Uniwersytet Zielonogórski
Instytut Sterowania i Systemów Informatycznych

Algorytmy i struktury danych

Laboratorium Nr 10

Teoria liczb

1 Cel ćwiczenia

Algorytmy teorioliczbowe znajdują szerokie zastosowanie dzięki wynalezieniu systemów kryptograficznych bazujących na dużych liczbach pierwszych. Możliwość praktycznego stosowania tych systemów opiera się na tym, że istnieją algorytmy pozwalające na łatwe znajdywanie dużych liczb pierwszych, natomiast ich bezpieczeństwo wynika z tego, iż nie potrafimy rozłożyć na czynniki iloczynu dużych liczb pierwszych. Ćwiczenie przedstawia wiadomości z teorii liczb oraz kilka algorytmów ilustrujących zastosowania tej teorii.

2 Podstawowe pojęcia teorii liczb

2.1 Podzielność i dzielniki

Podzielność jednej liczby całkowitej przez drugą jest podstawowym pojęciem w teorii liczb. Zapis $d|a$ oznacza, że liczba d **dzieli** liczbę a, czyli $a = k \cdot d$ dla pewnej liczby całkowitej k. Każda liczba całkowita dzieli liczbę 0. Jeśli $a > 0$ i $d|a$,to $|d| \leq |a|$. Jeśli $d|a$, to mówimy też, że liczba a jest **wielokrotnością** liczby d. Jeśli d nie dzieli a, to piszemy $d \nmid a$.

Jeśli $d|a$ i $d \geq 0$, to mówimy, że liczba d jest **dzielnikiem** liczby a. Dowolny dzielnik liczby a wzięty ze znakiem minus także dzieli a; ponadto dzielnik liczby całkowitej a jest równy przynajmniej 1, ale nie większy niż $|a|$. Przykładowo dzielnikami liczby 24 są $1, 2, 3, 4, 6, 8, 12$ i 24.

Każda liczba całkowita a jest podzielna przez dzielniki trywialne: 1 oraz a. Nietrywialne dzielniki liczby a są także zwane czynnikami liczby a. Na przykład czynnikami liczby 20 są: $2, 4, 5$ i 10.

2.2 Liczby pierwsze i złożone

Liczbę całkowitą $a > 1$, której jedynymi dzielnikami są trywialne dzielniki 1 i a, nazywamy **liczbą pierwszą**. Liczby pierwsze mają wiele szczególnych własności i odgrywają zasadniczą rolę w teorii liczb. Dwadzieścia początkowych liczb pierwszych to, po kolei:

$$2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71$$

Liczb pierwszych jest nieskończenie wiele. Liczbę całkowitą $a > 1$, która nie jest liczbą pierwszą, nazywamy liczbą złożoną. Na przykład 39 jest liczbą złożoną, bo $3|39$. Liczba 1 nie jest ani liczbą pierwszą, ani złożoną, podobnie jak 0 i wszystkie ujemne liczby całkowite.

2.3 Twierdzenie o dzieleniu z resztą, reszty i przystawanie modulo

Dla dowolnej liczby całkowitej a i dowolnej dodatniej liczby całkowitej n istnieją wyznaczone jednoznacznie liczby całkowite q i r takie, że $0 \leq r < n$ oraz $a = q \cdot n + r$. Wartość $q = \lfloor \frac{a}{n} \rfloor$ jest ilorazem dzielenia. Wartość $r = a \bmod n$ jest **resztą** z dzielenia. Relacja $n|a$ zachodzi wtedy i tylko wtedy, gdy $a \bmod n = 0$.

Liczby całkowite można podzielić na n klas równoważności ze względu na ich reszty modulo n. Klasa równoważności modulo n zawierająca liczbę a to zbiór $[a]_n = a + k \cdot n : k \in \mathbb{Z}$.

Na przykład $[3]_7 = \ldots, -11, -4, 3, 10, 17, \ldots$; inne oznaczenia tego samego zbioru to $[-4]_7$ i $[10]_7$. Zapis $a \in [b]_n$ oznacza to samo co $a \equiv b (\bmod\ n)$. Znak $\equiv$ oznacza **kongruencję**. Zbiór wszystkich takich klas równoważności to inaczej zupełny zbiór reszt modulo n, mający postać $\mathbb{Z}_n = [a]n : 0 \leq a \leq n - 1$. Często

można spotkać się z definicją $\mathbb{Z}_n = 0, l, \ldots, n-l$, Obie te definicje są równoważne w tym sensie, że 0 reprezentuje zbiór $[0]_n$, 1 reprezentuje $[1]_n$ itd., a każda klasa jest reprezentowana przez swój najmniejszy nieujemny element. Należy jednak pamiętać, że pojedyncza liczba oznacza tu całą klasę. Jeśli na przykład odwołujemy się do -1 jako do elementu $\mathbb{Z}_n$, chodzi nam o $[n-1]_n$, ponieważ $-1 \equiv n-1 (\text{mod } n)$.

2.4 Wspólne dzielniki i największy wspólny dzielnik

Jeśli d jest dzielnikiem a, ale także dzielnikiem b, to d jest wspólnym dzielnikiem a i b. Na przykład dzielnikami liczby 30 są: $1, 2, 3, 5, 6, 10, 15$ i 30, więc wspólnymi dzielnikami liczb 24 i 30 są: $1, 2, 3$ i 6. Należy zauważyć, że 1 jest wspólnym dzielnikiem dowolnych dwóch liczb całkowitych. Ważną własnością wspólnych dzielników jest to, że jeśli $d|a$ i $d|b$, to $d|(a+b)$ i $d|(a-b)$. Ogólniej, jeśli $d|a$ i $d|b$, to $d|(ax+by)$ dla dowolnych liczb całkowitych x i y. Oprócz tego, jeśli $a|b$, to albo $|a| \leq |b|$, albo $b = 0$, skąd wynika, że jeśli $a|b$ i $b|a$, to $a = \pm b$.

Największy wspólny dzielnik (**gcd**, od ang. greatest common divisor) dwóch liczb całkowitych a i b, które nie są jednocześnie równe 0, to największy spośród wspólnych dzielników a i b. Zapisujemy go jako gcd(a, b). Na przykład gcd(24, 30) = 6, gcd(5, 7) = 1, a gcd(0, 9) = 9. Jeśli żadna z liczb a, b nie jest zerem, to gcd(a, b) jest liczbą całkowitą z przedziału od 1 do min($|a|, |b|$). Definiujemy gcd(0, 0) jako 0, dzięki czemu standardowe własności funkcji gcd są zawsze prawdziwe.

Podstawowe własności funkcji gcd:

- $\gcd(a, b) = \gcd(b, a)$;
- $\gcd(a, b) = \gcd(-a, b)$;
- $\gcd(a, b) = \gcd(|a|, |b|)$;
- $\gcd(a, 0) = |a|$,
- $\gcd(a, k \cdot a) = |a|$ dla każdego $k \in \mathbb{Z}$.

Następujące twierdzenie daje alternatywną, użyteczną charakteryzację $\gcd(a, b)$: Jeśli a i b są dowolnymi liczbami całkowitymi, nie równymi jednocześnie zeru, to $\gcd(a, b)$ jest najmniejszym dodatnim elementem zbioru $ax + by : x, y \in \mathbb{Z}$ kombinacji liniowych a i b. Oznacza to, że dla dowolnych liczb całkowitych a i b, jeśli $d|a$ i $d|b$, to $d|\gcd(a, b)$. Dla dowolnych liczb całkowitych a i b oraz liczby naturalnej n zachodzi zależność $\gcd(a \cdot n, b \cdot n) = n * \gcd(a, b)$. Dla dowolnych dodatnich liczb całkowitych n, a i b, jeśli $n|a \cdot b$ oraz $\gcd(a, n) = 1$, to $n|b$.

2.5 Liczby względnie pierwsze

Dwie liczby całkowite a i b są **względnie pierwsze**, jeśli ich jedynym wspólnym dzielnikiem jest 1, tzn. jeśli $\gcd(a, b) = 1$. Na przykład liczby 8 i 15 są względnie pierwsze, ponieważ dzielnikami liczby 8 są: $1, 2, 4$ i 8, a dzielniki liczby 15 to: $1, 3, 5$ i 15. Poniższe twierdzenie mówi, że jeśli dwie liczby są względnie pierwsze z trzecią liczbą p, to ich iloczyn jest także względnie pierwszy z p. Dla dowolnych liczb całkowitych a, b i p, jeśli $\gcd(a, p) = 1$ i $\gcd(b, p) = 1$, to $\gcd(a \cdot b, p) = 1$. Mówimy, że liczby $n_1, n_2, \ldots, n_k$ są **parami względnie pierwsze**, jeśli dla $i \neq j$ mamy $\gcd(n_i, n_j) = 1$.

2.6 Jednoznaczność rozkładu

Dla dowolnej liczby pierwszej p i dowolnych liczb a, b, jeśli $p|a \cdot b$, to $p|a$ lub $p|b$.

Twierdzenie 1 (O niejednoznaczności rozkładu) *Każdą dodatnią liczbę całkowitą a można na dokładnie jeden sposób zapisać w postaci iloczynu $a = p_1^{e_1}, p_2^{e_2}, \ldots, p_r^{e_r}$, gdzie p_i są liczbami pierwszymi, $p_i < p_2 < \cdots < p_r$, zaś e_i są dodatnimi liczbami całkowitymi.*

Na przykład liczbę 6000 można jednoznacznie zapisać jako $2^4 \cdot 3 \cdot 53$.

2.7 Algorytm Euklidesa

Algorytm Euklidesa obliczania największego wspólnego dzielnika opiera się na następującym twierdzeniu: Dla dowolnej nieujemnej liczby całkowitej a i dodatniej liczby całkowitej b zachodzi zależność

$$\gcd(a, b) = \gcd(b, a \bmod b).$$

Poniższy algorytm obliczania największego wspólnego dzielnika został opisany w *Elementach* Euklidesa, około roku 300 p.n.e., chociaż jest być może jeszcze starszy. Zapisany jest jako procedura rekurencyjna oparta bezpośrednio na podanym powyżej twierdzeniu. Dane wejściowe a i b są dowolnymi nieujemnymi liczbami całkowitymi.

```
1 Euklides(a, b)
2 if b = 0 then return a;
3 else return Euklides(b, a mod b);
```

Przykład działania algorytmu Euklidesa - obliczenie $\gcd(30, 21)$:

$$\begin{aligned} \text{Euklides}(30, 21) &= \text{Euklides}(21, 9) \\ &= \text{Euklides}(9, 3) \\ &= \text{Euklides}(3, 0) \\ &= 3. \end{aligned}$$

W obliczeniu tym występują trzy rekurencyjne wywołania procedury *Euklides*.

Algorytm Euklidesa jest poprawny i zawsze się zakończy. Ciąg wywołań rekurencyjnych nie może być nieskończony, gdyż w każdym wywołaniu zmniejsza się wartość drugiego argumentu, a jest ona zawsze nieujemna.

2.8 Rozszerzony algorytm Euklidesa

Algorytm Euklidesa można zmodyfikować tak, aby obliczał dodatkowo całkowitoliczbowe współczynniki x i y takie, że

$$d = \gcd(a, b) = a \cdot x + b \cdot y \tag{1}$$

Należy zauważyć, że x i y mogą być równe 0 lub ujemne. Współczynniki te przydadzą się do obliczania odwrotności multyplikatywnej modulo. Dane wejściowe dla procedury *RozszerzonyEuklides* stanowi dowolna para nieujemnych liczb całkowitych, a zwraca ona trójkę liczb (d, x, y) spełniającą równanie (1).

```
1 RozszerzonyEuklides(a, b)
2 if b = 0 then return (a, 1, 0)
3 (d', x', y') ← RozszerzonyEuklides(b, a mod b)
4 (d, x, y) ← (d', y', x' − ⌊a/b⌋ · y')
5 return (d, x, y)
```

a	**b**	$\lfloor \frac{\mathbf{a}}{\mathbf{b}} \rfloor$	**d**	**x**	**y**
99	78	1	3	−11	14
78	21	3	3	3	−11
21	15	1	3	−2	3
15	6	2	3	1	−2
6	3	2	3	0	1
3	0	–	3	1	0

Przykład działania procedury *RozszerzonyEuklides* dla danych wejściowych 99 i 78. Każdy wiersz odpowiada jednemu poziomowi rekurencji i zawiera: wejściowe wartości a i b, obliczaną wartość $\lfloor \frac{a}{b} \rfloor$ oraz zwracane wartości d, x i y. Zwracana trójka (d, x, y) staje się trójką (d, x', y'), używaną do obliczeń na wyższym poziomie wywołań rekurencyjnych. W wyniku wywołania funkcji RozszerzonyEuklides$(99, 78)$ otrzymujemy $(3, -11, 14)$, zatem $\gcd(99, 78) = 3$ i $\gcd(99, 78) = 3 = 99 \cdot (-11) + 78 \cdot 14$.

3 Arytmetyka modularna

O arytmetyce modularnej można myśleć jak o zwykłej arytmetyce liczb całkowitych, z tą różnicą, że ponieważ obliczenia prowadzone są modulo n, każdy otrzymywany wynik należy zastąpić elementem zbioru $0, 1, \ldots, n-1$ przystającym do x modulo n (tzn. x zastępujemy przez x mod n). Ten nieformalny model wystarcza, jeśli ograniczamy się do operacji dodawania, odejmowania i mnożenia. Formalny model arytmetyki modularnej najlepiej opisywać w języku teorii grup.

3.1 Grupy skończone i podgrupy

Grupa $(S, \oplus)$ to zbiór S wraz ze zdefiniowaną w S dwuargumentową operacją $\oplus$ o następujących własnościach:

1. Domkniętość: Dla dowolnych $a, b \in S$ element $a \oplus b \in S$.
2. Element neutralny: Istnieje element $e \in S$ zwany elementem neutralnym, taki że $e \oplus a = a \oplus e = a$. dla dowolnego $a \in S$.
3. Łączność: Dla dowolnych $a, b, c \in S$ zachodzi $(a \oplus b) \oplus c = a \oplus (b \oplus c)$.
4. Element odwrotny: Dla każdego $a \in S$ istnieje dokładnie jeden element $b \in S$ zwany elementem odwrotnym do a, taki że $a \oplus b = b \oplus a = e$.

Jako przykład można rozważyć znaną już grupę $(\mathbf{Z}, +)$ liczb całkowitych $\mathbf{Z}$ z działaniem dodawania: 0 jest elementem neutralnym, a elementem odwrotnym do elementu a jest $-a$. Jeśli w grupie $(S, \oplus)$ jest spełnione prawo przemienności, to jest ona **grupą przemienną (abelową)**. Grupę $(S, \oplus)$, taką że $|S| < \infty$, nazywa się **grupą skończoną**.

Jeśli $(S, \oplus)$ jest grupą skończoną, $S' \subseteq S$ i $(S', \oplus)$ jest także grupą, to $(S', \oplus)$ nazywamy **podgrupą** $(S, \oplus)$. Na przykład liczby parzyste tworzą podgrupę grupy liczb całkowitych z operacją dodawania. Poniższe twierdzenie stanowi skuteczne narzędzie rozpoznawania podgrup:

Twierdzenie 2 (Niepusty zamknięty podzbiór grupy skończonej jest podgrupą) *Jeśli $(S, \oplus)$ jest grupą skończoną, a S' jest dowolnym niepustym podzbiorem S, takim, że $a \oplus b \in S'$ dla dowolnych $a, b \in S'$, to $(S', \oplus)$ jest podgrupą $(S, \oplus)$*

Na przykład zbiór $\{0, 2, 4, 6\}$ tworzy podgrupę $\mathbf{Z}_8$, ponieważ jest zamknięty ze względu na operację $+$.

Twierdzenie 3 (Lagrange'a)*: Jeśli $(S, \oplus)$ jest grupą skończoną, a $(S', \oplus)$ jest podgrupą $(S, \oplus)$, to $|S'|$ jest dzielnikiem $|S|$.*

Podgrupę S' grupy S nazywamy podgrupą **właściwą**, jeśli $S' \neq S$. Zatem jeśli S' jest właściwą podgrupą grupy skończonej S, to $|S'| \leq \frac{|S|}{2}$.

Istnieje następująca metoda tworzenia podgrupy grupy skończonej $(S, \oplus)$: należy wybrać element a i wziąć wszystkie elementy, które można uzyskać z a za pomocą działania grupowego. Ściślej, należy zdefiniować $a^{(k)}$ dla $k \geq 1$ jako:

$$a^{(k)} = \bigoplus_{i=1}^{k} a = \underbrace{a \oplus a \oplus a \cdots \oplus a}_{k}$$

Dla przykładu: jeśli $a = 2$ w grupie $\mathbb{Z}_6$, to ciąg $a^{(1)}, a^{(2)}, \ldots$ wygląda: $2, 4, 0, 2, 4, 0, 2, 4, 0, \ldots$.

W grupie $\mathbb{Z}_n$ spełnione jest $a^{(k)} = k \cdot a \bmod n$, w grupie $\mathbb{Z}_n^*$ spełnione jest zaś $a^{(k)} = a^k \bmod n$.

Podgrupę generowaną przez a, oznaczaną symbolem $\langle a \rangle$ lub $(\langle a \rangle, \oplus)$, definiuje się jako $\langle a \rangle = \{a^{(k)} : k \leq 1\}$. Mówi się, że element a **generuje** podgrupę $\langle a \rangle$ lub że a jest **generatorem** podgrupy $\langle a \rangle$. Ponieważ zbiór S jest skończony, więc $\langle a \rangle$ jest skończonym podzbiorem S, być może zawierającym cały zbiór S. Ponieważ z łączności operacji $\oplus$ wynika, że $a^{(i)} \oplus a^{(j)} = a^{(i+j)}$, zatem zbiór $\langle a \rangle$ jest zamknięty ze względu na działanie grupowe, czyli jest podgrupą S.

Rząd elementu a w grupie S, oznaczony przez $\text{ord}(a)$, definiowany jest jako najmniejsze $t > 0$, takie, że $a^{(t)} = e$. Dla dowolnej grupy skończonej $(S, \oplus)$ i dowolnego $a \in S$, rząd elementu a jest równy rozmiarowi generowanej przez niego grupy, czyli $\text{ord}(a) = |\langle a \rangle|$. Zatem ciąg $a^{(1)}, a^{(2)}, \ldots$ jest okresowy z okresem $t = \text{ord}(a)$, tzn. $a^{(i)} = a^{(j)}$ wtedy i tylko wtedy, gdy $i \equiv j$. Ponadto jeśli $(S, \oplus)$ jest grupą skończoną z elementem neutralnym e, to dla dowolnego $a \in S$ zachodzi zależność $a^{|S|} = e$.

3.2 Rozwiązywanie modularnych równań liniowych

Rozwiązywanie modularnych równań liniowych polega na znalezieniu rozwiązań równania

$$a \cdot x \equiv b (\bmod\ n) \tag{2}$$

gdzie $a > 0$ i $n > 0$.

Problem ten ma cały szereg zastosowań; można na przykład wykorzystać go w procedurze znajdowania kluczy w kryptosystemie z kluczem publicznym RSA. Zakłada się, że a, b i n są dane, szukane zaś to wszystkie liczby x modulo n, które spełniają powyższe równanie. Rozwiązań takich może nie być w ogóle, może być jedno lub więcej.

Niech $\langle a \rangle$ oznacza podgrupę $\mathbf{Z}_n$, generowaną przez a. Ponieważ $\langle a \rangle = \{a^{(x)} : x > 0\} = \{a \cdot x \bmod n : x > 0\}$, równanie 3 ma rozwiązanie wtedy i tylko wtedy, gdy $b \in \langle a \rangle$. Twierdzenie Lagrange'a, opisane w poprzednim paragrafie, mówi, że $|\langle a \rangle|$ musi być dzielnikiem n. Poniższe twierdzenie daje dokładną charakterystykę grupy $\langle a \rangle$.

Dla dowolnych dodatnich liczb całkowitych a i n, jeśli $d = \gcd(a, n)$, to

$$\langle a \rangle = \langle d \rangle = \left\{0, d, 2 \cdot d, \ldots, \left(\left(\frac{n}{d}\right) - 1\right) \cdot d\right\} \tag{3}$$

w $\mathbb{Z}_n$, a zatem $|\langle a \rangle| = \frac{n}{d}$.

A zatem równanie $a \cdot x \equiv b (\bmod\ n)$ jest rozwiązywalne ze względu na niewiadomą x wtedy i tylko wtedy, gdy $\gcd(a, n) | b$. Ponadto równanie to albo ma d różnych rozwiązań modulo n, gdzie $d = \gcd(a, n)$, albo wcale nie ma rozwiązań.

Niech $d = \gcd(a, n)$ i załóżmy, że $d = a \cdot x' + n \cdot y'$ dla pewnych całkowitych x' i y' (na przykład wyliczonych za pomocą procedury *RozszerzonyEuklides*). Jeśli $d | b$, to jednym z rozwiązań równania $a \cdot x \equiv b (\bmod\ n)$ jest wartość x_0, określona wzorem $x_0 = x' \cdot (\frac{b}{d}) \bmod n$.

Jeśli założyć, że równanie $a \cdot x \equiv b (\bmod\ n)$ jest rozwiązywalne (to znaczy $d | b$, gdzie $d = \gcd(a, n)$) i że x_0 jest dowolnym rozwiązaniem tego równania, to wówczas równanie to ma dokładnie d różnych rozwiązań modulo n, danych wzorem $x_i = x_0 + i \cdot (\frac{n}{d})$ dla $i = 0, 1, \ldots, d - 1$.

Cały aparat matematyczny, potrzebny do rozwiązania równania $a \cdot x \equiv b (\mathbf{mod}\ n)$ został już zaprezentowany. Teraz można przeanalizować poniższy algorytm, wypisujący wszystkie rozwiązania tego równania. Dane wejściowe a i n są dowolnymi dodatnimi liczbami całkowitymi, b zaś jest dowolną liczbą całkowitą.

```
1 ModularLinearEquationSolver(a, b, n)
2 (d, x', y') ← RozszerzonyEuklides(a, n)
3 if d|b
4     then x_0 ← x' · (b/d) mod n
5          for i ← 0 to d − 1
6              wypisz (x_0 + i · (n/d)) mod n
7     else wypisz "Brak rozwiązań"
```

Jako przykład działania tej procedury można rozważyć równanie $14 \cdot x \equiv 30 (\mathbf{mod}\ 100)$ ($a = 14$, $b = 30$ i $n = 100$). Wywołując w wierszu 1 procedurę *RozszerzonyEuklides*, otrzymujemy $(d, x, y) = (2, -7, 1)$. Ponieważ $2 | 30$, wykonywane są wiersze 3-5. W wierszu 3 jest obliczane $x_0 = (-7) \cdot (15) \bmod 100 = 95$. W pętli w wierszach 4-5 są wypisywane dwa rozwiązania: 95 i 45.

Procedura *ModularLinearEquationSolver* działa w następujący sposób: w wierszu 1 jest obliczane $d = \mathbf{gcd}(a, n)$, jak również wartości x' i y' takie, że $d = a \cdot x' + n \cdot y'$, co świadczy o tym, że x' jest rozwiązaniem równania $a \cdot x' \equiv d (\mathbf{mod}\ n)$. Jeśli d nie dzieli b, to równanie $a \cdot x \equiv b (\mathbf{mod}\ n)$ nie ma rozwiązań. W wierszu 2 jest sprawdzane, czy $d | b$; jeśli nie, w wierszu 6 jest wypisywana informacja o braku rozwiązań. W przeciwnym razie w wierszu 3 jest obliczane rozwiązanie x_0 równania $a \cdot x \equiv b (\mathbf{mod}\ n)$. Gdy jest jedno rozwiązanie, pozostałych $d - 1$ rozwiązań można otrzymać, dodając do niego wielokrotności $\frac{n}{d}$, modulo n. Pętla for w wierszach 4-5 wypisuje wszystkich d rozwiązań, poczynając od x_0, przy czym kolejne są odległe od siebie o $\frac{n}{d}$, modulo n.

Dla dowolnego $n > 1$, jeśli $\mathbf{gcd}(a, n) = 1$, to równanie $a \cdot x \equiv b (\mathbf{mod}\ n)$ ma dokładnie jedno rozwiązanie modulo n. W często spotykanym i ważnym przypadku gdy $b = 1$, element x, którego szukamy, jest **multyplikatywną odwrotnością** a modulo n. W przeciwyym razie równanie nie ma rozwiązania.

Wniosek taki pozwala na używanie oznaczenia $(a^{-1} \bmod n)$ w odniesieniu do jedynej multyplikatywnej odwrotności a modulo n, jeśli a i n są względnie pierwsze. Jeśli $\mathbf{gcd}(a, n) = 1$, to rozwiązaniem równania $a \cdot x \equiv b (\mathbf{mod}\ n)$ jest liczba x zwracana przez procedurę *RozszerzonyEuklides*, bo z równania $\mathbf{gcd}(a, n) = 1 = a \cdot x + n \cdot y$ wynika $a \cdot x = 1 (\mathbf{mod}\ n)$. Zatem $(a^{-1} \bmod n)$ można efektywnie obliczać za pomocą rozszerzonego algorytmu Euklidesa.

4 Symbol Legendre'a

Symbol Legendre'a $L(a, p)$ jest zdefiniowany, jeśli a jest dowolną liczbą całkowitą, a p jest liczbą pierwszą większą niż 2. Może on przyjmować wartości $0, 1$ lub -1:

$L(a, p) = 0$, jeśli a jest liczbą podzielną przez p; $L(a, p) = 1$, jeśli a jest resztą kwadratową mod p;
$L(a, p) = -1$, jeśli a jest nieresztą kwadratową mod p.

$L(a,p)$ można również łatwo obliczyć ze wzoru: $L(a,p) = a^{\frac{p-1}{2}} \bmod p$
Można też użyć następującego algorytmu:

1. Jeśli $a = 1$, to $L(a,p) = 1$.
2. Jeśli a jest parzyste, to $L(a,p) = L(\frac{a}{2},p) \cdot (-1)^{\frac{p^2-1}{8}}$.
3. Jeśli a jest nieparzyste (oraz $a \neq 1$), to $L(a,p) = L(p \bmod a, a) \cdot (-1)^{\frac{(a-1)\cdot(p-1)}{4}}$.

Jest to również efektywna metoda rozstrzygnięcia, czy a jest resztą kwadratową mod p (gdy p jest liczbą pierwszą).

4.1 Symbol Jacobiego

Symbol Jacobiego $J(a,n)$ jest uogólnieniem symbolu Legendre'a: jest zdefiniowany dla dowolnej pary liczb całkowitych a oraz n. Funkcja ta pojawia się przy badaniu, czy dana liczba jest liczbą pierwszą. Symbol Jacobiego jest funkcją określoną na zbiorze zredukowanych reszt dzielników liczby n i wartości jej mogą być określane za pomocą następującej metody:

- Definicja 1: $J(a,n)$ jest zdefiniowana tylko dla n nieparzystych.
- Definicja 2: $J(0,n) = 0$.
- Definicja 3: Jeśli n jest liczbą pierwszą, to $J(a,n) = 0$, gdy n jest podzielnikiem a.
- Definicja 4: Jeśli n jest liczbą pierwszą, to $J(a,n) = 1$, gdy a jest resztą kwadratową modulo n.
- Definicja 5: Jeśli n jest liczbą pierwszą, to $J(a,n) = -1$, gdy a jest niereszą kwadratową modulo n.
- Definicja 6: Jeśli n jest liczbą złożoną, to $J(a,n) = J(a,p_1) \cdot \ \ldots \ \cdot J(a,p_m)$, przy czym $p_1, \ldots, p_m$ są czynnikami pierwszymi n.

Rekurencyjny algorytm obliczania wartości symbolu Jacobiego ma następującą postać:

- Reguła 1. $J(1,n) = l$
- Reguła 2. $J(a \cdot b, n) = J(a,n) \cdot J(b,n)$
- Reguła 3. $J(2,n) = 1$, jeśli $\frac{n^2-l}{8}$ jest parzyste, $J(2,n) = -1$ w przeciwnym przypadku
- Reguła 4. $J(a,n) = J((a \bmod n), n)$
- Reguła 5. $J(a, b_1, \cdot b_2) = J(a_1, b_1) \cdot J(a_1, b_2)$
- Reguła 6. Jeśli $\gcd(a,b) = 1$ oraz a i b są nieparzyste, to:
  - $J(a,b) = J(b,a)$, gdy $\frac{(a-1)\cdot(b-1)}{4}$ jest parzyste
  - $J(a,b) = -J(b,a)$, gdy $\frac{(a-1)\cdot(b-1)}{4}$ jest nieparzyste.

Jeśli n jest liczbą pierwszą znaną wcześniej, to wystarczy po prostu obliczyć $a^{\frac{n-1}{2}}$ zamiast stosować pokazany algorytm. W tym przypadku $J(a,n)$ jest równoważny z symbolem Lagrange'a.

Symbolu Jacobiego nie można zastosować do rozstrzygnięcia, czy a jest resztą kwadratową mod n (o ile n jest liczbą pierwszą). Wystarczy zauważyć, że jeśli $J(a,n) = 1$ i n jest liczbą złożoną, to niekoniecznie prawdziwe jest stwierdzenie, że a jest resztą kwadratową modulo n. Na przykład:

$$J(7,143) = J(7,11) \cdot J(7,13) = (-1) \cdot (-1) = 1$$

Nie ma jednak takiej liczby całkowitej x, że $x^2 = 7 (\text{mod } 143)$.

5 Chińskie twierdzenie o resztach

Około roku 100 n.e. chiński matematyk Sun-Tsu rozwiązał problem znajdowania tych liczb całkowitych, które przy dzieleniu przez 3, 5 i 7 dają reszty, odpowiednio, 2, 3 i 2. Jednym z takich rozwiązań jest $x = 23$, pozostałe są postaci $23 + 105 * k$ dla dowolnej liczby całkowitej k. Tak zwane "chińskie twierdzenie o resztach" wskazuje zależność między układem równań modulo pewien zbiór liczb parami względnie pierwszych (np. $3, 5, 7$) a równaniem modulo ich iloczyn (tutaj 105).

Niech $n = n_1 n_2 \dots n_k$, gdzie n, są parami względnie pierwsze. Rozważmy współzależność

$$a \leftrightarrow (a_1, a_2, \dots, a_k) \tag{4}$$

gdzie $a \in \mathbb{Z}_n, a_i \in \mathbb{Z}_{n_i}$ oraz $a_i = a \bmod n$. dla $i = 1, 2, \dots, k$. Wówczas odwzorowanie 4 stanowi wzajemnie jednoznaczną odpowiedniość (bijekcję) między $\mathbb{Z}_n$ a iloczynem kartezjańskim $\mathbb{Z}_{n_1} \times \mathbb{Z}_{n_2} \times \mathbb{Z}_{n_k}$. Działania wykonywane na elementach $\mathbb{Z}_n$ można równoważnie wykonywać na odpowiadających im k-ciągach, na każdej współrzędnej niezależnie, wykonując działania z danego systemu. To znaczy, jeśli

$$a \leftrightarrow (a_1, a_2, \dots, a_k)$$

$$b \leftrightarrow (b_1, b_2, \dots, b_k)$$

to

$$\begin{aligned} (a+b) \bmod n &\leftrightarrow ((a_1+b_1) \bmod n_1, \dots, (a_k+b_k) \bmod n_k), \\ (a-b) \bmod n &\leftrightarrow ((a_1-b_1) \bmod n_1, \dots, (a_k-b_k) \bmod n_k), \\ (a \cdot b) \bmod n &\leftrightarrow ((a_1 \cdot b_1) \bmod n_1, \dots, (a_k \cdot b_k) \bmod n_k), \end{aligned}$$

Jeśli $n_1, n_2, \dots, n_k$ są parami względnie pierwsze i $n = n_1 \cdot n_2 \cdot \dots \cdot n_k$, to dla dowolnych liczb całkowitych $a_1, a_2, \dots, a_k$ układ równań

$$x \equiv a_i (\bmod\ n_i)$$

dla $i = 1, 2, \dots, k$ ma dokładnie jedno rozwiązanie modulo n ze względu na niewiadomą x. Ponadto dla dowolnych liczb całkowitych x i a

$$x \equiv a (\bmod\ n)$$

dla $i = 1, 2, \dots, k$ wtedy i tylko wtedy, gdy $x = a (\bmod\ n)$.

Przykład zastosowania chińskiego twierdzenia o resztach - dwa równania:

$$a \equiv 2 (\bmod\ 5)$$

$$a \equiv 3 (\bmod\ 13)$$

skąd $a_i = 2, n_1 = m_2 = 5, a_2 = 3$ oraz $n_2 = m_1 = 13$. Należy obliczyć $a \bmod 65$), bo $n = 65$. Ponieważ $13^{-1} \equiv 2 (\bmod\ 5)$ i $5^{-1} = 8 (\bmod\ 13)$, więc otrzymuje się:

$$c_1 = 13 \cdot (2 \bmod 5) = 26$$

$$c_2 = 5 \cdot (8 \bmod 13) = 40$$

oraz

$$\begin{aligned} a &\equiv 2 \cdot 26 + 3 \cdot 40 \pmod{65} \\ &\equiv 52 + 120 \pmod{65} \\ &\equiv 42 \pmod{65} \end{aligned}$$

Dodatkową ilustracją zastosowania chińskiego twierdzenia o resztach do obliczeń modulo 65 może być następująca tabela:

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	0	40	15	55	30	5	45	20	60	35	10	50	25
1	26	1	41	16	56	31	6	46	21	61	36	11	51
2	52	27	2	42	17	57	32	7	47	22	62	37	12
3	13	53	28	3	43	18	58	33	8	48	23	63	38
4	39	14	54	29	4	44	19	59	34	9	49	24	64

Powyższa tabela ilustruje chińskie twierdzenie o resztach dla $n_1 = 5$ i $n_2 = 13$. W tym przykładzie $c_1 = 26$, a $c_2 = 40$. W i-tym wierszu i j-tej kolumnie znajduje się taka wartość a modulo 65, że $(a \bmod 5) = i$ oraz $(a \bmod 13) = j$. Na przecięciu wiersza 0 i kolumny 0 jest wartość 0. Podobnie, w wierszu 4 i kolumnie 12 znajduje się 64 (równoważne -1). Ponieważ $c_1 = 26$, przejście o jeden wiersz w dół powoduje zwiększenie wartości a o 26. Analogicznie, $c_2 = 40$ oznacza, że przejście w prawo o jedną kolumnę powoduje zwiększenie a o 40. Zwiększenie a o 1 odpowiada przejściu po skosie w dół i w prawo, przy zawinięciu dolnego brzegu tabeli na górny i prawego na lewy.

6 Potęgi elementu

Można rozważać wielokrotności danego elementu a modulo n, analogicznie można rozważać ciąg potęg a modulo n, gdzie $a \in \mathbb{Z}_n^*$: $a^0, a^1, a^2, \ldots$ modulo n. Numerację ciągu zaczyna się od 0; zerową wartością ciągu jest $a^0 \bmod n = 1$, a i-tą wartością jest $a^i \bmod n$. Na przykład potęgami 3 modulo 7 są

i	0	1	2	3	4	5	6	7	8	9	10	11	...
$3^i \bmod 7$	1	3	2	6	4	5	1	3	2	6	4	5	...

a potęgami 2 modulo 7 są

i	0	1	2	3	4	5	6	7	8	9	10	11	...
$2^i \bmod 7$	1	2	4	1	2	4	1	2	4	1	2	4	...

W sekcji opisującej potęgi elementu $\langle a \rangle$ oznacza podgrupę $\mathbb{Z}_n^*$, generowaną przez a, natomiast $\mathrm{ord}(a)$ („rząd elementu a modulo n") oznacza rząd a w $\mathbb{Z}_n^*$. Na przykład $\langle 2 \rangle = \{1, 2, 4\}$ w $\mathbb{Z}_n^*$, a $\mathrm{ord}_7(2) = 3$.

Twierdzenie 4 (Twierdzenie Eulera) *Dla dowolnej liczby całkowitej $n > 1$ zachodzi zależność*

$$a^{\phi(n)} \equiv 1 \pmod{n} \ \forall\ a \in \mathbb{Z}_{\ltimes}^*.$$

Twierdzenie 5 (Twierdzenie Fermata) *Jeśli p jest liczbą pierwszą, to $a^{p-l} \equiv 1 \pmod{p}$ dla każdego $a \in \mathbb{Z}^*$.*

Twierdzenie 6 *Wartości $n > 1$, dla których grupa $\mathbb{Z}^*$ jest cykliczna, to 2, 4, p^e i $2p^e$ dla wszystkich nieparzystych liczb pierwszych p i dowolnych dodatnich liczb całkowitych e.*

Jeśli g jest pierwiastkiem pierwotnym $\mathbf{Z}^*$, zaś a jest dowolnym elementem $\mathbf{Z}^*$, to istnieje z takie, że $g^z \equiv a \pmod{n}$. Takie z nazywamy logarytmem dyskretnym lub indeksem a modulo n przy podstawie g; wartość tę oznaczamy przez $\mathrm{ind}_{n,g}(a)$.

Twierdzenie 7 (Twierdzenie o logarytmie dyskretnym) *Jeśli g jest pierwiastkiem pierwotnym $\mathbf{Z}^*$, to równanie $g^x \equiv g^y \pmod{n}$ jest spełnione wtedy i tylko wtedy, gdy zachodzi $x \equiv y \pmod{\phi(n)}$.*

Twierdzenie 8 *Jeśli p jest nieparzystą liczbą pierwszą i $e \geq 1$, to równanie $x^2 \equiv 1 \pmod{p^e}$ ma dokładnie dwa rozwiązania, a mianowicie $x = 1$ i $x = -1$.*

Liczba x jest nietrywialnym pierwiastkiem kwadratowym z 1 modulo n, jeśli spełnia równanie $x^2 \equiv 1 \pmod{n}$, ale x nie przystaje do żadnego z „trywialnych" pierwiastków kwadratowych: 1 ani -1 modulo n. Na przykład 6 jest nietrywialnym pierwiastkiem kwadratowym z 1 modulo 35.

Jeśli istnieje nietrywialny pierwiastek kwadratowy z 1 modulo n, to n jest liczbą złożoną.

6.1 Potęgowanie przez wielokrotne podnoszenie do kwadratu

W obliczeniach teorioliczbowych często występującą operacją jest podnoszenie jednej liczby do potęgi modulo druga liczba, zwane też potęgowaniem modularnym. Dokładniej, trzeba w efektywny sposób obliczać $a \cdot b \bmod n$, gdzie a i b są nieujemnymi liczbami całkowitymi, a n jest dodatnią liczbą całkowitą. Potęgowanie modularne jest podstawową operacją w wielu procedurach sprawdzania, czy dana liczba jest liczbą pierwszą, a także w RSA - kryptosystemie z kluczem publicznym. Metoda wielokrotnego podnoszenia do kwadratu stanowi efektywne rozwiązanie tego problemu z wykorzystaniem binarnej reprezentacji b.

Niech $\langle b_k, b_{k-1}, \ldots, b_1, b_0 \rangle$ będzie binarną reprezentacją b, składającą się z $k+1$ bitów, z których b_k jest najbardziej, a b_0 najmniej znaczący. Poniższa procedura oblicza $a \cdot c \bmod n$ dla wartości c zwiększających się od 0 do b przez podwajanie i zwiększanie o 1.

```
1  ModularExpotentiation(a, b, n)
2  c ← 0
3  d ← 1
4  niech ⟨b_k, b_{k−1}, ..., b_0⟩ będzie binarną reprezentacją b
5  for i ← k downto 0
6      do c ← 2 · c
7         d ← (d · d) mod n
8         if b_i = 1
9            then c ← c + 1
10                d ← (d · a) mod n
11 return d
```

Poniższa tabela przedstawia wyniki działania procedury ModularExpotentiation, obliczającej $a^b \bmod n$), gdzie $a = 7$, $b = 560 = \langle 1000110000 \rangle$ i $n = 561$:

i	9	8	7	6	5	4	3	2	1	0
b_i	1	0	0	0	1	1	0	0	0	0
c	1	2	4	8	17	35	70	140	280	560
d	7	49	157	526	160	241	298	166	67	1

Tabela przedstawia wartości po każdym przebiegu pętli for. Ostatecznym wynikiem jest 1.

7 Sprawdzanie czy dana liczba jest pierwsza

7.1 Gęstość rozmieszczenia liczb pierwszych

W wielu zastosowaniach, takich, jak kryptografia, występuje konieczność znajdowania dużych pseudolosowych liczb pierwszych. Na szczęście duże liczby pierwsze nie są szczególnie rzadkie, dlatego nie jest czasochłonne sprawdzanie losowo wybranych liczb całkowitych odpowiedniego rozmiaru, dopóki nie zostanie znaleziona liczba pierwsza. Liczb pierwszych jest bardzo dużo (np jest ponad 10^{151} liczb pierwszych 512-bitowych), więc nie należy się obawiać, że zbiór liczb pierwszych ulegnie wyczerpaniu i zaczną się one powtarzać. Dla porównania, we Wszechświecie jest tylko 10^{77} atomów. Jeśli każdy atom we Wszechświecie potrzebowałby miliarda nowych liczb pierwszych co jedną mikrosekundę, począwszy od początku świata aż do chwili obecnej, to wykorzystane zostałoby tylko 10^{109} liczb pierwszych 512-bitowych

Funkcja rozmieszczenia liczb pierwszych $\pi(n)$ określa, ile jest liczb pierwszych mniejszych bądź równych n. Na przykład $\pi(10) = 4$, bo istnieją 4 liczby pierwsze mniejsze bądź równe 10, a mianowicie $2, 3, 5$ i 7. Twierdzenie o liczbach pierwszych daje użyteczne oszacowanie funkcji $\pi(n)$.

Twierdzenie 9 (Twierdzenie o liczbach pierwszych)

$$\lim_{x \to \infty} \frac{\pi(n)}{\frac{n}{\ln n}} = 1 \tag{5}$$

Przybliżenie $\frac{n}{\ln n}$ daje dość dokładne oszacowanie wartości $\pi(n)$ już dla niewielkich n. Na przykład dla $n = 10^9$ różnica wynosi mniej niż 6%, bo $\pi(n) = 50\ 847\ 478$, a $\frac{n}{\ln n} = 48\ 254\ 942$, przy czym warto wspomnieć, że dla naukowca zajmującego się teorią liczb 10^9 to mała liczba.

Korzystając z twierdzenia o liczbach pierwszych, prawdopodobieństwo tego, że losowo wybrana liczba n okaże się liczbą pierwszą, możemy oszacować przez $\frac{1}{\ln n}$. W celu znalezienia liczby pierwszej tej samej długości co n niezbędne będzie zatem przebadanie około $\ln n$ losowo wybranych liczb w pobliżu n. Na przykład znalezienie 512-bitowej liczby pierwszej wymagałoby sprawdzenia około $\ln\ 2^{512} \approx 355$ losowo wybranych liczb 512-bitowych. Liczbę tę można zmniejszyć o połowę, wybierając tylko liczby nieparzyste.

Najprostsze podejście do problemu sprawdzania, czy liczba jest liczbą pierwszą, polega na dzieleniu do skutku. W tym celu należy próbować dzielić n przez kolejne liczby $2, 3, \ldots, \lfloor \sqrt{n} \rfloor$, przy czym oczywiście liczby parzyste większe od 2 można pominąć. Łatwo sprawdzić, że liczba n jest liczbą pierwszą wtedy i tylko wtedy, gdy nie dzieli się przez żadną z nich. Przy założeniu, że każde takie dzielenie zajmuje czas stały, czas sprawdzania w najgorszym przypadku wynosi $\theta(u)$, jest więc wykładniczy ze względu na długość liczby n.

Metoda dzielenia do skutku działa zatem dobrze jedynie wtedy, gdy n jest bardzo małe albo ma mały czynnik pierwszy. Jeśli metoda działa, to ma przy tym tę zaletę, że nie tylko uzyskuje się odpowiedź, czy n jest liczbą pierwszą czy złożoną, ale w tym drugim przypadku otrzymuje się także rozkład na czynniki pierwsze.

7.2 Sprawdzanie, czy dana liczba jest pseudopierwsza

Niech $\mathbb{Z}^+$ oznacza zbiór niezerowych elementów $\mathbb{Z}_n$:

$$\mathbb{Z}_n^+ = 1, 2, \ldots, n-1$$

Jeśli n jest liczbą pierwszą, to $\mathbb{Z}_n^+ = \mathbb{Z}_n^*$.

Mówimy, że liczba n jest **pseudopierwsza** przy podstawie a, jeśli n jest liczbą złożoną oraz

$$a^{n-1} \equiv 1 (\bmod\ n) \tag{6}$$

Z twierdzenia Fermata wynika, że jeśli n jest liczbą pierwszą, to spełnia powyższe równanie dla każdego a należącego do $\mathbb{Z}^+$. Jeśli zatem możemy znaleźć jakiekolwiek $a \in \mathbb{Z}^+$ takie, że n nie spełnia równania 6, to liczba n na pewno jest złożona. Co zadziwiające, stwierdzenie odwrotne jest prawie zawsze spełnione; kryterium to stanowi zatem dobry test rozstrzygający, czy liczba jest liczbą pierwszą. Wystarczy sprawdzić, czy n spełnia równanie 6 dla $a = 2$. Jeśli nie, n jest liczbą złożoną, a w przeciwnym razie n jest liczbą pierwszą (chociaż tak naprawdę wiadomo jedynie, że liczba n jest albo pierwsza, albo pseudopierwsza przy podstawie 2).

Poniższa procedura w taki właśnie sposób udaje, że sprawdza, czy n jest liczbą pierwszą. Korzysta ona z procedury ModularExpotentiation. O wejściowej liczbie n zakładamy, że jest nieparzysta i większa od 2.

```
1 Pseudopierwsza(n)
2 if ModularExpotentiation(2, n − 1, n) ≡ 1(mod n)
3 then return Złożona              ▷ Na pewno!
4 else return Pierwsza             ▷ Prawdopodobnie!
```

Procedura ta może popełniać błędy, ale tylko jednego rodzaju. Jeśli udziela odpowiedzi, że n jest liczbą złożoną, jest to zawsze prawda. Jeśli oświadcza, że n jest liczbą pierwszą, to myli się tylko wtedy, gdy n jest liczbą pseudopierwszą przy podstawie 2.

Jak często procedura udziela błędnej odpowiedzi? Okazuje się, że wśród n mniejszych niż 10000 są tylko 22 wartości, dla których się to zdarza; pierwsze cztery z nich to 341, 561, 645 i 1105. Można pokazać, że prawdopodobieństwo popełnienia błędu przez powyższy program dla losowo wybranej liczby β-bitowej dąży do zera przy $\beta \to \infty$. Szacując dokładniej, jest mniej niż jedna szansa na 1020, że losowo wybrana liczba 512-bitowa, którą powyższa procedura określi jako pierwszą, okaże się pseudopierwsza przy podstawie 2, a dla losowo wybranej liczby 1024-bitowej szansa ta jest mniejsza niż jeden na 1041. Jeśli zatem próbujemy jedynie znaleźć dużą liczbę pierwszą do jakiegoś celu, to z praktycznego punktu widzenia prawie nigdy nie popełnimy błędu, wybierając losowo duże liczby tak długo, aż dla jednej z nich procedura *Pseudopierwsza* zwróci w wyniku, że jest to liczba pierwsza. Jeśli jednak sprawdzane liczby nie są wybierane losowo, potrzebny jest lepszy sposób testowania, czy są one liczbami pierwszymi.

Niestety, nie uda się wyeliminować wszystkich błędów, sprawdzając po prostu, czy spełnione jest równanie 6 dla kolejnej podstawy, np $a = 3$, ponieważ istnieją liczby złożone n, które spełniają równanie 6 dla każdego $a \in \mathbb{Z}_n^*$. Liczby te znane są pod nazwą **liczb Carmichaela**. Pierwsze trzy liczby Carmichaela to 561, 1105 i 1729. Liczby Carmichaela występują niesłychanie rzadko; na przykład wśród liczb niniejszych od 100000000 jest ich zaledwie 255.

7.3 Probabilistyczny test Millera-Rabina sprawdzający, czy liczba jest pierwsza

Test Millera-Rabina sprawdzający, czy liczba jest pierwsza, jest ulepszoną wersją testu Pseudopierwsza. Zmodyfikowano dwie cechy tej funkcji:

- Sprawdzana jest nie jedna, lecz kilka losowo wybranych wartości podstawy a;
- Podczas obliczania każdego modularnego potęgowania algorytm sprawdza, czy nie został wykryty jakiś nietrywialny pierwiastek kwadratowy z 1 modulo n. Jeśli tak, to zwraca odpowiedź, że testowana liczba jest złożona.

Do procedury *Miller-Rabin* na wejście podawana jest nieparzysta liczba $n > 2$. Zadaniem funkcji jest rozstrzygnięcie, czy jest to liczba pierwsza. Drugi parametr s zaś jest liczbą losowo wybranych wartości podstawy ze zbioru $\mathbb{Z}_n^+$ do sprawdzenia. Procedura korzysta z generatora liczb losowych Random, wywoływanego Random(1, n - 1) i zwracającego losowo wybraną liczbę całkowitą a taką, że $1 \leq a \leq n - 1$. Zastosowano także pomocniczą funkcję *Witness*. Wartością funkcji *Witness(a, n)* jest *True* wtedy i tylko wtedy, gdy a jest "świadectwem" złożoności liczby n, czyli jeśli można użyć a do udowodnienia, że n jest liczbą złożoną. Test *Witness(a, n)* stanowi skuteczniejsze rozszerzenie testu sprawdzającego, czy

$$a^{n-1} \equiv 1 (\text{mod } n) \tag{7}$$

na którym opierała się (dla $a = 2$) procedura *Pseudopierwsza*.

Niech $n - 1 = 2^t \cdot u$, gdzie $t \geq 1$ i u jest nieparzyste; tzn. reprezentacja binarna liczby $n - 1$ jest równa reprezentacji binarnej liczby nieparzystej u, po której następuje dokładnie t zer. Zatem $a^{n-l} \equiv (a^u)^{2^t} (\text{mod } n)$, czyli można obliczyć a^{n-l} mod n, obliczając najpierw a^u mod n, a następnie kolejno podnosząc do kwadratu uzyskany wynik t razy.

```
1  Witness(a, n)
2  niech n-1 = 2^t · u, gdzie t>1 i u jest nieparzyste
3  x_0 ← ModularExpotentiation(a,u,n)
4  for i ← 1 to f
5       do x_i ← x_{i-1}^2 ≠ 1 i x_{i-1} ≠ n-1
6       if x_i = 1 i x_{i-1} ≠ 1 i x_{i-1} ≠ n-1
7            then return True
8  if x_t ≠ 1
9  then return True
10 return False
```

W procedurze *Witness* obliczane jest $a^{n-1} \bmod n$ poprzez wyznaczenie wartości $x_0 = a^u \bmod n$ w wierszu 2, a następnie t-krotne podnoszenie wyniku do kwadratu w pętli **for** w wierszach 3-6. Łatwo wykazać, że ciąg obliczanych wartości $x_0, x_1, \ldots, x_t$ spełnia równanie $x_i \equiv a^{2^i * u} \bmod n$, dla $i = 0, 1, \ldots, t$, więc w szczególności $x_t \equiv a^{n-1} \bmod n$). Po każdym kroku podnoszenia do kwadratu w wierszu 4 pętla może się zakończyć wcześniej, jeśli w wierszach 5-6 okaże się, że został wykryty nietrywialny pierwiastek kwadratowy z 1. Jeśli tak jest, to algorytm zatrzymuje się i zwraca wartość *True*. W wierszach 7-8 jest zwracana wartość *True*, jeśli obliczona wartość $x_i = a^{n-1} (\bmod\ n)$ nie jest równa 1, podobnie jak w procedurze *Pseudopierwsza*, zwracającej w tym przypadku wartość złożona. W wierszu 9 jest zwracana wartość false, jeśli wcześniej nie została zwrócona wartość *True* w wierszach 6 ani 8.

Jeśli wywołanie procedury *Witness(a,n)* zwraca wartość *True*, to n na pewno jest liczbą złożoną, co można łatwo wykazać korzystając z liczb a i n. Test Millera-Rabina sprawdzający, czy liczba jest pierwsza, który korzysta z procedury *Witness* ma postać podaną poniżej. Tak jak wcześniej, zakłada się, że liczba n jest nieparzysta i większa od 2.

```
1  Miller-Rabin(n, s)
2  for j ← 1 to s
3       do a ← Random(1, n - 1)
4       if Witness (a, n)
5            then return Złożona      ▷ Na pewno.
6  return pierwsza                   ▷ Prawie na pewno.
```

Procedura Miller-Rabin polega na probabilistycznym poszukiwaniu dowodu, że liczba n jest złożona. W głównej pętli (zaczynającej się w wierszu 1) jest wybieranych s losowych wartości a ze zbioru $\mathbb{Z}_n^+$ (wiersz 2). Jeśli jedna z wybranych wartości a jest świadectwem złożoności n, to procedura *Miller-Rabin* zwraca wartość *Złożona* w wierszu 4. Odpowiedź taka jest zawsze zgodna z prawdą, co wynika z poprawności procedury *Witness*. Jeśli po 5 próbach nie udało się znaleźć żadnego świadectwa, procedura *Miller-Rabin* przyjmuje, iż stało się tak, ponieważ takich świadectw w ogóle nie ma, bo n jest liczbą pierwszą. Odpowiedź taka jest z dużym prawdopodobieństwem prawdziwa, jeśli s jest dostatecznie duże, ale istnieje niewielka szansa, że to wybór wartości a był pechowy, a świadectwa istnieją, chociaż żadne nie zostało znalezione.

Działanie procedury *Miller-Rabin* zilustrowane zostanie na przykładzie, w którym n jest liczbą Carmichaela 561, więc $n-1 = 560 = 2^4 * 35$. Jako podstawa została wybrana liczba $a = 7$. Procedura *Witness* oblicza $x_0 \equiv a^{35} \equiv 241 (\bmod 561)$, a następnie oblicza ciąg $X = \langle 241, 298, 166, 67, 1 \rangle$. W ostatnim kroku podnoszenia do kwadratu zostaje zatem odkryty nietrywialny pierwiastek kwadratowy z 1, ponieważ $a^{280} \equiv 67 (\bmod\ n)$, zaś $a^{560} = 1 (\bmod\ n)$. Liczba $a = 7$ jest zatem świadectwem złożoności n, wywołanie *Witness(7, n)* zwraca wartość *True*, a odpowiedź procedury *Miller-Rabin* brzmi *Złożona*.

7.3.1 Prawdopodobieństwo błędu w teście Millera-Rabina

Jeśli procedura *Miller-Rabin* udziela odpowiedzi Pierwsza, to istnieje niewielka szansa, że popełniła ona błąd. Jednak w odróżnieniu od procedury *Pseudopierwsza* prawdopodobieństwo błędu nie zależy od n; w tym wypadku nie ma żadnych złych danych wejściowych. Możliwość powstania błędu zależy raczej od liczby prób s oraz „szczęścia w losowaniu" podczas wybierania wartości podstawy a. Oprócz tego, ponieważ każdy z testów jest silniejszy niż proste sprawdzenie równania 6, ogólnie można się spodziewać, że procent błędów dla losowo wybieranych liczb n powinien być niewielki. Poniższe twierdzenie uściśla te oczekiwania.

Jeśli n jest nieparzystą liczbą złożoną, to liczba świadectw złożoności n wynosi przynajmniej $\frac{n-l}{2}$.

Dla dowolnej nieparzystej liczby $n > 2$ i dodatniej liczby całkowitej s prawdopodobieństwo tego, że procedura *Miller-Rabin(n, s)* udzieli błędnej odpowiedzi, nie przekracza 2^s.

7.4 Test Solovaya - Strassena

Robert Solovay i Volker Strassen opracowali probabilistyczny algorytm testowania liczb, mający na celu ustalenie, czy są one liczbami pierwszymi. Algorytm ten wykorzystuje symbol Jacobiego do testowania, czy dana liczba p jest liczbą pierwszą.

Algorytm postępowania:

```
1 (1) Wybieramy liczbę losową a mniejszą niż p.
2 (2) Jeśli gcd(a,p)·1, to p jest liczbą złożoną i nie przechodzi testu.
3 (3) Oblicz j = a^((p-1)/2) modp.
4 (4) Oblicz symbol Jacobiego J(a, p).
5 (5) Jeśli j·J(a,p), to p na pewno nie jest liczbą pierwszą.
6 (6) Jeśli j = J(a,p), to prawdopodobieństwo
7         tego, że p nie jest liczbą pierwszą jest najwyżej równe 50%.
```

Liczba a, która określa, że p na pewno jest liczbą złożoną, jest nazywana **świadkiem**. Jeśli p jest liczbą złożoną, to szansa na to, że losowa liczba jest świadkiem jest nie mniejsza niż 50%. Test ten powtarzamy t razy z t różnymi, losowymi wartościami liczby a. Prawdopodobieństwo, że złożona liczba p przejdzie wszystkie n testów, jest nie większe niż 1 do 2^t.

7.5 Test Lehmanna

Test ten, opracowany przez Lehmanna, jest prostszy niż test Solovaya - Strassena. Algorytm sprawdza, czy p jest liczbą pierwszą:

```
1 (1) Wybierz liczbę losową a mniejszą niż p.
2 (2) Oblicz a^((p-1)/2) modp.
3 (3) Jeśli a^((p-1)/2) ≢ 1 lub a^((p-1)/2) ≢ −1(modp), to p na pewno jest liczbą złożoną.
4 (4) Jeśli a^((p-1)/2) ≡ 1 lub a^((p-1)/2) ≡ −1(modp) to prawdopodobieństwo,
5         że p nie jest liczbą pierwszą, nie jest większe niż 50%.
```

Ponowna szansa na to, że liczba losowa a jest świadkiem złożoności p jest nie mniejsza niż 50%. Test należy powtórzyć t razy. Jeśli obliczenia dają 1 lub -1, ale nie są zawsze równe 1, to p jest prawdopodobnie liczbą pierwszą z prawdopodobieństwem popełnienia błędu równym $\frac{1}{2}$.

8 Zadania do wykonania

1. Zaimplementować test Millera-Rabina.
2. Udowodnić, że liczb pierwszych jest nieskończenie wiele.

Literatura

[1] L.Banachowski i in. *Algorytmy i struktury danych*; WNT, 1996.

[2] T.H.Cormen i in. *Wprowadzenie do algorytmów*, WNT, 2000

[3] A.Drozdek *Struktury danych w jezyku C*, WNT, 1996

[4] D.E.Knuth *Sztuka programowania*, WNT, 2002

[5] R.Sedgewick *Algorytmy w C++*, Wydawnictwo RM, 1999

[6] P.Wróblewski, *Algorytmy, struktury danych i techniki programowania*, HELION, 1996

[7] N. Wirth, *Algorytmy + struktury danych = programy*, WNT, 2001

[8] B. Schneier, *Kryptografia dla praktyków*, WNT, 1995-2002