ i $GCD(n,m) = 1$, nie daje się zapisać w zapisie stałoprzecinkowym bez znaku przy podstawie W (i podobnie w zapisie stałoprzecinkowym moduł znak i stałoprzecinkowym uzupełnień do W). W szczególności liczby 1/3 ani 1/p (dla liczby pierwszej $p \geq 3$) nie można zapisać w stałoprzecinkowym zapisie binarnym.

Rozwiązanie:

Dowód przeprowadzimy nie wprost. Rozważmy najpierw zapis stałoprzecinkowy przy podstawie W. Załóżmy, że $\frac{m}{n}$ daje się przedstawić w zapisie stałoprzecinkowym przy podstawie W. Mamy wówczas dla pewnych $k, r \in N \cup \{0\}$ i $a_k, a_{k-1}, a_{k-2},..., a_0, a_{-1},..., a_{-r} \in \langle 0, W-1 \rangle$ (zakładamy, że operujemy słowem o długości $l = k + r + 1$)

$$\frac{m}{n} = a_k W^k + a_{k-1} W^{k-1} + ... + a_1 W + a_0 + a_{-1} W^{-1} + ... + a_{-r} W^{-r}.$$

Mnożąc obie strony powyższej równości przez nW^r dostajemy

$$mW^r = n(a_k W^{k+r} + a_{k-1} W^{(k-1)+r} + ... + a_{-r+1} W + a_{-r}).$$

Tak być jednak nie może, bo prawa strona powyższej równości jest podzielna przez n, a lewa nie. Dostaliśmy więc sprzeczność, a więc przy przyjętych założeniach o n, m i W liczba wymierna $\frac{m}{n}$ nie daje się przedstawić w zapisie stałoprzecinkowym przy podstawie W.

Rozumowania dla zapisu stałoprzecinkowego moduł znak z podstawą W i zapisu stałoprzecinkowego uzupełnień do W są analogiczne.

Zauważmy, że założenie, iż $m, n \in N$ można zastąpić założeniem, że $m, n \in Z$. Powyższy dowód trzeba w tej sytuacji nieznacznie zmodyfikować.

Zadanie 1.14

Napisać prosty program (w Pascalu lub C++ lub Javie lub Turbo Assemblerze) do konwersji liczby

a) ze zbioru $N \cup \{0\}$ zapisanej w zapisie dziesiętnym na liczbę w kodzie NKB

b) ze zbioru $N \cup \{0\}$ zapisanej w kodzie NKB na liczbę w zapisie dziesiętnym

Zadanie 1.15

Niech będą dane moduły zapisu resztowego $m_1 = 3,\ m_2 = 5,\ m_3 = 7$ i niech zakresem naszego zapisu będzie zbiór $< 0,104 >$.

a. niech będą dane dwie liczby w tym zapisie $x = (2,0,5)$ i $y = (0,3,3)$. Stosując algorytm dodawania i mnożenia dla zapisu resztowego znaleźć liczby $x + y$ oraz $x \cdot y$.

b. niech będą dane dwie liczby w tym zapisie $x = (2,0,5)$ i $y = (1,3,3)$. Stosując algorytm dodawania i mnożenia dla zapisu resztowego znaleźć liczby $x + y$ oraz $x \cdot y$.

Rozwiązanie:

a. Stosując algorytm dodawania i mnożenia dla zapisu resztowego dostajemy

$$x + y = (2,0,5) + (0,3,3) = (2 \oplus_3 0, 0 \oplus_5 3, 5 \oplus_7 3) = (2,3,1)$$
$$x \cdot y = (2,0,5) \cdot (0,3,3) = (2 \otimes_3 0, 0 \otimes_5 3, 5 \otimes_7 3) = (0,0,1)$$

Liczby $x = (2,0,5)$ i $y = (0,3,3)$ specjalnie w zadaniu zostały dobrane tak, by można było łatwo bez żmudnych obliczeń stwierdzić, jak te liczby wyglądają w zapisie dziesiętnym. Otóż łatwo sprawdzić, że w zapisie dziesiętnym x =5, y= 3, ich suma równa jest 8 a iloczyn 15 i oba wyniki mieszczą się w zadanym zakresie.

b. Stosując algorytm dodawania i mnożenia dla zapisu resztowego dostajemy

$$x + y = (2,0,5) + (1,3,3) = (2 \oplus_3 1, 0 \oplus_5 3, 5 \oplus_7 3) = (0,3,1)$$
$$x \cdot y = (2,0,5) \cdot (1,3,3) = (2 \otimes_3 1, 0 \otimes_5 3, 5 \otimes_7 3) = (2,0,1)$$

Tym razem x =5 oraz y= 73 w zapisie dziesiętnym. Ich suma 88 mieści się w zakresie liczb reprezentowanych w używanym zapisie resztowym, natomiast iloczyn 365 wykracza poza ten zakres.

Zadanie 1.16

Dodać i pomnożyć 2 liczby x=(4,6,12) i y=(3,5,11) w zapisie RNS (z modułami $m_1 = 5, m_2 = 7, m_3 = 13$) zgodnie z algorytmami dodawania i mnożenia w RNS. Zakładając, że reprezentujemy w rozważanym zapisie RNS liczby $\langle 0, m_1 \cdot m_2 \cdot m_3 - 1 \rangle$ sprawdzić poprawność tak uzyskiwanych wyników.

Rozwiązanie:

Stosujemy algorytm dodawania w zapisie resztowym i dostajemy

$a + b = (4,6,12) + (3,5,11) = (4 \oplus_5 3, 6 \oplus_7 5, 12 \oplus_{13} 11) = (7(\mathrm{mod}\,5), 11(\mathrm{mod}\,7), 23(\mathrm{mod}\,13)) = (2,4,10)$

2. Podobnie stosując algorytm mnożenia dla zapisu resztowego dostajemy
$a \cdot b = (4,6,12) \cdot (3,5,11) = (4 \otimes_5 3, 6 \otimes_7 5, 12 \otimes_{13} 11) = (12 (\mathrm{mod}\, 5), 30 (\mathrm{mod}\, 7), 132 (\mathrm{mod}\, 13)) = (2,2,2)$

Stosując algorytm konwersji z zapisu RNS na zapis dziesiętny stwierdzamy jednak, że ani suma ani iloczyn liczb x i y nie mieszczą się w zakresie reprezentowanych zapisem liczb. Nie ma w tym nic złego. Takie przygody (tzn. tzw. nadmiar) przydarzają się znacznie nobliwiej wyglądającym zapisom takim jak np. NKB o stałej długości słowa kodowego czy zapis U2 o stałej długości słowa kodowego. Po to właśnie (by kontrolować sytuacje patologiczne) umieszczane są w rejestrze znaczników mikroprocesorów znaczniki CF i OF.

Zadanie 1.17

Niech będą dane moduły zapisu resztowego $m_1 = 5,\ m_2 = 7,\ m_3 = 13,\ m_4 = 17,\ m_5 = 19$ oraz niech $x = (4,6,10,0,0)$ i $y = (3,5,11,0,0)$. Znaleźć liczby $x + y$ i $x \cdot y$ w zapisie resztowym wiedząc, że zakresem naszego zapisu jest zbiór $< 0, m_1 \cdot m_2 \cdot m_3 \cdot m_4 \cdot m_5 - 1 >$.

Rozwiązanie: $x + y = (2,4,8,0,0)$, $x \cdot y = (2,2,6,0,0)$

Zadanie 1.18

Niech będą dane moduły zapisu resztowego $m_1 = 3, m_2 = 5, m_3 = 7$ oraz niech $x = (2,0,5)$. Znaleźć liczbę x wiedząc, że należy ona do zbioru $< 0,104 >$.

Rozwiązanie:
Skorzystamy z następującego algorytmu konwersji. Niech $(a_1, a_2, ..., a_n)$ będzie zapisem *RNS* liczby x oraz $M = m_1 \cdot m_2 \cdot ... m_n$ ..Znajdujemy liczby $b_j \in Z$ takie, że:

$$b_j \cdot \frac{M}{m_j} \equiv 1 \ (\mathrm{mod}\, m_j) \ (\text{dla } j=1,2,...,n).$$

W naszym przypadku łatwo wyznaczamy stałe b_1, b_2, b_3 spełniające powyższy warunek i dostajemy $b_1 = 2$, $b_2 = 1$, $b_3 = 1$. .Jeśli $x \in [0, M]$ (tak jest w naszym przypadku) to

$$x = [a_1 \cdot b_1 \cdot \frac{M}{m_1} + a_2 \cdot b_2 \cdot \frac{M}{m_2} + ... + a_n \cdot b_n \cdot \frac{M}{m_n}]_M ,$$

Korzystając z powyższego wzoru otrzymujemy $x = 5$.

Zadanie 1.19

Mamy liczbę $a = (2,3,6,0)$ zapisaną w zapisie resztowym (czyli RNS) dla modułów $m_1 = 3$, $m_2 = 5$, $m_3 = 7$, $m_4 = 13$. Znaleźć liczbę a, tzn. przedstawić ją w zapisie dziesiętnym.

Zadanie 1.20

Ile jest wszystkich kodów BCD o długości słowa kodowego n. Jakie jest najmniejsze możliwe n. Ile jest wszystkich różnych 4-bitowych kodów BCD.

Rozwiązanie:
Kodem BCD o długości n nazywamy dowolne odwzorowanie różnowartościowe $f : \{0,1,2,...,9\} \to \{0,1\}^n$. Zbiór wartości funkcji f czyli zbiór $\{0,1\}^n$ ma 2^n elementów. By funkcja f była różnowartościowa n musi być większe od 4. Najmniejszym możliwym n jest więc 4. Kod BCD można więc traktować jako wariację 10 elementową ze zbioru 2^n-elementowego. Wszystkich kodów BCD jest więc tyle ile wszystkich wariacji

10-elementowych ze zbioru 2^n elementowego a więc $\frac{2^n!}{(2^n-10)!}$. W najbardziej typowym przypadku n=4 mamy więc $\frac{2^4!}{(2^4-10)!}=7\cdot 8\cdot 9\cdot 10\cdot 11\cdot 12\cdot ...\cdot 16 \geq 10^8$. W praktyce używamy jedynie kilku kodów BCD np. 8421, excess 3.

Zadanie 1.21

Zapisać w kodzie BCD 8421 liczby

a) 8421.

b) 95

c) 88

Zadanie 1.22

Załóżmy, że m i n są liczbami naturalnymi oraz $m<n$. Ile jest wszystkich kodów m z n (w systemach cyfrowych stosowane są np. kody 2 z 5 i 1 z n).

Rozwiązanie:

Kody m z n są kodami numerycznymi o stałej długości n charakteryzujące się stałą liczbą m jedynek na n pozycjach w słowie kodowym. Przy ustalonym m i n liczba takich słów kodowych jest równa

$$\binom{n}{m}=\frac{n!}{m!(n-m)!}.$$

Jeśli zbiór obiektów kodowanych ma k elementów, gdzie $k \leq \binom{n}{m}$, to różnych kodów m z n jest

$$\frac{\binom{n}{m}!}{(\binom{n}{m}-k)!}.$$

Zadanie 1.23

Wykazać, że w naturalnym zapisie pozycyjnym przy podstawie W (gdzie $W \in N, W \geq 2$) liczba cyfr $l(m)$ użyta do zapisu liczby $m \in N$ jest równa $\lfloor \log_W m \rfloor + 1$ czyli

$$l(m)=\lfloor \log_W m \rfloor + 1 \qquad (*)$$

gdzie $\lfloor \cdot \rfloor : R \to R$ jest tzw. funkcją podłogi.

Rozwiązanie:

Dowolna liczba $m \in N$ daje się jednoznacznie zapisać dla ustalonego $W \in N, W \geq 2$ w postaci

$$m = a_n W^n + a_{n-1}W^{n-1} + ... + a_1 W + a_0$$

gdzie $a_1 \in <0, W-1>$ dla $i = 0,1,...,n$ oraz $a_n \neq 0$.

Powyższy fakt umożliwia nam określenie „naturalnego zapisu pozycyjnego przy podstawie W" jako odwzorowania

$$f : N \cup \{0\} \ni m \to a_n, a_{n-1}, ..., a_0 \in \langle 0, W-1 \rangle *$$

przy czym $f(0) \overset{df}{=} 0 \in \langle 0, W-1 \rangle *$.

Łatwo zauważyć, że tak zdefiniowana funkcja f jest funkcją równowartościową. Zatem funkcja $l : N \cup \{0\} \ni m \to l(m) \in N$ jest dobrze określoną funkcją (monotoniczną) podającą dla każdej liczby m ze zbioru $N \cup \{0\}$ jej długość w zapisie pozycyjnym przy podstawie W.

Najprostszy, ale pracochłonny sposób obliczenia $l(m)$ polega po prostu na znalezieniu słowa $f(m) = a_n, a_{n-1}, ..., a_0 \in \langle 0, W-1\rangle *$ (tzw. rozwinięcia liczby m przy podstawie W) i obliczeniu jego długości. Metodę postępowania (dzielenie przez W i branie jako kolejnej cyfry rozwinięcia reszty z dzielenia) opisuje szczegółowo algorytm konwersji. Dla dużych m powyższy sposób staje się jednak kłopotliwy i wzór postaci (*) byłby bardzo użyteczny.

Widać jednak od razu, że liczba $m = W^n$ dla ustalonego $n \in N \cup \{0\}$ ma długość $n+1$, a więc $l(m) = \lfloor \log_W m \rfloor + 1$, czyli wzór (*) jest dla takich m spełniony. Jeśli $m = W^{n+1} - 1$, to również łatwo zauważyć, że $l(m) = n+1$. Wzór (*) jest prawdziwy również w tym przypadku, mamy bowiem dla $m = W^{n+1}$, $\log_W m = n+1$ i z uwagi na ścisłą monotoniczność funkcji logarytmicznej $\log_W : R^+ \setminus \{0\} \ni x \to \log_W x \in R)$ mamy dla $m < W^{n+1}$, $\log_W m < n+1$ i wartość funkcji podłogi $\lfloor \log_W m \rfloor \le n$ zatem dla $m = W^n$ i $m = W^{n+1} - 1$ wzór (*) zachodzi dając tę samą wartosć n+1. Z uwagi na monotoniczność funkcji l i monotoniczność funkcji $m \to \lfloor \log_W m \rfloor$ dla każdego $m \in < W^n, W^{n+1} - 1 >$ wzór (*) jest prawdziwy.

Zadanie 1.24

Ilu bitów potrzeba do zapisania liczb naturalnych a) b) c) w kodzie NKB. Sprawdzić, czy podane liczby dają się zapisać w kodzie NKB o długości słowa kodowego 16 (2 bajty).

a) 59
b) 1025
c) 2^{1024}

Rozwiązanie:
Korzystając z wyników i oznaczeń zadania 1 (wzór $l(m) = \lfloor \log_W m \rfloor + 1$) dostajemy dla W=2

a) $l(59) = \lfloor \log_2 59 \rfloor + 1 = 5 + 1 = 6$

b) $l(1025) = \lfloor \log_2 1025 \rfloor + 1 = 10 + 1 = 11$

c) $l(2^{1024}) = \lfloor \log_2 2^{1024} \rfloor + 1 = 1024 + 1 = 1025$

Zadanie 1.25

Załóżmy, że używamy alfabetu angielskiego i umówmy się, że używamy tylko małych liter: *a,b,c,...,z* (w sumie mamy więc 26 liter). Oznaczmy ten alfabet przez V. W kryptografii często utożsamiamy tekst szyfrowany (czyli pewne słowo nad alfabetem *V)* z liczbą traktując słowo nad alfabetem V jako zapis liczb naturalnych w systemie pozycyjnym (ściślej jako zapis liczb ze zbioru $N \cup \{0\}$).

a) Jaka jest podstawa W tego zapisu pozycyjnego?
b) Jaką maksymalnie liczbę naturalną możemy zapisać słowem o długości 100?
c) Zapisać w rozważanym zapisie liczbę n=10293 jeśli przyjmiemy, że a odpowiada 0, b odpowiada 1, c odpowiada 2, itd. ...,a z odpowiada 25.
d) Ile to jest: *abrakadabra* ?

Rozwiązanie:

a) Podstawa W zapisu pozycyjnego jest równa liczbie liter alfabetu V a więc W=26.

b) Jeśli przyjmiemy, że a odpowiada 0, b odpowiada 1, c odpowiada 2,...,a z odpowiada 25, to maksymalna liczba naturalna, jaką możemy zapisać słowem o długości 100, to liczba

$$\underbrace{zzz...zz}_{100} = 26^{100} - 1$$

c) Stosując algorytm przedstawiania liczby ze zbioru $N \cup \{0\}$ w zapisie wagowym o podstawie (wadze) W dostajemy

$10293 : 26 = 395 \cdot 26 + 23$, $395 : 26 = 15 \cdot 26 + 5$, $15 : 26 = 0 \cdot 26 + 15$

zatem 10293 (w zapisie dziesiętnym) = *pfx* (w zapisie przy podstawie W=26)

d) *abrakadabra* =

$$0 \cdot W^{10} + 1 \cdot W^{9} + 17 \cdot W^{8} + 0 \cdot W^{7} + 10 \cdot W^{6} + 0 \cdot W^{5} + 3 \cdot W^{4} + 0 \cdot W^{3} + 1 \cdot W^{2} + 17 \cdot W^{1} + 0 \cdot W^{0} =$$

$$= 1 \cdot 26^{9} + 17 \cdot 26^{8} + 10 \cdot 26^{6} + 3 \cdot 26^{4} + 1 \cdot 26^{2} + 17 \cdot 26^{1}$$

Zadanie 1.26

Załóżmy, że wiadomość jawna i szyfrogram zapisujemy w 26 literowym alfabecie angielskim złożonym z 26 liter. Wiemy, że *grgsgquzg* jest tekstem zaszyfrowanym za pomocą szyfru Cezara, ale nie znamy klucza k (k jest liczbą pozycji o które następuje przesunięcie w prawo w alfabecie). Znaleźć klucz k i tekst jawny wiedząc, że tekst jawny odpowiadający powyższemu tekstowi zaszyfrowanemu jest sensownym zdaniem dotyczącym zwierząt

Rozwiązanie:

Przestrzeń kluczy dla szyfru Cezara jest bardzo mała. Mamy tylko 25 kluczy $\{1,2,3,...,25\}$. Zastosujemy tzw. atak brutalny czyli przeglądanie po kolei zbioru kluczy. Sprawdzamy czy dla klucza k=1,2,...,25 uzyskamy wiadomość sensowną. Już dla k=6 uzyskujemy wiadomość sensowną dotyczącą zwierząt: *alamakota* . Oczywiście dla długich tekstów procedurę przeglądania przestrzeni kluczy i rozszyfrowywania możemy zautomatyzować pisząc odpowiedni program.

Zadanie 1.27

W typowym kryptograficznym dzielimy tekst jawny (czyli z matematycznego punktu widzenia pewne słowo m nad ustalonym alfabetem V) na tzw. jednostki tekstu, czyli słowa o ustalonej długości, po czym przyporządkowujemy każdej jednostce tekstu liczbę i następnie dopiero przekształcamy ją funkcją szyfrującą. Do przekształcania jednostki tekstu o długości $r \geq 1$ w liczbę można użyć dowolnej funkcji różnowartościowej $f : V^r \to N \cup \{0\}$ np.: można potraktować napis $a_{r-1}, a_{r-2}...a_0 \in V^r$ jako liczbę w zapisie wagowym o wadze $W = \mathrm{card}\, V$. Umawiamy się, że taką funkcję zastosujemy w zadaniu.

a) przyporządkować jednostce tekstu „pies” liczbę $n \in N \cup \{0\}$ przyjmując, że tekst jawny zapisywany jest w 26 literowym alfabecie angielskim złożonym z samych małych liter, do którego dołączono spację (w sumie mamy więc 27 symboli). Przyjmujemy również następującą umowę:

spacja = 0
a = 1
b = 2
c = 3
.
.
.
z = 26

Nasz alfabet jest więc taki $V = \{spacja, a, b, c, ..., z\}$ i card $V = 27$

b) przyporządkować liczbie 8444 (w zapisie dziesiętnym) odpowiadające jej słowo nad alfabetem V.

c) przyjmijmy, ze kryptosystem działa na liczbach z ciała F_q (ciało skończone o q elementach). Znaleźć największą dopuszczalną długość jednostki tekstu.

Rozwiązanie:

a) Obliczamy szukaną liczbę n traktując słowo „pies" jako naturalny zapis wagowy liczby n (z wagą W = 27).

$$\text{pies} = 19 + 5 \cdot 27 + 9 \cdot 27^2 + 16 \cdot 27^3 = 321643$$

Przy dłuższych słowach warto zastosować do powyższych obliczeń schemat Hornera będący sposobem obliczenia wartości wielomianu w punkcie, oparty na wzorze:

$$a_r x^r + a_{r-1} x^{r-1} + ... + a_0 = (...(((a_r x + a_{r-1})x + a_{r-2})x + +a_{r-3})...)x + a_0$$

b) Stosując zwykły algorytm zapisu liczby w zapisie naturalnym wagowym z wagą W uzyskamy 8444 (zapis dziesiętny) = kot (zapis z wagą $W = 27$). Poszukiwanym słowem jest więc słowo „kot".

Uwaga: Algorytm konwersji liczby $n \in N \cup \{0\}$ ma zapis naturalny wagowy z wagą W, tzn. sposób obliczenia słowa $a_{r-1}a_r...a_0 \in V^r$, polega na wykonaniu wielokrotnego dzielenia liczby n przez wagę W i braniu jako kolejnych cyfr reszt z dzielenia. Dokładniej:

Wejście: n i W

Wyjście: $a_{r-1}a_{r-2}...a_0 \in V^r$

$m_0 = n;$

for $i = 0$ to $i = r-1$ **do begin**

podziel m_1 przez W uzyskując m_{i+1} i resztę a_i, czyli przedstaw m_1 w postaci

$m_1 = m_{i+1} \cdot W + a_i$, gdzie $0 \le a_i < W$

end

Jest to algorytm dla przypadku, gdy wiemy, że liczba n zmieści się na r pozycjach. Jeśli nie mamy tej informacji, to algorytm konwersji musi mieć regułą stopu. Najprościej ją zrealizować badając czy $m_1 = 0$, jeśli tak zatrzymujemy obliczenia.

Uwaga: Możemy też zdefiniować funkcję $f : V^r \to N \cup \{0\}$ postępując tak. Ciąg r znaków ASCII (8 bitów na znak) zestawiamy w słowo binarne $8 \cdot r$ bitowe i traktujemy to słowo jako liczbę w kodzie NKB (naturalny kod binarny). Metoda taka jest prostsza, ale z reguły ten sam tekst prowadzi do liczby o większej liczbie bitów.

Jeśli jednostka tekstu ma długość r, to maksymalna liczba jaką możemy zapisać w naturalnym kodzie wagowym z wagą W jest równa $W^r - 1$. Oczywiście musi być przy tym spełniona nierówność $W^r - 1 \le q$, zatem maksymalna dopuszczalna długość jednostki tekstu jest równa $\max\{r \in N; W^r - 1 \le q\}$ lub inaczej $r_{\max} = \lfloor \log_W (q-1) \rfloor$.

Zadanie 1.28

W systemie operacyjnym Linux (w dystrybucji Linuxa o nazwie Red Hat 6.0 Hewig) można stosować hasła użytkowników systemu o długości do 256 znaków. Ile różnych haseł można używać w tym systemie.

Rozwiązanie:

Załóżmy, że hasła tworzymy jako słowa nad alfabetem V zawierającym W symboli. Istnieje dokładnie W^k różnych haseł o długości k (czyli słów) k literowych nad alfabetem V zatem liczba wszystkich możliwych haseł łącznie z hasłem pustym wynosi

$$1+W+W^2+W^3+\ldots+W^{256}=\frac{W^{257}-1}{W-1}$$

Zadanie 1.29

Załóżmy, że zapisujemy wiadomości jawne w W literowym alfabecie. Jaka jest maksymalna długość jednostki tekstu, jeśli jednostkę tekstu zamieniamy w przyjętym systemie kryptograficznym na:

a) liczbę z grupy multiplikatywnej F_q^* ciała F_q (tak jest np. w systemie kryptograficznym EL Gamala).

b) liczbę z pierścienia Z_n, gdzie $n=p\cdot q$ i $p,\ q$ są różnymi liczbami pierwszymi (tak jest w systemie kryptograficznym RSA).

Rozwiązanie:

a) Ilość różnych słów o długości k nad alfabetem W literowym V jest równa W^k (jest to ilość k elementowych wariacji z powtórzeniami ze zbioru W elementowego lub co na jedno wychodzi, liczba elementów zbioru V^k). Odwzorowanie szyfrujące $f:V^k\to F_q^*$ przyporządkowujące jednostce tekstu liczbę, musi być różnowartościowe, zatem musimy mieć $W^k\le q-1$, ponieważ liczba elementów grupy multiplikatywnej F_q^* jest równa q-1. Ostatecznie więc $k\le\log_W(q-1)$. Jeśli np. liczba q elementów ciała jest równa 2^{400}, a $W=32$, to mamy $\log_{32}2^{400}=\log_{32}32^{80}=80$.

b) W przypadku pierścienia Z_n rozumujemy analogicznie otrzymując warunek $W^k\le n$, co daje $k\le\log_W n$.

Zadanie 1.30

Zaszyfrować klasycznym szyfrem Cezara wiadomość jawną

m= this cipher is certainly not secure

zapisaną w 26 literowym alfabecie angielskim złożonym z małych liter.

Rozwiązanie:

Szyfr Cezara jest szyfrem podstawieniowym. Zgodnie z definicją klasycznego szyfru Cezara każdą literę wiadomości jawnej m zastępujemy literą położoną o 3 pozycje w prawo (modulo 26) przy zwykłym uporządkowaniu 26 literowego alfabetu (por. wstęp teoretyczny); zatem kryptogram c wiadomości m będzie następujący:

c=wklvf lskhu lvfhu wdlqo bqrwv hfxuh

Zadanie 1.31

Niech V będzie dowolnym alfabetem a ρ_d metryką dyskretną w zbiorze V, tzn. funkcją $\rho_d:V\times V\to R^+$ zdefiniowaną wzorem

$$\rho_d(x,y)=\begin{cases}1 \text{ gdy } x\ne y\\ 0 \text{ gdy } x=y\end{cases}$$

Pokazać, że funkcja $\rho:V^n\times V^n\to R^+$, (gdzie n jest ustaloną dowolną liczbą naturalną) zdefiniowana dla każdego $x=(x_1,x_2,\ldots,x_n)\in V^n$ i $y=(y_1,y_2,\ldots,y_n)\in V^n$ wzorem

$$\rho_H((x_1,x_2,...,x_n),(y_1,y_2,...,y_n))\overset{df}{=}\sum_{i=1}^{n}\rho_d(x_i,y_i)$$

jest metryką (inaczej jak mówimy odległością) w przestrzeni V^n.
Tę metrykę ρ_H nazywamy metryką Hamminga.

Rozwiązanie:
Musimy wykazać, że zdefiniowana w treści zadania funkcja $\varphi : V^n \times V^n \to R^+$ spełnia trzy następujące warunki

(*) dla każdego $x,y \in V^n, \rho(x,y)=0 \Leftrightarrow x=y$

(**) dla każdego $x,y \in V^n, \rho(x,y)=\rho(y,x)$ (symetria)

(***) dla każdego $x,y,z \in V^n, \rho(x,z) \le \rho(x,y)+\rho(y,z)$ (warunek trójkąta)

Sprawdzimy najpierw warunek (*).
Wynikanie $\Leftarrow$ jest oczywiste, bo z warunku $x=y$ czyli $(x_1,x_2,...,x_n)=(y_1,y_2,...,y_n)$ wynika, że $x_i=y_i$ dla $i=1,2,...,n$ zatem $\rho_d(x_i,y_i)=0$ dla $i=1,2,...,n$ i $\rho(x,y)=0$.

Wynikanie $\Rightarrow$. Jeśli $\rho(x,y)=0$, to wynika stąd, że dla każdego $i=1,2,...,n$ $\rho_d(x_i,y_i)=0$, a ponieważ ρ_d jest metryką mamy dalej: dla każdego $i=1,2,...,n$ $x_i=y_i$ czyli $x=y$.
To, że $\rho(x,y)=\rho(y,x)$ (warunek (**)) wynika bezpośrednio z faktu, że dla każdego $i=1,2,...,n$ $\rho_d(x_i,y_i)=\rho_d(y_i,x_i)$.
Weźmy dowolne $x,y,z \in V^n$, ponieważ metryka dyskretna ρ_d jest metryką, mamy

$$\rho(x,z)=\sum_{i=1}^{n}\rho_d(x_i,z_i)\le\sum_{i=1}^{n}(\rho_d(x_i,y_i)+\rho_d(y_i,z_i))$$

$$=\sum_{i=1}^{n}\rho_d(x_i,y_i)+\sum_{i=1}^{n}(\rho_d(x_i,z_i)=\rho(x,y)+\rho(y,z).$$

Zatem warunek trójkąta (warunek (***)) zachodzi..

Zadanie 1.32

Wagą słowa binarnego z przestrzeni $\{0,1\}^n$ (lub ogólniej z $\{0,1\}^*$) nazywamy liczbą jedynek w tym słowie. W ten sposób definiujemy na $\{0,1\}^*$ funkcję $w:\{0,1\}^* \to N \cup \{0\}$. Oczywiście dla $a \in \{0,1\}^n$ mamy $w(a)=\varphi_H(0,a)$ gdzie φ_H jest metryką Hamminga. Wyrazić metrykę Hamminga ρ_H w $\{0,1\}^n$ za pomocą funkcji wagi *w*.

Rozwiązanie:
Niech $x,y \in \{0,1\}^n, x=(x_1,x_2,...,x_n), y=(y_1,y_2,...,y_n)$, wówczas

$$\rho(x,y)\overset{df}{=}\sum_{i=1}^{n}\rho_d(x_i,y_i)=\sum_{i=1}^{n}w(x_i -_2 y_i)=w(x-y)=w(x+y)$$

gdzie ρ_d jest metryką dyskretną w zbiorze $\{0,1\}$ a symbol $-_2$ oznacza działanie odejmowania w Z_2 tzn.

$$x_i -_2 y_i \overset{df}{=} z \text{ wtedy i tylko wtedy } z \oplus_2 y_i = x_i.$$

Łatwo sprawdzić, że działania $\oplus_2$ i $-_2$ w Z_2 są tym samym dwuargumentowym działaniem.

Działanie " - " jest działaniem odejmowania wektorów z $\{0,1\}^n$ (odejmowane modulo 2 w Z_2 po współrzędnych). Działanie "+ " jest działaniem dodawania wektorów z $\{0,1\}^n$ (dodawanie modulo 2 w Z_2 po współrzędnych).

Zadanie 1.33

Z ilu elementów składa się kula domknięta $K(x,r) \overset{df}{=} \{y \in \{0,1\}^n ; \rho_H(x,y) \le r\}$ o promieniu $r > 0$ w przestrzeni Hamminga $\{0,1\}^n$. Jak „wygląda" taka kula? Czy coś się zmieni jeśli zastąpimy zbiór $\{0,1\}$ dowolnym niepustym zbiorem skończonym V tzn. rozważamy przestrzeń metryczną (V^n, ρ_H) (V może być np. dowolnym ustalonym alfabetem) ?

Rozwiązanie:
Zauważmy najpierw, że metryka Hamminga jest odwzorowaniem

$$\rho_H : \{0,1\}^n \times \{0,1\}^n \to N \cup \{0\}$$

a więc przyjmuje wartości całkowitoliczbowe. Oznaczając $r' = \lfloor r \rfloor$ (gdzie $\lfloor x \rfloor$ jest podłogą dla liczby rzeczywistej $x \in R$) dostajemy $K(x,r) = K(x,r')$.

Oznaczmy $x = (x_1, x_2,..., x_n) \in \{0,1\}^n$, $y = (y_1, y_2,..., y_n) \in \{0,1\}^n$ wówczas

$$K(x,r) = K(x,r') = \{(y_1, y_2,..., y_n) \in \{0,1\}^n ; \rho_H((x_1, x_2,..., x_n),(y_1, y_2,..., y_n)) \le r\} =$$

$$= \begin{cases} \{(y_1, y_2,...y_n) \in \{0,1\}^n ;\ card\{i;\ x_i \ne y_i\} \le r'\} & \text{jesli} \quad r' \le n\} \\ \{0,1\}^n & \text{jesli} \quad r' \ge n \end{cases}$$

Zatem

$$card\ K(x,r) = \begin{cases} \binom{n}{r'} + \binom{n}{r'-1} + ... + \binom{n}{0} & \text{jesli} \quad r' \le n \\ 2^n & \text{jesli} \quad r' \ge n \end{cases}$$

Jeśli zastąpimy zbiór $\{0,1\}$ zbiorem skończonym V, to wówczas mamy:

$$K(x,r) = \begin{cases} \{(y_1, y_2,...y_n) \in V^n ;\ card\{i;\ x_i \ne y_i\} \le r'\} & \text{jesli} \quad r' \le n\} \\ V^n & \text{jesli} \quad r' \ge n \end{cases} \qquad (*)$$

oraz

$$card\ K(x,r) = \begin{cases} \binom{n}{r'}(card\ \ V - 1)^{r'} + \binom{n}{r'-1}(card\ \ V - 1)^{r'-1} + ... + \binom{n}{0} & \text{jesli} \quad r' \le n \\ (card\ V)^n & \text{jesli} \quad r' \ge n \end{cases}$$

Łatwo zauważyć, że w przypadku zbioru V nieskończonego, wzór (*) pozostaje niezmieniony, natomiast kula $K(x,r)$ zawiera nieskończoną liczbę elementów.

Zadanie 1.34

Pokazać, że przestrzeń Hamminga $\{0,1\}^n$ jest przestrzenią liniową nad ciałem $Z_2 = \{0,1\}$.

Rozwiązanie:
Wiadomo (por. zadanie xx), że Z_p (gdzie p jest liczba pierwszą) jest ciałem, zatem Z_2 jest ciałem.

Ogólnie rzecz biorąc, jeśli K jest ciałem, to $K^n \overset{ozn}{=} \underbrace{K \times K \times \ldots \times K}_{n}$ jest przestrzenią liniową nad ciałem K, przy czym działanie mnożenia wektora $x = (x_1, x_2, \ldots, x_n) \in K^n$ przez skalar $\alpha \in K$ definiujemy jako

$$\alpha x \overset{df}{=} (\alpha \cdot x_1, \alpha \cdot x_2, \ldots, \alpha \cdot x_n) \quad (*)$$

(gdzie kropki po prawej stronie równości (*) oznaczają mnożenie w ciele K), a działanie dodawania wektorów $x = (x_1, x_2, \ldots, x_n), y = (y_1, y_2, \ldots, y_n)$ z K^n tak:

$$x + y \overset{df}{=} (x_1 + y_1, x_2 + y_2, \ldots, x_n + y_n)$$

Biorąc to co powyżej pod uwagę stwierdzamy, że Z_2^n jest przestrzenią liniową nad Z_2.

Zadanie 1.35

Niech A będzie zbiorem $< 0, pq - 1 >$, gdzie p i q są różnymi liczbami pierwszymi takimi, że liczby p-1 i q-1 nie są podzielne przez 3. Pokazać, że funkcja zadana wzorem

$$f(x) = x^3 (\mathrm{mod}\, pq)$$

czyli podnoszenie do trzeciej potęgi w pierścieniu Z_n, (gdzie $n = pq$) jest permutacją.

Funkcja f jest klasycznym przykładem tzw. zapadkowej funkcji jednokierunkowej (ang. trapdoor one-way function). Pokazać, co stanowi informację zapadkową (ang. the trapdoor information) dla funkcji f.

Uwaga: W praktyce p i q są liczbami pierwszymi mającymi po około 100-150 cyfr dziesiętnych.

Uwaga: Nie ma również efektywnych algorytmów obliczenia pierwiastka trzeciego stopnia w pierścieniu Z_n, czyli inaczej algorytmów odwracania funkcji f. Dla dużych p i q odwracanie f jest praktycznie nierealizowalne.

Rozwiązanie:

Niech p będzie ustaloną liczbą pierwszą i $a \in Z_p$. Jeśli $NWD(a, p-1)$ z 1, czyli liczby a i p-1 są względnie pierwsze, to istnieje element odwrotny $a^{-1} \in Z_{p-1}$ i funkcja

$$h : Z_{p-1} \ni j \to a \otimes_{p-1} j \in Z_{p-1}$$

jest różnowartościowa i „na”. Istotnie, wystarczy wykazać, że funkcja h jest „na”, bo dziedzina i przeciwdziedzina funkcji są równolicznymi zbiorami skończonymi. Z kolei to, że h jest „na”, wynika z rozwiązalności dla każdego $b \in Z_{p-1}$ równania (1) względem x.

$$a \otimes_{p-1} x = b \qquad (1)$$

Mnożąc obie strony równania (1) przez $a^{-1} \in Z_{p-1}$ dostajemy:

$$x = a^{-1} \otimes_{p-1} b \qquad (2)$$

Zatem dla dowolnego $b \in Z_{p-1}$ równanie (1) ma rozwiązanie (2). Biorąc jako a liczbę 3 i korzystając z założenia z treści zadania $NWD(3, p-1) = 1$ dostajemy, że przekształcenie

$$Z_{p-1} \ni j \to 3 \otimes_{p-1} j \in Z_{p-1}$$

jest różnowartościowe i „na”.

Zauważmy jeszcze, że każdy element grupy multiplikatywnej Z_p^* (jest to grupa cykliczna o p-1 elementach) daje się jednoznaczne przedstawić w postaci g^j dla pewnego $j = 1,2,..., p-1$, gdzie g jest generatorem grupy Z_p^*.

By wykazać, że $f : Z_n \to Z_n$ jest permutacją, wystarczy tak jak w przypadku funkcji h z punktu 1 wykazać, że funkcja f jest „na", tzn. rozwiązalne jest dla każdego $b \in Z_n$ równanie (3) w pierścieniu Z_n.

$$x^3 = b \tag{3}$$

Równość (3) jest równoważna kongruencji (4):

$$x^3 \equiv b (\bmod n) \tag{4}$$

Z kolei (ponieważ $n = p \cdot q$ i liczby pierwsze p i q są różne) kongruencja (4) jest równoważna układowi dwu kongruencji

$$x^3 \equiv b (\bmod p) \tag{5}$$

$$x^3 \equiv b (\bmod q) \tag{6}$$

Wystarczy więc pokazać, że dla każdego $b' \in Z_p$ (dla liczby pierwszej q rozumujemy analogicznie) rozwiązalne jest w ciele Z_p równanie

$$x^3 \equiv b' \tag{7}$$

Jeśli $b' = 0$, to $x = 0$, zatem rozwiązanie równania (7) istnieje. Załóżmy teraz, że $b' \neq 0$, a dokładniej $b' \in Z_p^*$, wówczas oczywiście rozwiązanie x równania (7), o ile istnieje, jest $\neq 0$.

Jeśli g oznacza generator grupy multiplikatywnej Z_p^*, a rozwiązanie x równania (7) istnieje, to istnieją takie liczby $j, k \in Z_{p-1}$, że $x = g^j$ i $b' = g^k$. Równanie (7) w ciele Z_p można zapisać teraz tak

$$g^{3j} = g^k \tag{8}$$

i jest to równanie względem j.

Biorąc pod uwagę fakt, że dla każdego $x \in Z_p^*$ mamy w Z_p $x^{p-1} = 1$ (wniosek z małego twierdzenia Fermata lub ogólniejszego faktu z teorii grup, ze rząd elementu jest dzielnikiem rzędu grupy , a grupa Z_p^* ma p-1 elementów) równanie (8) można zapisać jako:

$$g^{3 \otimes_{p-1} j} = g^k \tag{9}$$

co jest równoważne równości wykładników potęg w równaniu (9) modulo p-1, czyli równoważne równaniu (10) w pierścieniu Z_{p-1}

$$3 \otimes_{p-1} j = k \tag{10}$$

skąd

$$j = k \otimes_{p-1} 3^{-1} \tag{11}$$

gdzie 3^{-1} jest odwrotnością w Z_{p-1}. Zatem poszukiwane rozwiązanie x równania (7) jest równe:

$$x = g^j = g^{k \otimes_{p-1} 3^{-1}}$$

co dowodzi rozwiązalności równania (3) w pierścieniu Z_n, czyli równania $x^3 = b$ i ostatecznie tego, że funkcja $f : Z_n \to Z_n$, przy przyjętych założeniach jest różnowartościowa i „na”. Funkcja f jest więc permutacją zbioru Z_n, czego mieliśmy dowieść.

Zauważmy, że znajomości rozkładu liczby $n = p \cdot q$ na czynniki pierwsze, czyli znajomość liczby p i q umożliwia efektywne rozwiązywanie równania (3). Pewnym problemem może wydawać się znalezienie generatora g grupy multiplikatywnej Z_p^*. Istnieją jednak efektywne algorytmy znajdujące generator grupy Z_p^*.

Zatem znajomość rozkładu liczby n na czynniki pierwsze, czyli znajomość p i q stanowi informację zapadkową (trapdoor information).

Uwaga: Analogicznie jak w powyższym zadaniu rozumowanie można przeprowadzić dla każdej funkcji $f : Z_n \to Z_n$, zadanej wzorem

$$f : Z_n \ni x \to x^m \in Z_n$$

przy założeniach, że $n = p_1 \cdot p_2 \cdot \ldots \cdot p_r$, gdzie p_i dla $i = 1,2,\ldots,r$ są parami różnymi liczbami pierwszymi, oraz dla każdego $i = 1,2,\ldots,r$ mamy:

$$NWD(m, p_i - 1) = 1.$$

W tej ogólniejszej sytuacji informacją zapadkową będzie rozkład liczby $n = p_1 \cdot p_2 \cdot \ldots \cdot p_r$ na czynniki pierwsze, czyli znajomość liczb pierwszych $p_1, p_2, \ldots, p_r$.

Uwaga: Zastosowana technika rozwiązywania równania (3) polega w gruncie rzeczy na sprowadzeniu tego równania do dwóch równań w pierścieniu Z_p i Z_q.

$$x_1^3 = [b]_p$$

$$x_2^3 = [b]_q$$

rozwiązaniu tych równań, uzyskaniu $x_1 \in Z_p$ i $x_2 \in Z_q$, a następnie odtworzeniu $x \in Z_n$ z wartości $x_1 \in Z_p$ i $x_2 \in Z_q$. Odtworzenie sprowadza się do skorzystania z tezy chińskiego twierdzenia o resztach.