

# *WSTĘP DO KRYPTOGRAFII*

**ĆWICZENIA V**  
(podstawy teorii liczb)

## Zagadnienia:

- największy wspólny dzielnik,
- rozszerzony algorytm Euklidesa,
- liniowe równania modularne,
- mnożymy odwrotność,
- szybkie potęgowanie modularne,
- liczby pseudopierwsze Eulera,

- liczby pseudopierwsze Millera-Rabina,
- faktoryzacja metodą  $\rho$  Pollarda.

**Przypomnienie:** niech  $n, a, b \in \mathbb{Z}$ . Liczbę  $n$  nazywamy największym wspólnym dzielnikiem liczb  $a, b$  (ozn.  $NWD(a, b) = n$ ) wttw. gdy

$$\neg \exists m \in \mathbb{N} : m > |n| \wedge a \bmod m = 0 \wedge b \bmod m = 0.$$

Algorytm Euklidesa obliczający  $NWD(a, b)$ :

---

```
int EUKLIDES(int a,int b)
{

    if (b=0) return a;

    else return EUKLIDES(b,a mod b);

}
```

---

### **Zadanie 1:**

Oblicz  $NWD(241, 557)$ .

Rozwiązanie:

$$\begin{aligned} \textit{EUKLIDES}(241, 557) &= \textit{EUKLIDES}(557, 241) \\ &= \textit{EUKLIDES}(241, 75) \\ &= \textit{EUKLIDES}(75, 16) \\ &= \textit{EUKLIDES}(16, 11) \\ &= \textit{EUKLIDES}(11, 5) \\ &= \textit{EUKLIDES}(5, 1) \\ &= \textit{EUKLIDES}(1, 0) \\ &= 1. \end{aligned}$$



**Przypomnienie:** niech  $n, a, b \in \mathbb{Z}$  oraz  $x, y \in \mathbb{Z}$ . Załóżmy, że

$$NWD(a, b) = n = x \cdot a + y \cdot b.$$

Jak jednocześnie obliczyć wartość liczb  $n$ ,  $x$  oraz  $y$  oraz. Odpowiedź na to pytanie daje nam rozszerzony algorytm Euklidesa:

---

```
(int,int,int) ROZSZERZONY_EUKLIDES(int a,int b)
{

    int n, x, y;

    if (b=0) return (a,1,0);

    (n,x,y):=ROZSZERZONY_EUKLIDES(b,a mod b);

    (n,x,y):=(n,y,x-[a/b]*y);

    return (n,x,y);

}
```

---

## Zadanie 2:

Oblicz wartość liczb  $n$ ,  $x$  i  $y$  dla wyrażenia  $NWD(99, 78) = n = x \cdot 99 + y \cdot 78$ .

Rozwiązanie:

$a$	$b$	$\lfloor a/b \rfloor$	$n$	$x$	$y$
99	78	-	-	-	-
78	21	-	-	-	-
21	15	-	-	-	-
15	6	-	-	-	-
6	3	-	-	-	-
3	0	-	3	1	0
6	3	2	3	0	1
15	6	2	3	1	-2
21	15	1	3	-2	3
78	21	3	3	3	-11
99	78	1	3	-11	14



**Przypomnienie:** niech  $z, a, b \in \mathbb{Z}$  oraz  $k \in \mathbb{N} \setminus \{0\}$ . Liniowym równaniem modularnym nazywamy zależność postaci

$$a \cdot z \equiv b \pmod{k},$$

gdzie  $z$  jest zmienną. Jeżeli równanie to jest rozwiązywalne, to istnieje dokładnie  $n$  rozwiązań  $z_i$ , gdzie  $n = \text{NWD}(a, k)$  i  $i \in \{0, 1, \dots, n - 1\}$ . Poniższy algorytm generuje wszystkie  $n$  rozwiązań:

---

```
LINIOWE_RÓWNANIE_MODULARNE(int a,int b,int k)
{

    int i, n, x, y, z;

    (n,x,y):=ROZSZERZONY_EUKLIDES(a,k);

    if (b mod n = 0)

    {

        z:=(x*b/n) mod k;

        for i:=0 to n-1 do wypisz (z+i*k/n) mod k;

    }

    else return 'BRAK ROZWIĄZANIA';

}
```

---

### **Zadanie 3:**

Rozwiąż równanie

$$14 \cdot z \equiv 30 \pmod{100}.$$

Rozwiązanie: przyjmujemy, że  $a = 14$ ,  $b = 30$  oraz  $k = 100$ . Obliczamy używając rozszerzony algorytm Euklidesa  $NWD(14, 100) = n = x \cdot 14 + y \cdot 100$ :

$a$	$b$	$\lfloor a/b \rfloor$	$n$	$x$	$y$
14	100	-	-	-	-
100	14	-	-	-	-
14	2	-	-	-	-
2	0	-	2	1	0
14	2	7	2	0	1
100	14	7	2	1	-7
14	100	0	2	-7	1

Ponieważ  $30 \bmod 2 = 0$  to

$$z = (-7) \cdot 30/2 \bmod 100 = 95.$$

Zatem procedura wypisze dwa rozwiązania:

- $z_0 = (z + 0 \cdot 100/2) \bmod 100 = (95 + 0) \bmod 100 = 95,$
- $z_1 = (z + 1 \cdot 100/2) \bmod 100 = (95 + 50) \bmod 100 = 45.$



**Przypomnienie:** niech  $z, a \in \mathbb{Z}$  oraz  $k \in \mathbb{N} \setminus \{0\}$ . Rozważmy liniowe równanie modularne postaci

$$a \cdot z \equiv 1 \pmod{k}.$$

Liczbę  $z$  nazywamy multiplikatywną odwrotnością modulo  $k$  liczby  $a$  (ozn.  $a^{-1}$ ). Jeżeli  $NWD(a, k) = 1$  to równie to ma dokładnie jedno rozwiązanie, wpp. jest nierozwiązywalne.

#### **Zadanie 4:**

Znajdź mnożącą odwrotność modulo 127 liczby 23.

Rozwiązanie: przyjmujemy, że  $a = 23$ ,  $b = 1$  oraz  $k = 127$ . Obliczamy używając rozszerzony algorytm Euklidesa  $NWD(23, 127) = n = x \cdot 23 + y \cdot 127$ :

$a$	$b$	$\lfloor a/b \rfloor$	$n$	$x$	$y$
23	127	-	-	-	-
127	23	-	-	-	-
23	12	-	-	-	-
12	11	-	-	-	-
11	1	-	-	-	-
1	0	-	1	1	0
11	1	11	1	0	1
12	11	1	1	1	-1
23	12	1	1	-1	2
127	23	5	1	2	-11
23	127	0	1	-11	2

Ponieważ  $1 \bmod 1 = 0$  to:

$$z = (-11) \cdot 1/1 \bmod 127 = 116,$$

zatem jedyne rozwiązanie równania jest następującej postaci:

$$z_0 = (z + 0 \cdot 127/1) \bmod 127 = (116 + 0) \bmod 100 = 116.$$

Dla pewności sprawdzamy:  $23 \cdot 116 = 2668 \bmod 127 = 1$ , czyli  $a^{-1} \bmod 127 = 116$ .



**Przypomnienie:** niech  $a, b \in \mathbb{N}$  oraz  $k \in \mathbb{N} \setminus \{0\}$ . Interesuje nas szybkie obliczenie  $x$  takiego, że

$$x = a^b \bmod k.$$

Założmy, że  $b = \begin{bmatrix} b_n & b_{n-1} & \dots & b_1 & b_0 \end{bmatrix}$  jest binarną reprezentacją liczby  $b$ , gdzie  $n = \lfloor \log_2 b \rfloor + 1$ . Algorytm szybkiego potęgowania modularnego ma następującą postać:

---

```
int SZYBKIE_POTĘGOWANIE_MODULARNE(int a,int b,int k)
{

    int tmp1a:=0, tmp1b:=0 tmp2a:=0, tmp2b:=1;

    for i:=n downto 0 do

    {

        tmp1a:=2*tmp1b;

        tmp2a:=(tmp2b*tmp2b) mod k;

        if (b[i] = 1)

        {

            tmp1b:=tmp1a+1;
```

```
    tmp2b:=(tmp2a*a) mod k;  
  
}  
  
else  
  
{  
  
    tmp1b:=tmp1a;  
  
    tmp2b:=tmp2a;  
  
}  
  
}  
  
return tmp2b;  
  
}
```

---

### **Zadanie 5:**

Oblicz  $11^{4487} \bmod 100$ .

Rozwiązanie: liczba 561 zapisana w postaci binarnej to ciąg bitów

1000110000111.

Dalej wykonujemy obliczenia:

$b_i$	tmp1a	tmp2a	tmp1b	tmp2b
-	0	0	0	1
1	0	1	1	11
0	2	21	2	21
0	4	41	4	41
0	8	81	8	81
1	16	61	17	71
1	34	41	35	51
0	70	1	70	1
0	140	1	140	1
0	280	1	280	1
0	560	1	560	1
1	1120	1	1121	11

1	2242	21	2243	31
1	4486	61	4487	71

Odpowiedź to:  $11^{4487} \bmod 100 = 71$ .



**Przypomnienie:** niech  $a, k \in \mathbb{N}$  oraz  $2 \leq a < k$ . Jeżeli:

$$a^{k-1} \bmod k = 1$$

to liczbę  $k$  nazywamy liczbą pseudopierwszą Eulera przy podstawie  $a$ , wpp. liczba  $k$  jest złożona. Algorytm sprawdzający czy dana liczba jest liczbą pseudopierwszą Eulera przy podstawie  $a$  ma następującą postać:

---

```
TEST_EULERA(int a,int k)
{

    if (SZYBKIE_POTĘGOWANIE_MODULARNE(a,k-1,k)=1)

        wypisz "Liczba k jest liczbą pseudopierwszą Eulera przy podstawie a";

    else

        wypisz "Liczba k jest złożona";

}
```

---

### **Zadanie 6:**

Stwierdź, czy liczba 341 jest liczbą pseudopierwszą Eulera przy podstawie 2.

Rozwiązanie: nasze zadanie sprowadza się do obliczenia wartości wyrażenia  $2^{340} \bmod 341$ . Liczba 340 zapisana w postaci binarnej to ciąg bitów

101010100.

Dalej wykonujemy obliczenia:

$b_i$	tmp1a	tmp2a	tmp1b	tmp2b
-	0	0	0	1
1	0	1	1	2
0	2	4	2	4
1	4	16	5	32
0	10	1	10	1
1	20	1	21	2
0	42	4	42	4
1	84	16	85	32
0	170	1	170	1
0	340	1	340	1

Ponieważ  $2^{340} \bmod 341 = 1$  to liczba 341 jest liczbą pseudopierwszą przy podstawie 2. Należy jednak pamiętać, że  $341 = 31 \cdot 11$ .



**Przypomnienie:** niech  $z, p \in \mathbb{N}$ , wtedy jeżeli liczba  $p > 2$  jest pierwsza to równanie to równanie

$$z^2 \equiv 1 \pmod{p}$$

ma dokładnie dwa rozwiązania, a mianowicie  $z_1 = 1$  oraz  $z_2 = -1$  nazywane trywialnymi pierwiastkami kwadratowymi z 1 modulo  $p$ . Pierwiastki dowolnego równania kwadratowego

$$z^2 \equiv 1 \pmod{k},$$

dla  $k \in \mathbb{N} \setminus \{0\}$  nazywamy nietrywialnymi pierwiastkami kwadratowymi z 1 modulo  $k$ , jeżeli  $z_1$  oraz  $z_2$  nie przystają do 1 bądź  $-1$  modulo  $k$ .

Niech  $a, k \in \mathbb{N}$  oraz  $2 \leq a < k$ . Jeżeli:

- liczba  $k$  jest liczbą pseudopierwszą Eulera przy podstawie  $a$ ,
- podczas wykonywania algorytmu składowego SZYBKIE\_POTĘGOWANIE\_MODALNE( $a, k-1, k$ ) procedury TEST\_EULERA(int  $a$ , int  $k$ ) nie wystąpił nietrywialny pierwiastek kwadratowy z 1 modulo  $k$ ,

to liczba  $k$  jest liczbą pseudopierwszą Millera-Rabina przy podstawie  $a$ . Algorytm sprawdzający czy dana liczba jest liczbą pseudopierwszą Millera-Rabina przy podstawie  $a$  ma następującą postać:

---

```
TEST_MILLERA_RABINA(int a,int k)
{

    if (SZYBKIE_POTĘGOWANIE_MODULARNE(a,k-1,k)=1)

        if (w trakcie wykonywania procedury SZYBKIE_POTĘGOWANIE_MODULARNE(a,k-1,k)
            nie wystąpił nietrywialny pierwiastek kwadratowy z 1 modulo k)

            wypisz "Liczba k jest liczbą pseudopierwszą Millera-Rabina przy podstawie a";

        else

            wypisz "Liczba k jest złożona";

    else

        wypisz "Liczba k jest złożona";

}
```

---

### **Zadanie 7:**

Stwierdź, czy liczba 561 jest liczbą pseudopierwszą Millera-Rabina przy podstawie 2.

Rozwiązanie: nasze zadanie sprowadza się do obliczenia wartości wyrażenia  $2^{560} \bmod 561$ . Liczba 560 zapisana w postaci binarnej to ciąg bitów

1000110000.

Dalej wykonujemy obliczenia:

$b_i$	tmp1a	tmp2a	tmp1b	tmp2b
-	0	0	0	1
1	0	1	1	2
0	2	4	2	4
0	4	16	4	16
0	8	256	8	256
1	16	460	17	359
1	34	412	35	263
0	70	166	70	166
0	140	67	140	67
0	280	1	280	1

0	560	1	560	1
---	-----	---	-----	---

Liczba 561 nie jest więc liczbą pseudopierwszą Millera-Rabina przy podstawie 2, jest jednak liczbą pseudopierwszą Fermata (przy tej samej podstawie). Faktycznie  $561 = 3 \cdot 11 \cdot 17$ .



### **Zadanie 8:**

Stwierdź, czy liczba 241 jest liczbą pseudopierwszą Millera-Rabina przy podstawie 7.

Rozwiązanie: nasze zadanie sprowadza się do obliczenia wartości wyrażenia  $7^{240} \bmod 241$ . Liczba 240 zapisana w postaci binarnej to ciąg bitów

11110000.

Dalej wykonujemy obliczenia:

$b_i$	tmp1a	tmp2a	tmp1b	tmp2b
-	0	0	0	1
1	0	1	1	7
1	2	49	3	102
1	6	41	7	46
1	14	188	15	111
0	30	30	30	30
0	60	177	60	177
0	120	240	120	240
0	240	1	240	1

Liczba 241 jest więc liczbą pseudopierwszą Millera-Rabina przy podstawie 2, jest także liczbą pseudopierwszą Fermata (przy tej samej podstawie). Co więcej, 241 jest liczbą pierwszą.



**Przypomnienie:** niech  $k \in \mathbb{N}$  będzie liczbą złożoną. Faktoryzacją liczby  $k$  nazywamy  $n$ -elementowy iloczyn liczb pierwszych  $p_i$  postaci  $k = p_1^{i_1} \cdot p_2^{i_2} \cdot \dots \cdot p_n^{i_n}$ , gdzie  $i \in \{1, 2, \dots, n\}$ . Poniższa metoda faktoryzacji dowolnej liczby  $k$  nazywana metodą  $\rho$  Pollarda jest heurystyką, więc nie gwarantuje poprawności wyniku jak i złożoności działania. W praktyce dla stosunkowo niewielkich  $k$  jest ona jednak bardzo efektywna:

---

```
int FAKTORYZACJA_RO_POLLARDA(int k)
{

    int tmp1:=2, tmp2:=2, dzielnik:=1;

    while (dzielnik=1) do

    {

        tmp1:=(tmp1*tmp1+1) mod k;

        tmp2:=(tmp2*tmp2+1) mod k; tmp2:=(tmp2*tmp2+1) mod k;

        dzielnik:=Euklides(tmp1-tmp2,k);

    }

    return dzielnik;

}
```

---

### **Zadanie 9:**

Używając metody Pollarda podaj dowolny nietrywialny dzielnik liczby 455459.

Rozwiązanie:

tmp1	tmp2	tmp1-tmp2	dzielnik
2	2	0	1
5	26	-21	1
26	2871	-2854	1
677	179685	-179018	1
2871	155260	-152389	1
44380	416250	-371870	1
179685	43670	136015	1
121634	164403	-42769	1
155260	247944	-92684	1
44567	68343	-23776	743

Zatem dzielnikiem 455459 jest 743 jak i  $455459/743 = 613$ .

