POLITECHNIKA POZNAŃSKA

WYDZIAŁ ELEKTRYCZNY

Instytut Matematyki

PRACA DYPLOMOWA MAGISTERSKA

TESTY PIERWSZOŚCI LICZB I ICH ZASTOSOWANIA W KRYPTOGRAFII

Małgorzata Lipińska

Promotor:
dr hab. Małgorzata Migda

POZNAŃ, 2018

KARTA PRACY DYPLOMOWEJ Z DZIEKANATU
(kserokopia z podpisami)

Podziękowania

Składam serdecznie podziękowania
moim rodzicom,

Spis treści

Wstęp

Spis oznaczeń i symboli

a, b - liczby naturalne

$\mathbb{N}$ - zbiór liczb naturalnych

$\mathbb{Z}$ - zbiór liczb całkowitych

$\mathbb{P}$ - zbiór liczb pierwszych

$a \mid b$ - a dzieli b

$a \nmid b$ - a nie dzieli b

$NWW(a, b)$ - największa wspólna wielokrotność a i b

$NWWD(a, b)$ - największy wspólny dzielnik a i b

$(a, b) = 1$ - a i b są względnie pierwsze

1. Liczby pierwsze

Poniższy rozdział jest zbiorem najważniejszych pojęć i twierdzeń związanych z liczbami pierwszymi oraz ich własnościami.

1.1. Zasadnicze twierdzenie arytmetyki i twierdzenie Euklidesa

Jednym z najważniejszymi twierdzeń dotyczących liczb pierwszych jest zasadnicze twierdzenie arytmetyki. Jego wprowadzenie i udowodnienie wymaga zdefiniowania kilku podstawowych pojęć z zakresu teorii liczb, które zostaną omówione w tym podrozdziale.

Definicja 1.1. [2, s. 3] *Liczba całkowita b jest podzielna przez liczbę całkowitą a*, jeśli istnieje taka liczba całkowita e, że zachodzi $b = ae$. Wtedy liczba a jest dzielnikiem liczby b, z kolei o liczbie b mówi się, że jest wielokrotnością a.

Jeśli liczba b jest podzielna przez a (a dzieli b), to używa się oznaczenia $a \mid b$, natomiast w przeciwnym przypadku stosuje się oznaczenie $a \nmid b$.

Definicja 1.2. *Liczbą pierwszą* nazywa się liczbę naturalną większą od 1, która ma dokładnie dwa dzielniki: jedynkę i samą siebie.

Zbiór wszystkich liczb pierwszych oznacza się literą $\mathbb{P}$. Liczbą złożoną jest więc liczba, która ma więcej niż dwa dzielniki. Warto zauważyć, że 1 nie jest liczbą ani pierwszą, ani złożoną.

Definicja 1.3. *Najmniejsza wspólna wielokrotność* dla zadanych liczb całkowitych to najmniejsza liczba naturalna, którą dzieli się bez reszty przez każdą z tych liczb.

Najmniejszą wspólną wielokrotność liczb a i b oznacza się symbolem $NWW(a, b)$, np. $NWW(2, 15) = 30$.

Definicja 1.4. *Największy wspólny dzielnik* dla zadanych dwóch lub więcej liczb całkowitych to największa liczba naturalna, która dzieli każdą z tych liczb.

Największy wspólny dzielnik liczb a i b oznaczany jest symbolem $NWD(a, b)$, np. $NWD(18, 15) = 3$.

Definicja 1.5. *Liczby względnie pierwsze* to liczby całkowite, których największym wspólnym dzielnikiem jest liczba 1.

Jeśli dwie liczby a i b są względnie pierwsze to zapisuje się to jako $NWD(a,b) = 1$ lub $(a,b) = 1$.

Lemat 1.1. (Euklides) [5, s. 10] Jeśli liczba pierwsza p dzieli iloczyn ab, to dzieli przynajmniej jeden z tych czynników. Inaczej, jeśli $p \in \mathbb{P}$ oraz $p \mid ab$, to $p \mid a$ lub $p \mid b$.

Dowód. Przyjmijmy, że $p \nmid a$, czyli $NWD(a,p) = 1$, gdyż $p \in \mathbb{P}$. Wtedy $p \mid b$.

■

Lemat 1.2. [2, s. 9] Jeżeli $p \in \mathbb{P}$ i $p \mid a_1 \cdot ... \cdot a_n$, to istnieje $k \in \mathbb{N}$ $(1 \leq k \leq n)$ takie, że $p \mid a_k$.

Dowód. Z lematu 1.1 wiemy, że jeżeli $p \in \mathbb{P}$ oraz $p \mid ab$, to $p \mid a$ lub $p \mid b$. Iloczyn $a_1 \cdot ... \cdot a_n$ można przedstawić w postaci $(a_1 \cdot ... \cdot a_{n-1})a_n$. Wtedy $p \mid (a_1 \cdot ... \cdot a_{n-1})a_n$, czyli $p \mid a_1 \cdot ... \cdot a_{n-1}$ lub $p \mid a_n$. Jeżeli $p \mid a_n$, to lemat jest prawdziwy. Jeśli $p \mid a_1 \cdot ... \cdot a_{n-1}$ i $p \nmid a_n$ to powtarzamy powyższe rozumowanie dla $p \mid (a_1 \cdot ... \cdot a_{n-2})a_{n-1}$ i postępujemy podobnie do chwili znalezienia takiego k, że $p \mid a_k$ lub do momentu, gdy przedmiotem naszych rozważań będzie $p \mid a_1 a_2$. Wtedy zgodnie z lematem 1.1 $p \mid a_1$ lub $p \mid a_2$, co kończy dowód.

■

Lemat 1.3. [2, s. 9] Jeśli liczby $p, q_1, ..., q_n \in \mathbb{P}$ oraz $p \mid q_1 \cdot ... \cdot q_n$, to $p = q_k$ dla pewnego k, gdzie $1 \leqslant k \leqslant n$.

Dowód. Z lematu 1.2 wiadomo, że istnieje takie k, że $p \mid q_k$, ale $q_k \in \mathbb{P}$ oraz $p > 1$, skąd wynika, że $p = q_k$.

■

Trzy powyższe lematy to lematy pomocnicze, które posłużą do udowodnienia zasadniczego twierdzenia arytmetyki, mówiącego o tym, że liczby pierwsze są czynnikami, na które można rozłożyć wszystkie złożone liczby naturalne.

Twierdzenie 1.1. *(Zasadnicze twierdzenie arytmetyki) [2, s. 10] Każdą liczbę naturalną większą od 1, która nie jest liczbą pierwszą, można jednoznacznie przedstawić w postaci iloczynu liczb pierwszych.*

Dowód. Dowód powyższego twierdzenia należy podzielić na dwie części. Pierwsza dotyczyć będzie istnienia rozkładu, a druga jego jednoznaczności.

Istnienie rozkładu. Załóżmy, że liczba n jest liczbą złożoną. Wtedy

$$D = \{d \in N, 1 < d < n, d \mid n\}$$

jest niepustym zbiorem dzielników liczby n. Niech p_1 będzie najmniejszą liczbą zawartą w tym zbiorze. Zatem p_1 jest liczbą pierwszą, gdyż w przeciwnym przypadku istniałoby $q < p_1$ takie, że $q > 1$ i $q \mid p_1$, co byłoby sprzeczne z wyborem p_1. Stąd wynika, że

$$n = p_1 \cdot n_1,$$

gdzie $1 < n_1 < n$. Jeżeli liczba $n_1 \in \mathbb{P}$, to żądany rozkład został osiągnięty, w przeciwnym wypadku rozumowanie należy powtórzyć dla n_1, skąd dostajemy

$$n = p_1 \cdot p_2 \cdot n_1,$$

gdzie $1 < n_2 < n_1$, $p_1, p_2 \in \mathbb{P}$. Na koniec uzyskuje się rozkład

$$n = p_1 \cdot ... \cdot p_k,$$

ponieważ $n_1 > n_2 > ... > 1$ jest malejącym ciągiem liczb naturalnych, czyli jest ciągiem skończonym.

Jednoznaczność rozkładu. Załóżmy przeciwnie, że rozkład nie jest jednoznaczny, czyli $n = p_1 \cdot ... \cdot p_r = q_1 \cdot ... \cdot q_s$, $(r \leqslant s)$, gdzie p_i i q_j dla $i = 1, 2, ..., r$ i $j = 1, 2, ..., s$ są pierwsze i uporządkowane niemalejąco. Ponieważ $p_1 \mid q_1 ... q_s$, zatem z Lematu 1.3 wynika, że $p_1 = q_k$ dla pewnego k, gdzie $1 \leqslant k \leqslant s$, skąd wynika, że $p_1 \geqslant q_1$. Analogicznie $q_1 \mid p_1 ... p_r$, więc $q_1 \geqslant p_1$, czyli $p_1 = q_1$. Dzieląc początkową równość przez $p_1 = q_1$ i powtarzając to samo rozumowanie dla kolejnych czynników obu rozkładów, otrzymujemy

$$1 = q_{r+l} \cdot ... \cdot q_s,$$

jeśli $r < s$. Jest to sprzeczne dla $q_{r+l}, ..., q_s \in \mathbb{P}$, a więc $r = s$ oraz $p_i = q_i$ dla każdego $i = 1, 2, ..., r$, czyli rozkład jest jednoznaczny.

■

Z zasadniczego twierdzenia arytmetyki 1.1 wynika bezpośrednio poniższy wniosek.

Wniosek 1.1. [2, s. 10] Każda liczba naturalna $n > l$ może być jednoznacznie zapisana w postaci kanonicznej

$$n = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot ... \cdot p_r^{\alpha_r},$$

gdzie $p_i \in \mathbb{P}$, $\alpha_i \in \mathbb{N}$ dla każdego $i = 1, 2, ..., r$ oraz $p_1 < p_2 < ... < p_r$.

Do obliczania NWW i NWD stosuje się najczęściej rozkład badanych liczb na iloczyn liczb pierwszych.

Przykład 1.1. Obliczymy NWW i NWD liczb 123 i 567.
Rozkład danych liczb na czynniki pierwsze:

$$123 = 3 \cdot 41,$$
$$567 = 3 \cdot 3 \cdot 3 \cdot 3 \cdot 7.$$

Aby obliczyć $NWW(a, b)$ należy pomnożyć wszystkie czynniki pierwsze liczby a i czynniki pierwsze liczby b, które się nie powtarzają przy rozkładzie a. Zatem

$$NWW(123, 567) = 3 \cdot 41 \cdot 3 \cdot 3 \cdot 3 \cdot 7 = 3^4 \cdot 7 \cdot 41 = 23247.$$

$NWD(a, b)$ to iloczyn czynników pierwszych, które powtarzają się w rozkładzie liczby a i b. Stąd

$$NWS(123, 567) = 3.$$

Wniosek 1.2. [5, s. 8] Dla liczb $a, b \in \mathbb{N}$ zachodzi

$$NWD(a, b) \cdot NWW(a, b) = ab.$$

Dowód. Niech

$$a = p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot ... \cdot p_n^{\alpha_n},$$
$$b = p_1^{\beta_1} \cdot p_2^{\beta_2} \cdot ... \cdot p_n^{\beta_n},$$

będą rozkładami liczb a i b na czynniki pierwsze. Wówczas

$$NWD(a, b) = p_1^{min(\alpha_1,\beta_1)} \cdot p_2^{min(\alpha_2,\beta_2)} \cdot ... \cdot p_n^{min(\alpha_n,\beta_n)},$$
$$NWW(a, b) = p_1^{max(\alpha_1,\beta_1)} \cdot p_2^{max(\alpha_2,\beta_2)} \cdot ... \cdot p_n^{max(\alpha_n,\beta_n)}.$$

Po pomnożeniu $NWD(a, b) \cdot NWW(a, b)$ otrzymujemy

$$p_1^{min(\alpha_1,\beta_1)+max(\alpha_1,\beta_1)} \cdot p_2^{min(\alpha_2,\beta_2)+max(\alpha_2,\beta_2)} \cdot ... \cdot p_n^{min(\alpha_n,\beta_n)+max(\alpha_n,\beta_n)} =$$
$$= p_1^{\alpha_1+\beta_1} \cdot p_2^{\alpha_2+\beta_2} \cdot ... \cdot p_n^{\alpha_n+\beta_n} =$$
$$= p_1^{\alpha_1} \cdot p_2^{\alpha_2} \cdot ... \cdot p_n^{\alpha_n} \cdot p_1^{\beta_1} \cdot p_2^{\beta_2} \cdot ... \cdot p_n^{\beta_n} = a \cdot b.$$

■

1.2. Twierdzenie i algorytm Euklidesa

Euklides, żyjący około 300 roku p.n.e., to grecki matematyk pochodzący z Aten, uczeń Akademii Platońskiej i wykładowca w słynnej Szkole Aleksandryjskiej. Zaliczany jest do grona największych matematyków w historii, znany głównie jako twórca podstaw geometrii klasycznej. Jego największym dziełem są *Elementy*, które stanowią jedną z pierwszych prac teoretycznych z matematyki. Podręcznik ten zawiera między innymi dwa klasyczne i bardzo ważne pojęcia z zakresu teorii liczb - twierdzenie Euklidesa o liczbach pierwszych i algorytm pozwalający znaleźć największy wspólny dzielnik dwóch liczb naturalnych.

Twierdzenie 1.2. *(Euklides) [5, s. 5] Istnieje nieskończenie wiele liczb pierwszych.*

Powstało kilkanaście dowodów twierdzenia 1.2. Poniżej przedstawiono pierwszy z nich, który został sformułowany w IV wieku p.n.e. przez Euklidesa w *Elementach.*
Dowód. Załóżmy, że istnieje skończony zbiór liczb pierwszych ponumerowanych kolejno $p_1, p_2, ..., p_k$. Rozważmy liczbę

$$n = p_1 p_2 ... p_k + 1.$$

Z Twierdzenia 1.1 wynika, że n może być liczbą pierwszą różną od wszystkich p_i lub ma rozkład na czynniki pierwsze. Załóżmy, że jednym z tych czynników jest p. Liczba n przy dzieleniu przez każde z p_i daje resztę 1, więc p jest różne od wszystkich p_i. Z tego wynika, że albo p jest pierwsze lub samo n jest kolejną liczbą pierwszą, większą od każdego p_i, co jest sprzeczne z założeniem.

■

Algorytm Euklidesa to szybki sposób wyznaczania największego wspólnego dzielnika dwóch liczb naturalnych.

Do zrozumienia działania powyższego algorytmu potrzebne będzie zdefiniowanie pojęcia reszty z dzielenia.

Definicja 1.6. [5, s. 8] Dla dowolnej liczby całkowitej n i dowolnej liczby naturalnej k istnieje tylko jedna para liczb całkowitych q i r taka, że

$$n = qk + r, \text{ gdzie } 0 \leq r \leq k - 1.$$

Liczbę r nazywa się wtedy *resztą z dzielenia.*

Zasadę działania algorytmu Euklidesa najlepiej wyjaśnić przy pomocy przykładu.

Przykład 1.2. Za pomocą algorytmu Euklidesa wyznaczymy $NWD(1628, 374)$. Działanie algorytmu przebiega następująco:

$$\begin{aligned} 1628 &= 4 \cdot 374 + 132 \\ 374 &= 2 \cdot 132 + 110 \\ 132 &= 1 \cdot 110 + 22 \\ 110 &= 5 \cdot 22 + 0. \end{aligned}$$

Ostatnia niezerowa reszta jest szukanym największym wspólnym dzielnikiem badanych liczb. W przypadku powyższego przykładu $NWD(1628, 374) = 22$.

Lemat 1.4. (Bézout) [5, s. 9] Niech $NWD(a, b) = d$. Wówczas istnieją dwie liczby całkowite k i l takie, że

$$d = ka + lb.$$

Dowód. Niech a i b będą niezerowymi liczbami całkowitymi, a K oznacza zbiór wszystkich dodatnich liczb całkowitych postaci $am + bn$, gdzie $m, n \in \mathbb{Z}$. Zbiór K jest niepusty, więc istnieje w nim element najmniejszy d, gdzie $d = ax + by$. Zgodnie z algorytmem dzielenia z resztą, istnieją takie liczby $q, r \in \mathbb{Z}$, dla których zachodzi $a = qd + r$, przy czym $0 \leq r < d$. Z drugiej strony

$$r = a - qd = a - q(ax + by) = a - aqx - bqy = a(1 - qx) + b(-qy).$$

Jeżeli $0 < r < d$, to znaczy, że $r \in K$, co jest sprzeczne ze stwierdzeniem, że d jest najmniejszym elementem w K. Stąd $r = 0$, a z tego wynika, że $a = qd$, a to oznacza, że $d|a$. Podobnie można wykazać, że $d|b$, skąd wynika, że d jest wspólnym dzielnikiem dla a i b. Jeśli c jest innym wspólnym dzielnikiem liczb a i b, to c dzieli również $ax + by = d$, co z definicji oznacza, że $d = NWD(a, b)$.

■

Wniosek 1.3. Niech $(a, b) = 1$. Wtedy istnieją dwie liczby całkowite k i l takie, że

$$ka + lb = 1.$$

1.3. Kongruencje i ich zastosowania

W poniższym rozdziale zawarte zostały najważniejsze informacje i twierdzenia związane z kongruencją. Będą one wykorzystywane w dalszej części pracy.

Definicja 1.7. [4, s. 41] Niech n będzie dowolną liczbą naturalną. Liczby całkowite a i b *przystają modulo* n, jeżeli ich różnica $a-b$ jest podzielna przez n. Symbolicznie zapisuje się to jako

$$a \equiv b \ (mod\ n).$$

Relację tą nazywamy kongruencją lub przystawaniem modulo.

Jeśli zachodzi kongruencja $a \equiv b \ (mod\ n)$, to b jest resztą z dzielenia liczby a przez n. Jeżeli $a \equiv 0 \ (mod\ n)$, to znaczy, że $n \mid a$.

Twierdzenie 1.3. *Kongruencja jest relacją równoważności. Jest ona*

1. *Zwrotna, ponieważ każda liczba przystaje sama do siebie*

$$a \equiv a \ (mod\ n).$$

2. *Symetryczna, tj.*

$$a \equiv b \ (mod\ n) \Leftrightarrow b \equiv a \ (mod\ n).$$

3. *Przechodnia, ponieważ zachodzi*

$$(a \equiv b \ (mod\ n) \wedge b \equiv c \ (mod\ n)) \Rightarrow a \equiv c \ (mod\ n).$$

Dowód.

1. Wystarczy zauważyć, że $a - a = 0$ jest podzielna przez każde $n \in \mathbb{N}$.
2. Jeżeli $a - b = kn$, gdzie $k \in \mathbb{Z}$, to $b - a = -kn$. Stąd wynika, że $a \equiv b \ (mod\ n)$.
3. Jeżeli $(a \equiv b \ (mod\ n)$ i $b \equiv c \ (mod\ n)$, to $a - b \equiv 0 \ (mod\ n)$ i $b - c \equiv 0 \ (mod\ n)$. Opierając się na tożsamości

$$a - c = (a - b) + (b - c)$$

otrzymujemy, że $a - c \equiv 0 \ (mod\ n)$, skąd wynika, że $a \equiv c \ (mod\ n)$.

■

Twierdzenie 1.4. *[3, s. 37] Kongruencje mają następujące własności:*

(i) Jeżeli $a \equiv b \ (mod\ n)$ *oraz* $c \equiv d \ (mod\ n)$, *to*

$$a + c \equiv b + d \ (mod\ n),$$

$$a - c \equiv b - d \ (mod\ n),$$

$$ac \equiv bd \ (mod\ n).$$

(ii) Kongruencja $a \equiv b \pmod{n}$ *zachodzi wtedy i tylko wtedy, gdy* $n|(a-b)$.

(iii) Jeżeli $ab \equiv ac \pmod{n}$ *i* $(a,n) = 1$, *to* $b \equiv c \pmod{n}$.

(iv) Jeżeli $a \in \mathbb{N}$ *i* $ab \equiv ac \pmod{an}$, *to* $b \equiv c \pmod{n}$.

(v) Jeżeli $a \equiv b \pmod{n}$, *to* $a^k \equiv b^k \pmod{n}$.

Dowód. (i) Wiemy, że $a - b = hn$ i $c - d = gn$, gdzie $g, h \in \mathbb{Z}$. Z tożsamości

$$(a + c) - (b + d) = (a - b) + (c - d) = (h + g)n,$$

wynika, że

$$(a + c) - (b + d) \equiv 0 \pmod{n} \Rightarrow a + c \equiv b + d \pmod{n}$$

Podobnie dla odejmowania prawdziwa jest tożsamość

$$(a - c) - (b - d) = (a - b) - (c - d) = (h - g)n,$$

skąd wynika, że

$$(a - c) - (b - d) \equiv 0 \pmod{n} \Rightarrow a - c \equiv b - d \pmod{n}.$$

Opierając się na tożsamości

$$ac - bd = (a - b)c + (c - d)b = hnc + gnb = (hc + gb)n,$$

otrzymujemy, że

$$ac - bd = 0 \pmod{n} \Rightarrow ac = bd \pmod{n}.$$

(ii) Jeżeli $a \equiv b \pmod{n}$ to $a - b \equiv 0 \pmod{n}$, więc $a - b = hn$, gdzie $h \in \mathbb{Z}$. Zachodzi $n|hn$, skąd wynika, że $n|a - b$. Dowód implikacji w drugą stronę przebiega analogicznie.

(iii) Jeżeli $ab \equiv ac \pmod{n}$, to znaczy, że $n|(b-c)a$. Założenie $(a, n) = 1$ oznacza, że $n \nmid a$, więc $n|(b - c)$. Stąd wynika, że $b \equiv c \pmod{n}$.

(iv) Jeżeli $ab \equiv ac \pmod{an}$, to $an|a(b - c)$, z czego wynika, że $n|(b - c)$, więc $b \equiv c \pmod{n}$.

(v) Dowód indukcyjny. Dla $k = 1$ kongruencja zachodzi. Załóżmy, że teza jest spełniona dla pewnego k. Mamy więc $a \equiv b \pmod{n}$ oraz $a^k \equiv b^k \pmod{n}$. Wykorzystując własność (i) mnożymy przez siebie obie kongruencje, otrzymując

$$aa^k \equiv bb^k \pmod{n} \Rightarrow a^{k+1} \equiv b^{k+1} \pmod{n},$$

co kończy dowód.

■

Definicja 1.8. [2, s. 30] *Klasami reszt modulo m* nazywamy warstwy (klasy abstrakcji) kongruencji w zbiorze liczb całkowitych $\mathbb{Z}$.

Każdą liczbę całkowitą a można zapisać w postaci $a = qm + r$, gdzie $0 \leq r < m$, więc każda liczba całkowita przystaje modulo m do jednej z liczb ze zbioru $\{0, 1, ..., m-1\}$. Żadne dwie liczby spośród liczb 0,1, ... , $m-1$ nie przystają do siebie modulo m, więc zbiór postaci $\{0, 1, ..., m-1\}$ tworzy pełny układ reprezentantów warstw.

Definicja 1.9. [2, s. 31] *Pełnym układem reszt modulo m* nazywamy pełny układ reprezentantów klas relacji przystawania modulo m.

Definicja 1.10. [2, s. 31] *Zredukowanym układem reszt modulo m* nazywamy układ reprezentantów tych klas relacji przystawania modulo m, które składają się z liczb względnie pierwszych z modułem m.

Definicja 1.11. [5, s. 31] *Funkcją φ (Eulera)* lub tocjentem nazywamy funkcję arytmetyczną, która każdej liczbie naturalnej m przypisuje liczbę elementów zredukowanego układu reszt modulo m.

Można zauważyć, że dla każdej liczby naturalnej m zbiór

$$\phi(m) := \{k : 1 \leq k \leq m, (k, m) = 1\}$$

tworzy zredukowany układ reszt modulo m.

Stwierdzenie 1.1. Funkcji Eulera ma następujące własności:

(1) Dla każdej liczby naturalnej $n > 1$ zachodzi $\varphi(n) \leq n - 1$.
(2) Jeśli liczby naturalne n i m są względnie pierwsze to $\varphi(nm) = \varphi(n)\varphi(m)$.
(3) Jeśli p jest liczbą pierwszą to $\varphi(p) = p - 1$.
(4) Jeśli p jest liczbą pierwszą to $\varphi(p^k) = p^{k-1}(p - 1)$.
(5) Jeśli p i q są różnymi liczbami pierwszymi to $\varphi(pq) = (p - 1)(q - 1)$.
(6) Jeśli $p_1, p_2, ..., p_k$ są wszystkimi czynnikami pierwszymi liczby n bez powtórzeń, to

$$\varphi(n) = n\left(1 - \frac{1}{p_1}\right)\left(1 - \frac{1}{p_2}\right)...\left(1 - \frac{1}{p_k}\right).$$

Przykład 1.3. Obliczymy $\varphi(9)$, $\varphi(12)$, $\varphi(13)$.

Pełnym układem reszt modulo 9 jest zbiór $R = \{0, 1, 2, 3, 4, 5, 6, 7, 8\}$. Jeśli usuniemy z niego wszystkie liczby, które nie są względnie pierwsze z 9, otrzymamy zbiór $R = \{1, 2, 4, 5, 7, 8\}$. Liczba elementów zbioru R jest równa 6, więc $\varphi(9) = 6$.

Wartość funkcji Eulera dla liczby 9 można obliczyć również korzystając z własności (4) stwierdzenia 1.1. Wtedy $\varphi(9) = \varphi(3^2) = 3(3-1) = 3 \cdot 2 = 6$.

Pełnym układem reszt modulo 12 jest zbiór $R = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11\}$. Usuwamy z niego wszystkie liczby, które nie są względnie pierwsze z 12 i otrzymujemy zbiór $R = \{1, 5, 7, 11\}$. Liczba elementów zbioru R wynosi 4, więc $\varphi(12) = 4$.

Wartość funkcji φ dla liczby 13 można obliczyć z własności (3) stwierdzenia 1.1:

$$\varphi(13) = 13 - 1 = 12.$$

Funkcja Eulera jest bardzo prosta do zaimplementowania w środowisku Matlab (skrypt 3.3). Poniżej przedstawiono tabelę z wynikami i czasem działania powyższej funkcji dla liczb o różnej wielkości.

Lp.	Liczba n	Wartość funkcji $\varphi(n)$	Czas działania funkcji $[s]$
1	23	22	0.023398
2	100	40	0.029472
3	567	324	0.047793
4	1000	400	0.051363
5	12345	6576	0.496064
6	123456	41088	5.326313
7	1234567	1224720	65.953887

Tabela 1.1. Wartość funkcji $\varphi(n)$ dla liczb o różnej wielkości.

Stwierdzenie 1.2. Jeśli zbiór $\{a_1, a_2, ..., a_{\varphi(m)}\}$ tworzy zredukowany układ reszt modulo m i $(k, m) = 1$, to zbiór $\{ka_1, ka_2, ..., ka_{\varphi(m)}\}$ także tworzy zredukowany układ reszt modulo m.

Definicja 1.12. [5, s. 15] *Odwrotnością elementu a modulo n* jest taki element, oznaczany przez a^{-1}, że

$$aa^{-1} \equiv 1 \ (mod\ n).$$

Twierdzenie 1.5. *Liczba naturalna a jest odwracalna modulo n wtedy i tylko wtedy, gdy a i n są względnie pierwsze.*

Dowód. Niech $(a, n) = 1$. Wtedy na mocy lematu Bézouta 1.4 dla pewnych k i l zachodzi $ka + ln = 1$. Korzystając z własności (i) twierdzenia 1.4 otrzymujemy

$$ka + ln = 1 \ (mod\ n),$$

skąd

$$ka = 1 \ (mod \ n),$$

więc k jest odwrotnością elementu a modulo n.

Implikacja w drugą stronę: Odwracalność elementu a modulo n oznacza, że istnieje k, dla którego prawdziwa jest kongruencja $ka \equiv 1 \ (mod \ n)$. Wówczas $ka - 1 \equiv 0 \ (mod \ n)$, skąd wynika, że

$$ka - 1 = ln \Rightarrow ka + (-l)n = 1.$$

Z powyższej tożsamości wynika, że a i n są liczbami względnie pierwszymi.

■

Przykład 1.4. Znajdziemy odwrotność liczby 12 modulo 5. Zachodzi $(12, 5) = 1$, więc odwrotność liczby 12 istnieje. Zgodnie z definicją odwrotności modulo sprawdzamy kolejne iloczyny

$$12 \cdot 1 = 12 \equiv 2 \ (mod \ 5),$$

$$12 \cdot 2 = 24 \equiv 4 \ (mod \ 5),$$

$$12 \cdot 3 = 36 \equiv 1 \ (mod \ 5).$$

Stąd wynika, że odwrotnością liczby 12 modulo 5 jest liczba 3.

1.4. Sito Eratostenesa, Atkina i Sundarama

Problem z liczbami pierwszymi polega na ich nieregularnym rozmieszczeniu pośród liczb naturalnych. Nie odkryto dotąd żadnego wzoru pozwalającego na wyszukiwanie kolejnych liczb pierwszych, ale powstało kilka metod znajdowania takich liczb w zadanym przedziale. Metody te są nazywane sitami, ponieważ aby znaleźć wszystkie liczby pierwsze mniejsze od liczby n wystarczy ze zbioru liczb naturalnych mniejszych lub równych n *odsiać* liczby złożone i jedynkę. Pierwszy służący do tego algorytm został przedstawiony przez Eratostenesa z Cyreny (III-II w. p.n.e.). Sito to zostało później ulepszone przez Arthura Atkina i D.J. Bernsteina. Innym, mniej znanym sitem, jest sito Sundarama.

Sito Eratostenesa [1, s. 55] jest algorytmem wyszukiwania kolejnych liczb pierwszych z przedziału $\{2, n\}$.

Algorytm polega na usuwaniu liczb złożonych ze zbioru liczb naturalnych większych od 1 i mniejszych bądź równych n. Na początek z zadanego zbioru wybieramy liczbę najmniejszą i usuwamy wszystkie jej wielokrotności z wyjątkiem jej samej.

Następnie wybieramy najmniejszą liczbę z pozostałych i znów usuwamy wielokrotności tej liczby większe od niej. Postępujemy tak dopóki liczba, której wielokrotności mamy usuwać, jest mniejsza niż $\sqrt{n}$. Na koniec procesu w zbiorze pozostaną tylko liczby pierwsze.

Przykład 1.5. Za pomocą sita Eratostenesa wyznaczymy wszystkie liczby pierwsze należące do zbioru $P = \{2, 3, ..., 40\}$.

I. Najmniejszą liczbą w zbiorze P jest 2, więc usuwamy wszystkie większe od 2 wielokrotności liczby 2.

$$P = \{2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39\}.$$

II. Kolejną najmniejszą liczbą w zbiorze (nie biorąc pod uwagę liczby 2) jest liczba 3, więc usuwamy wszystkie jej wielokrotności, przy czym nie usuwamy samej trójki.

$$P = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35, 37\}.$$

III. Następną najmniejszą liczbą w zbiorze (nie biorąc pod uwagę liczb 2 i 3) jest liczba 5, więc wyrzucamy ze zbioru P wszystkie jej wielokrotności, zostawiając samą 5.

$$P = \{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37\}.$$

Krok trzeci jest ostatni, ponieważ w czwartym kroku najmniejszą liczbą, której wielokrotności musielibyśmy usunąć, jest liczba 7, ale $n = 40$, a $\sqrt{40} \approx 6,32$. Liczba 7 jest więc większa niż 6.32, możemy zatem zatrzymać algorytm. Po trzecim kroku w zbiorze P pozostały jedynie liczby pierwsze.

Sito Eratostenesa można teoretycznie zapisać w postaci funkcji, na przykład w programie Matlab (skrypt 3.1). Poniżej przedstawiona została tabela z wynikami działania algorytmu dla przedziałów o różnej długości.

Lp.	Koniec przedziału $\{2, n\}$	Czas działania programu $[s]$	Liczba liczb pierwszych
1	100	0.002345	25
2	1000	0.00219	168
3	10 000	0.107349	1229
4	100 000	0.191775	9592
5	1 000 000	0.253845	78 498
6	10 000 000	6.684239	664 579
7	100 000 000	174.164354	5 761 455

Tabela 1.2. Wyniki działania funkcji szukającej liczb pierwszych sitem Eratostenesa.

Niestety w przypadku większego zakresu liczb czas oczekiwania na wynik jest bardzo długi. Na przykład przeszukiwanie zbioru liczb naturalnych $\{2, 1\,000\,000\,000\}$ może trwać już kilka godzin. Co więcej, największa znana obecnie liczba pierwsza jest o wiele większa od liczb, które można znaleźć za pomocą tego algorytmu nawet na nowoczesnych komputerach o dużej wydajności.

Sito Atkina lub *sito Atkina-Bernsteina* [1, s. 55] to algorytm autorstwa dwóch matematyków - Arthura Atkina i D.J. Bernsteina, który służy do wyszukiwania liczb pierwszych w dużych przedziałach.

Metoda ta działa podobnie do sita Eratostenesa, jednak dzięki wykorzystaniu pewnych zależności jest efektywniejsza i wymaga znacznie mniej pamięci. Na przykład algorytm całkowicie pomija wszystkie liczby podzielne przez 2, 3 lub 5, liczby z parzystą resztą z dzielenia przez 60, liczby, które mają resztę z dzielenia przez 60 podzielną przez 3 oraz liczby z resztą z dzielenia przez 60 podzielną przez 5. Pierwszość pozostałych liczb rozpatrywana jest na podstawie ich reszty z dzielenia przez liczbę 12. Jeżeli wynosi ona:

- 1 lub 5 — liczba jest pierwsza, jeśli liczba rozwiązań równania $4x^2 + y^2 = n$ (gdzie n jest analizowaną przez nas liczbą, a x i y to dowolne liczby naturalne) jest nieparzysta, a n nie jest kwadratem innej liczby naturalnej,
- 7 — liczba jest pierwsza, jeśli liczba rozwiązań równania $3x^2 + y^2 = n$ (gdzie n to analizowana przez nas liczba, a x i y to dowolne liczby naturalne) jest nieparzysta, a n nie jest kwadratem innej liczby naturalnej,
- 11 — liczba jest pierwsza, jeśli liczba rozwiązań równania $3x^2 \breve{} y^2 = n$ (gdzie n to analizowana przez nas liczba, a x i y to dowolne liczby naturalne i $x > y$) jest nieparzysta, a n nie jest kwadratem innej liczby naturalnej.

Sito Sundarama [1, s. 55] to prosty algorytm autorstwa XX-wiecznego indyjskiego matematyka Sundarama, który służy do wyszukiwania liczb pierwszych w przedziale $\{1, n\}$.

Algorytm przebiega w następujący sposób:

1. Wybieramy przedział $\{1, n\}$.
2. Z zadanego przedziału usuwamy liczby obliczone wzorem $l = i + j + 2ij$, gdzie $i, j \in \mathbb{N}$, $1 \leq i \leq j$ oraz $i + j + 2ij \leq n$.
3. Pozostałe liczby ze zbioru mnożymy przez 2 i dodajemy 1.

Po przeprowadzeniu algorytmu uzyskujemy listę liczb pierwszych z przedziału $\{3, 2n+1\}$.

Sito Sundarama można zapisać w postaci funkcji, na przykład w programie Matlab (skrypt 3.2). Poniższa tabela zawiera wyniki działania sita Sundarama dla przedziałów o różnej długości.

Lp.	Koniec przedziału $\{1, n\}$	Czas działania programu $[s]$	Liczba liczb pierwszych w przedziale $\{2, 2n+1\}$
1	50	0.000764	26
2	500	0.000663	168
3	5000	0.001613	1229
4	50 000	0.006975	9592
5	500 000	0.055643	78 498
6	5 000 000	0.639726	664 579
7	50 000 000	7.374154	5 761 455
8	500 000 000	137.090207	50 847 534

Tabela 1.3. Wyniki działania funkcji szukającej liczb pierwszych sitem Sundarama.

Jak łatwo zauważyć, algorytm Sundarama jest o wiele szybszy niż sito Eratostenesa.

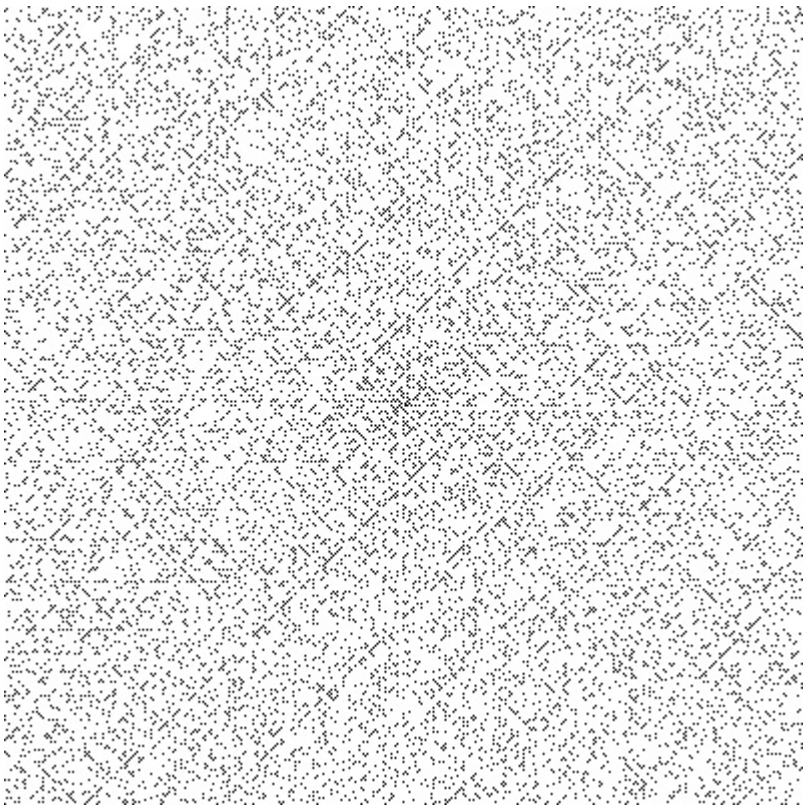
1.5. Spirala Ulama

Stanisław Marcin Ulam (ur. 13 kwietnia 1909 we Lwowie, zm. 13 maja 1984 w Santa Fe) to polski matematyk i przedstawiciel lwowskiej szkoły matematycznej. W swoim dorobku ma wiele dokonań w zakresie matematyki i fizyki matematycznej

oraz metod numerycznych. W 1963 roku zaprezentował przedstawienie rozkładu liczb pierwszych w formie graficznej, zwanej dziś spiralą Ulama.

Spirala Ulama lub *spirala liczb pierwszych* [8] to graficzna metoda przedstawiania rozkładu liczb pierwszych.

Sposób zapisu polega na tym, że na kwadratowej tablicy zaczynając od 1 w środku spiralnie wypisuje się kolejne liczby naturalne. Okazuje się, że na pewnych przekątnych liczby pierwsze pojawiają się o wiele częściej niż na innych. Co więcej, zjawisko nie zależy od wyboru pierwszej liczby w spirali. Na Rysunku 1.5 przedstawiono spiralę Ulama o rozmiarze 400x400.



Rysunek 1.1. Spirala Ulama o rozmiarze 400x400 [8].

1.6. Twierdzenie Wilsona i małe twierdzenie Fermata

Podrozdział ten jest poświęcony dwóm klasycznym twierdzeniom z zakresu teorii liczb - twierdzeniu Wilsona i małemu twierdzeniu Fermata. Mają one szerokie zastosowanie w testach pierwszości liczb.

Twierdzenie 1.6. *(Twierdzenie Wilsona, 1770 r.) [5, s. 18] Liczba naturalna $p > 1$ jest liczbą pierwszą wtedy i tylko wtedy, gdy*

$$(p-1)! + 1 \equiv 0 \ (mod\ p).$$

Za odkrywcę powyższego twierdzenia uznaje się Johna Wilsona, zostało jednak opublikowane przez Edwarda Waringa. Jak się okazuje, wcześniej mówił o nim już al-Hajsam (X/XI w.) i Leibniz (XVII w.). Pierwszy dowód twierdzenia Wilsona pochodzi od Lagrange'a i został przedstawiony w 1777 roku.

Dowód. Na początek należy pokazać, że jeśli p jest liczbą pierwszą, to zachodzi powyższa kongruencja. Dla $p = 2$ implikacja jest spełniona, więc ograniczono się do liczb pierwszych nieparzystych. Ponieważ p jest liczbą pierwszą, więc każda z liczb $1, 2, 3, ..., p-2, p-1$ ma swoją odwrotność modulo p. Poszukuje się, dla jakich a spośród tych liczb zachodzi

$$a \equiv a^{-1} \ (mod\ p),$$

tzn.

$$a^2 \equiv 1 \ (mod\ p).$$

Ostatni warunek oznacza, że liczba p musi dzielić

$$a^2 - 1 = (a+1)(a-1).$$

Z Lematu 1.1 (Euklidesa) wynika, że p dzieli $a+1$ lub $a-1$, więc $a = p-1$ lub $a = 1$. Te dwie liczby na razie pominięto. Wówczas liczby $2, 3, ..., p-2$ dzielą się na pary, liczba i jej odwrotność modulo p, skąd wynika, że

$$2 \cdot 3 \cdot ... \cdot (p-2) \equiv 1 \ (mod\ p).$$

Mnożąc obie strony powyższej kongruencji przez $p-1$ otrzymano

$$(p-1)! \equiv p-1 \equiv -1 \ (mod\ p),$$

co kończy dowód pierwszej implikacji.

Teraz wystarczy pokazać implikację w drugą stronę. Należy zatem udowodnić, że jeżeli n jest liczbą złożoną to zachodzi

$$(n-1)! + 1 \not\equiv 0 \ (mod\ n).$$

Jeśli n jest liczbą złożoną, to zachodzi $n = pq$. Założono, że $p < q$. Wówczas

$$(n-1)! = 1 \cdot 2 \cdot 3 \cdot ... \cdot (p-1) \cdot p \cdot ... \cdot (q-1) \cdot q \cdot ... \cdot (n-1),$$

więc $(n-1)!$ jest podzielne przez n. Stąd $(n-1)!+1$ nie może być podzielne przez n. Wynika stąd, że

$$(n-1)!+1 \not\equiv 0 \ (mod\ n),$$

co kończy dowód.

■

Twierdzenie 1.7. *(Eulera) [2, s. 33] Jeżeli n i a są względnie pierwszymi liczbami naturalnymi, to zachodzi*

$$a^{\varphi(n)} \equiv 1 \ (mod\ n).$$

Dowód. Niech układ

$$r_1, r_2, ..., r_{\varphi(n)}$$

będzie układem reszt modulo n. Po pomnożeniu każdej z nich przez a i podzieleniu przez m powstaje nowy układ reszt $r_i^{'}$:

$$r_i a = q_i n + r_i^{'},\ i = 1, 2, ..., \varphi(n).$$

Układ można również przedstawić w postaci kongruencji

$$r_i a = r_i^{'} \ (mod\ n),\ i = 1, 2, ..., \varphi(n). \tag{1.1}$$

Po wymnożeniu przez siebie wszystkich $\varphi(n)$ kongruencji 1.1 otrzymano

$$a^{\varphi(n)} r_1 r_2 ... r_{\varphi(n)} \equiv r_1^{'} r_2^{'} ... r_{\varphi(n)}^{'} \ (mod\ n)$$

Zarówno r_i i $r_i^{'}$ dla $i = 1, 2, ..., \varphi(n)$ są względnie pierwsze z n, więc obie strony można podzielić przez ich identyczny iloczyn, co daje

$$a^{\varphi(n)} \equiv 1 \ (mod\ n).$$

■

Twierdzenie 1.8. *(Małe twierdzenie Fermata, 1640 r.) [5, s. 19] Jeśli p jest liczą pierwszą, to dla każdej liczby całkowitej a niepodzielnej przez p zachodzi*

$$a^{p-1} \equiv 1 \ (mod\ p).$$

Twierdzenie to zostało odkryte jednocześnie przez Fermata i Leibniza. Żaden z nich nie opublikował jednak dowodu.

Dowód. Jeśli liczba pierwsza p nie dzieli a, to ciąg reszt z dzielenia a przez p $1, 2, 3, ..., p-1$ oraz ciąg liczb

$$a, 2a, 3a, ..., (p-1)a \ (mod\ p) \tag{1.2}$$

różnią się jedynie kolejnością. Łatwo zauważyć, że jeśli p nie dzieli a, to niemożliwe jest, aby $ka \equiv 0 \ (mod\ p)$ dla $k = 1, 2, ..., p-1$. Z tego wynika, że wszystkie reszty $ka \ (mod\ p)$ są liczbami naturalnymi od 1 do $p-1$. Ponieważ liczb tych jest dokładnie $p-1$, więc należy pokazać, że są one parami różne.

Założono, że dla $i, j < p$, gdzie $i \neq j$ zachodzi $ai \equiv aj \ (mod\ p)$. Z treści twierdzenia wiadomo, że a nie jest podzielne przez p, więc obie strony można podzielić przez a, skąd otrzymano $i \equiv j \ (mod\ p)$. Obie liczby są z założenia mniejsze od p, więc $i = j$, co prowadzi do sprzeczności.

Jak wcześniej zauważono, ciąg liczb 1.2 modulo p to ciąg $1, 2, ..., p-1$, więc

$$a \cdot 2a \cdot 3a \cdot ... \cdot (p-1)a \equiv 1 \cdot 2 \cdot 3 \cdot ... \cdot (p-1) \ (mod\ p),$$

a po przekształceniu

$$(p-1)!a^{p-1} \equiv (p-1)! \ (mod\ p).$$

Z Twierdzenia 1.6 wiadomo, że liczba $(p-1)!$ jest względnie pierwsza z p, więc obie strony kongruencji można podzielić przez $(p-1)!$, co kończy dowód.

■

Definicja 1.13. Jeżeli n jest nieparzystą liczbą złożoną, a liczba b, względnie pierwsza z n, spełnia warunek

$$b^{n-1} \equiv 1 \ (mod\ n),$$

to n jest *liczbą pseudopierwszą* przy podstawie b.

Z twierdzenia Fermata 1.8 wynika, że dla każdej liczby a, która jest względnie pierwsza z modułem n, istnieje liczba naturalna k taka, że $a^k \equiv 1 \ (mod\ n)$.

Definicja 1.14. [10] *Rzędem multiplikatywnym* liczby całkowitej a modulo n, gdzie $(a, n) = 1$, nazywamy najmniejszą liczbę naturalną k, dla której zachodzi

$$a^k \equiv 1 \ (mod\ n).$$

Rząd elementu a modulo n oznacza się $ord_n(a)$ lub $o_n(a)$.

Przykład 1.6. Obliczymy rząd multiplikatywny liczby 4 modulo 7.

Szukamy najmniejszej liczby $k \in \mathbb{N}$, dla której prawdziwa jest kongruencja

$$4^k \equiv 1 \ (mod\ 7).$$

Sprawdzamy kongruencję dla kolejnych wartości liczby k:

$$k = 1 \Rightarrow 4^1 = 4 \equiv 4 \ (mod\ 7) \neq 1 \ (mod\ 7),$$
$$k = 2 \Rightarrow 4^2 = 16 = 2 \cdot 7 + 2 \equiv 2 \ (mod\ 7) \neq 1 \ (mod\ 7),$$
$$k = 3 \Rightarrow 4^3 = 64 = 9 \cdot 7 + 1 \equiv 1 \ (mod\ 7).$$

Rzędem multiplikatywnym liczby 4 modulo 7 jest 3, czyli $o_7(4) = 3$.

Obliczanie rzędu elementu a modulo n jest również proste do zaimplementowania w środowisku Matlab (skrypt 3.4). Poniżej przedstawiono tabelę z wynikami i czasem działania powyższej funkcji.

Lp.	Liczba n	Liczba a	Wartość funkcji $o_n(a)$	Czas działania funkcji $[s]$
1	7	4	3	0.033575
2	4	7	2	0.025657
3	100	387	20	0.031364
4	167	379	166	0.036658
5	1000	37	100	0.035297

Tabela 1.4. Wartość funkcji $o_n(a)$ dla liczb o różnej wielkości.

1.7. Własności liczb pierwszych

Liczby pierwsze mogą przyjmować różne postaci. Podrozdział ten skupia się na różnych rodzajach liczb pierwszych, takich jak liczby pierwsze Fermata, bliźniacze, czworacze, Sophie Germain czy Mersenne'a. W części tej przedstawione zostanie również zestawienie dziesięciu największych znanych liczb pierwszych na rok 2018.

Twierdzenie 1.9. *[2, s. 46] Istnieje nieskończenie wiele liczb pierwszych postaci* $4k - 1$.

Dowód. Przypuśćmy przeciwnie, że $\{q_1, q_2, ..., q_r\}$ jest zbiorem wszystkich liczb pierwszych postaci $4k - 1$. Zdefiniujmy $N = 4q_1q_2...q_r - 1$.

Ponieważ N jest liczbą nieparzystą, więc wszystkie dzielniki liczby N również są nieparzyste. Z kolei każda liczba nieparzysta ma postać $4k - 1$ lub $4k + 1$. Wiadomo

także, że iloczyn liczb postaci $4k+1$ jest też liczbą postaci $4k+1$, więc N musi mieć dzielnik pierwszy q postaci $4k-1$. Jak wiadomo, $q_1 \nmid N$, $q_2 \nmid N$, ..., $q_r \nmid N$, skąd wynika, że $q \notin \{q_1, q_2, ..., q_r\}$, co jest sprzeczne z założeniem.

■

Twierdzenie 1.10. *[2, s. 46] Istnieje nieskończenie wiele liczb pierwszych postaci* $4k+1$.

Twierdzenie 1.11. *(Dirichlet) [2, s. 47] Jeśli* $m \in \mathbb{N}$, $a \in \mathbb{Z}$ *oraz* $(a, m) = 1$, *to w ciągu arytmetycznym* $a + mk$, *gdzie* $k = 1, 2, ...$, *istnieje nieskończenie wiele liczb pierwszych.*

Definicja 1.15. Niech $n \in \mathbb{N}$. Liczbę w postaci

$$F_n = 2^{2^n} + 1, \tag{1.3}$$

gdzie $n \geq 1$, nazywa się *n-tą liczbą Fermata* .

Fermat obliczył, że $F_1 = 5$, $F_2 = 17$, $F_3 = 2575$, $F_4 = 65537$ są liczbami pierwszymi i założył, że każda liczba postaci 1.3 jest liczbą pierwszą. Hipotezę tą obalił Euler, który udowodnił, że F_5 jest podzielne przez 641.

Definicja 1.16. [6] Liczby pierwsze p i $q = p + 2$ nazywane są *liczbami pierwszymi bliźniaczymi* .

Liczbami pierwszymi bliźniaczymi są np.: (3, 5), (11, 13), (1949, 1951), (4967, 4969). Największe znane dziś liczby bliźniacze, każda składająca się z 388 342 cyfr, to

$$2\,996\,863\,034\,895 \cdot 2^{1\,290\,000} \pm 1,$$

znalezione w 2016 roku.

Definicja 1.17. [6] *Liczba pierwsza Sophie Germain* to dowolna liczba pierwsza p, dla której liczba $2p+1$ także jest liczbą pierwszą (np. 23, gdyż $2 \cdot 23 + 1 = 47$).

Największą znaną obecnie liczbą pierwszą Sophie Germain jest

$$2\,618\,163\,402\,417 \cdot 2^{1\,290\,000} - 1,$$

a jej zapis dziesiętny wymaga 388 342 cyfr. Została ona odkryta w 2016 roku.

Definicja 1.18. [2, s. 52] *Liczbami Mersenne'a* są liczby postaci $M_n = 2^n - 1$, $n \geq 1$. Jeżeli liczba Mersenne'a jest pierwsza to nazywa się *liczbą pierwszą Mersenne'a* .

Nie wiadomo, czy istnieje nieskończenie wiele liczb pierwszych Mersenne'a. Do 2018 roku znanych ich 50. Co więcej, dziewięć na dziesięć największych odkrytych do tej pory liczb pierwszych to właśnie liczby Mersenne'a. Zostały one przedstawione w tabeli 1.5. Ich poszukiwaniem zajmuje się projekt GIMPS (ang. Great Internet Mersenne Prime Search), w którym mogą wziąć udział ochotnicy z całego świata.

Nr	Liczba pierwsza	Liczba cyfr	Data odkrycia	Odkrywcy
1	$2^{77\,232\,917} - 1$	23 249 425	2018-01-03	J. Pace, GIMPS
2	$2^{74\,207\,281} - 1$	22 338 618	2016-01-07	Cooper, GIMPS
3	$2^{57\,885\,161} - 1$	17 425 170	2013-01-25	Cooper, GIMPS
4	$2^{43\,112\,609} - 1$	12 978 189	2008-08-23	Edson Smith, GIMPS
5	$2^{42\,643\,801} - 1$	12 837 064	2009-06-04	Strindmo, GIMPS
6	$2^{37\,156\,667} - 1$	11 185 272	2008-09-06	H. Elvenich, GIMPS
7	$2^{32\,582\,657} - 1$	9 808 358	2006-09-04	Boone, Cooper, GIMPS
8	$2^{30\,402\,457} - 1$	9 152 052	2005-12-15	Boone, Cooper, GIMPS
9	$2^{25\,964\,951} - 1$	7 816 230	2005-02-18	Nowak, GIMPS
10	$2^{24\,036\,583} - 1$	7 235 733	2004-05-15	Findley, GIMPS

Tabela 1.5. Dziesięć największych znanych liczb pierwszych Mersenne'a na 2018 rok [7].

2. Testy pierwszości liczb

Test pierwszości to algorytm, który określa, czy zadana liczba jest liczbą pierwszą, czy złożoną. Poniższy rozdział ma na celu przedstawienie algorytmów, które testują pierwszość zadanej liczby.

2.1. Testy deterministyczne

Testy deterministyczne to testy, które wydają certyfikat pierwszości, czyli ich wynik jest prawidłowy z prawdopodobieństwem 1. Zwracają odpowiedź, że podana na wejściu liczba n jest pierwsza wtedy i tylko wtedy, gdy naprawdę tak jest, czyli wynik działania testu jest jednoznaczny.

2.1.1. Metoda naiwna

Najprostszą deterministyczną metodą testowania pierwszości jest *metoda naiwna* . Algorytm działania jest bardzo prosty: dla danej liczby $n > 2$ wystarczy sprawdzić, czy dzieli się ona kolejno przez $2, 3, ..., n-1$. Jeżeli wśród tych liczb nie ma dzielnika liczby n, wtedy n jest pierwsza. Jeśli chociaż przez jedną z tych liczb n dzieli się bez reszty, wtedy n jest złożona.

Jak się okazuje, nie trzeba testować podzielności przez wszystkie liczby od 2 do $n-1$, wystarczy ograniczyć się do przedziału $2, 3, ..., \sqrt{n}$.

Co więcej, algorytm można ulepszyć dzieląc liczbę n jedynie przez wszystkie liczby pierwsze mniejsze lub równe $\sqrt{n}$. Listę takich liczb możemy wyznaczyć sitem Eratostenesa, które omówione zostało w podrozdziale 1.4.

Warto zauważyć, że jeśli liczba n okaże się być złożona, to jednocześnie wyznaczany jest jej nietrywialny dzielnik (dzielnik różny od 1).

Algorytm metody naiwnej w wersji udoskonalonej działa następująco:

1. Pobieramy liczbę n.
2. Wyznaczamy zbiór p wszystkich liczb pierwszych mniejszych lub równych $\sqrt{n}$.
3. Dzielimy liczbę n przez kolejne liczby ze zbioru p.
4. Jeśli n dzieli się przez przynajmniej jedną z liczb ze zbioru p bez reszty, wtedy n jest złożona.
5. Jeśli wśród liczb p nie ma dzielnika liczby n, wtedy n jest pierwsza.

Jest to algorytm deterministyczny, który uznaje pierwszość liczby ze stuprocentową pewnością. W związku z tym, że metoda jest bardzo dokładna, wymaga dużej liczby dzieleń, więc jej efektywność spada dla większych liczb.

Przykład 2.1. Za pomocą algorytmu metody naiwnej sprawdzimy, czy liczba 53 jest pierwsza.

Zgodnie z algorytmem działania metody naiwnej, wyznaczamy zbiór p wszystkich liczb pierwszych mniejszych lub równych $\sqrt{53} \approx 7,28$. Zbiór ten ma więc postać

$$p = \{2, 3, 5, 7\}.$$

W następnym kroku sprawdzamy reszty z dzielenia liczby 53 przez liczby pierwsze ze zbioru p:

$$53 = 1 \ (mod\ 2),$$
$$53 = 2 \ (mod\ 3),$$
$$53 = 3 \ (mod\ 5),$$
$$53 = 4 \ (mod\ 7).$$

Liczba 53 nie dzieli się bez reszty przez żadną z liczb ze zbioru p, więc 53 jest liczbą pierwszą.

Przykład 2.2. Za pomocą algorytmu metody naiwnej sprawdzimy, czy liczba 25 jest pierwsza.

W pierwszym kroku wyznaczamy zbiór p wszystkich liczb pierwszych mniejszych lub równych $\sqrt{25} = 5$. Zbiór ten jest postaci

$$p = \{2, 3, 5\}.$$

W następnym kroku sprawdzamy reszty z dzielenia liczby 25 przez liczby pierwsze ze zbioru p:

$$25 = 1 \ (mod\ 2),$$
$$25 = 1 \ (mod\ 3),$$
$$25 = 0 \ (mod\ 5).$$

Liczba 25 dzieli się bez reszty przez liczbę 5, więc jest liczbą złożoną.

Algorytm metody naiwnej w programie Matlab został przedstawiony na końcu tej pracy (skrypt 3.5. Tabela 2.1 przedstawia wyniki i czas działania algorytmu dla liczb o różnej wielkości.

Lp.	Testowana liczba	Czas działania programu $[s]$	Liczba pierwsza/złożona
1	7	0.005050	pierwsza
2	23	0.004529	pierwsza
3	365	0.002450	złożona
4	5737	0.002264	pierwsza
5	99 487	0.015185	pierwsza
6	99 489	0.014734	złożona
7	100 001	0.013290	złożona
8	12 192 109	9.409341	pierwsza
9	14 476 001	11.304216	pierwsza
10	100 000 001	215.084406	złożona

Tabela 2.1. Wyniki działania funkcji testującej pierwszość liczb metodą naiwną z sitem Eratostenesa.

Łatwo zauważyć, że dla niewielkich liczb czas działania jest bardzo krótki, ale dla liczb rzędu 10^8 sprawdzanie pierwszości trwa już kilka minut.

2.1.2. Test pierwszości AKS

Test pierwszości AKS (lub inaczej test pierwszości Agrawal-Kayal-Saxena lub cyklotomiczny test AKS) jest deterministycznym testem pierwszości liczb opracowanym przez trójkę indyjskich naukowców - Manindra Agrawala, Neeraja Kayala i Nitina Saxena z Indyjskiego Instytutu Technologii w Kanpurze. Test został po raz pierwszy opublikowany 6 sierpnia 2002 roku w artykule zatytułowanym *PRIMES is in P*. W 2006 roku autorzy otrzymali za to odkrycie dwie nagrody - Nagrodę Gödla, którą przyznaje się za osiągnięcia w dziedzinie informatyki teoretycznej, oraz nagrodę Fulkersona, przyznawaną za wybitne prace z zakresu matematyki dyskretnej.

Poniższe opracowanie opiera się na artykule *PRIMES is in P* [9].

Test AKS jest pierwszym opublikowanym algorytmem testującym czy dana liczba jest pierwsza, który jest jednocześnie szybki, jednoznaczny i bezwarunkowy. Oznacza to, że algorytm ten zawsze zwróci poprawną odpowiedź (w przeciwieństwie

do testów probabilistycznych), a jego działanie nie bazuje na żadnych nieudowodnionych hipotezach.

Test pierwszości AKS opiera się na pewnej tożsamości liczb pierwszych, która jest uogólnieniem małego twierdzenia Fermata 1.8.

Lemat 2.1. (małe twierdzenie Fermata dla wielomianów) Niech $a, n \in \mathbb{N}$ i $n \geq 2$ oraz $(a, n) = 1$. Wówczas n jest liczbą pierwszą wtedy i tylko wtedy, gdy

$$(X + a)^n = X^n + a \ (mod\ n). \tag{2.1}$$

Dowód. Równanie 2.1 można zapisać w postaci

$$(X + a)^n - (X^n + a) = 0 \ (mod\ n).$$

Dla $0 \leq i \leq n$ współczynnik przy X^i w wyrażeniu $((X + a)^n - (X^n + a))$ wynosi $\binom{n}{i}\ a^{n-i}$.

Załóżmy, że n jest liczbą pierwszą. Wtedy $\binom{n}{i} = 0 \ (mod\ n)$, a więc wszystkie współczynniki są zerowe.

Załóżmy, że n jest liczbą złożoną. Rozważmy liczbę pierwszą q, która jest dzielnikiem n i dla pewnego k niech $q^k \mid n$. Wtedy q^k nie dzieli $\binom{n}{q}$ i jest względnie pierwsza z a^{n-q}. Stąd współczynnik przy X^q jest niezerowy (mod n). Zatem wyrażenie $((X + a)^n - (X^n + a))$ nie jest tożsamościowo równe zero (mod n).

■

Powyższa tożsamość sugeruje prosty test pierwszości: pobieramy badane n, wybieramy a i sprawdzamy, czy kongruencja 2.1 jest spełniona. Jednak wymaga to czasu, ponieważ w najgorszym przypadku należy oszacować n współczynników po lewej stronie równania. Prostym sposobem na zmniejszenie liczby tych współczynników jest oszacowanie obu stron kongruencji 2.1 modulo wielomian postaci $X^r - 1$ dla odpowiednio wybranego małego r. Innymi słowy, sprawdzić, czy spełnione jest równanie

$$(X + a)^n = X^n + a \ (mod\ X^r - 1, n). \tag{2.2}$$

Z Lematu 2.1 wynika natychmiast, że wszystkie liczby pierwsze n spełniają równanie 2.2 dla wszystkich wartości a i r. Problem polega na tym, że niektóre liczby złożone n mogą również spełnić równanie dla kilku wartości a i r. Jednak możemy pokazać, że dla odpowiednio dobranego r, jeśli równanie 2.2 jest spełnione dla kilku a, to n musi być potęgą liczby pierwszej.

Algorytm działania metody AKS jest następujący:

1. Wybieramy testowaną liczbę $n > 1$.
2. Jeśli istnieje takie $a \in \mathbb{N}$ i $b > 1$, że $n = a^b$, to liczba n jest złożona - stop.
3. Znajdujemy najmniejsze r takie, że $o_r(n) > (log_2 n)^2$.
4. Jeśli $1 < NWD(a, n) < n$ dla pewnego $a \leq r$, wtedy n jest złożona - stop.
5. Jeśli $n \leq r$, n jest pierwsza - stop.
6. Dla $a = 1$ do $\lfloor \sqrt{\varphi(r)} log_2 n \rfloor$ sprawdzamy
   jeśli $((X + a)^n \neq X^n + a \ (mod\ X^r - 1, n))$, n jest złożona - stop.
7. n jest pierwsza.

Przykład 2.3. Za pomocą algorytmu testu AKS sprawdzimy, czy liczba 11 jest pierwsza.

1. $n = 11 > 1$.
2. n nie jest kwadratem żadnej liczby całkowitej.
3. $(log_2 n)^2 \approx 11,97$, więc $r = 13$, ponieważ $o_{13}(11) = 12$.
4. Dla każdego $a \leq 13$ i $a \neq 11$ $NWD(a, 11) = 1$. Dla $a = 11$ $NWD(a, 11) = 11$, więc $NWD(a, 11) \notin (1, 11)$.
5. $11 \leq 13$, więc 11 jest pierwsza.

Kod metody AKS zaimplementowany w środowisku Matlab został przedstawiony w części pracy poświęconej skryptom (skrypt 3.6). Tabela poniżej przedstawia wyniki i czas działania testu pierwszości AKS.

Lp.	Liczba n	Pierwsza/złożona	Czas działania funkcji $[s]$
1	3	pierwsza	0.034859
2	23	pierwsza	0.047291
3	25	złożona	0.009026
4	123	złożona	0.045841
5	199	pierwsza	4.284376
6	1000	złożona	0.020794
7	1001	złożona	0.047622
8	1223	pierwsza	35.298198
9	3313	pierwsza	113.642901
10	3315	złożona	0.044053

Tabela 2.2. Wartość funkcji $\varphi(n)$ dla liczb o różnej wielkości.

Łatwo zauważyć, że algorytm radzi sobie bardzo szybko z liczbami złożonymi i niewielkimi liczbami pierwszymi, jednak już dla liczb pierwszych czterocyfrowych potrzebne jest ponad pół minuty na potwierdzenie ich pierwszości. Test AKS ma niestety znikomą wartość praktyczną, ponieważ złożoność tego algorytmu jest duża, a to oznacza, że liczba wykonywanych operacji rośnie bardzo szybko wraz ze wzrostem wielkości badanej liczby.

2.1.3. Test pierwszości APR

2.1.4. Test Lucasa-Lehmera

2.2. Testy probabilistyczne

Testy probabilistyczne (znane także jako testy randomizacyjne) są obecnie najczęściej stosowanymi i najbardziej efektywnymi testami pierwszości liczb, które w swoim działaniu używają losowości. Wykorzystuje się w nich losowo wygenerowane liczby z zadanego przedziału. Zależność od takich elementów powoduje, że algorytm może dać błędny wynik testu pierwszości, lecz prawdopodobieństwo takiego zdarzenia jest bardzo małe.

2.2.1. Test pierwszości Fermata

Zgodnie z małym twierdzeniem Fermata 1.8 dla liczb pierwszych p zachodzi kongruencja

$$a^{p-1} \equiv 1 \ (mod\ p). \tag{2.3}$$

Test pierwszości Fermata [5, s. 49] to probabilistyczny test pierwszości, który opiera się na założeniu, że zachodzi wynikanie odwrotne do twierdzenia 1.8, czyli jeśli prawdziwa jest kongruencja 2.3, to liczba p jest pierwsza z dużym prawdopodobieństwem.

Algorytm testu pierwszości Fermata dla zadanej liczby p wygląda następująco:

1. Losujemy $a \in [2, p-2]$.
2. Obliczamy $d = NWD(a, p)$. Jeśli $d > 1$ to p jest liczbą złożoną.
3. Sprawdzamy, czy zachodzi kongruencja $a^{p-1} \equiv 1 \ (mod\ p)$.
4. Jeśli zachodzi, to p może być liczbą pierwszą, w innym przypadku p jest liczbą złożoną.

Przy kilkukrotnym powtórzeniu losowania dla różnych liczb test Fermata daje bardzo wiarygodny wynik.

Przykład 2.4. Używając testu pierwszości Fermata sprawdzimy, czy liczba 23 jest liczbą pierwszą.

1. Losujemy $a \in [2, 21]$. Niech $a = 4$.
2. $d = NWD(4, 23) = 1$.
3. Kongruencja $a^{p-1} \equiv 1 \ (mod\ p)$ jest spełniona, ponieważ $4^{22} \equiv 1 \ (mod\ 23)$.
4. Liczba 23 może być liczbą pierwszą.

Przykład 2.5. Używając testu pierwszości Fermata sprawdzimy, czy liczba 25 jest liczbą pierwszą.

1. Losujemy $a \in [2, 24]$. Niech $a = 8$.
2. $d = NWD(8, 25) = 1$.
3. Kongruencja $a^{p-1} \equiv 1 \ (mod\ p)$ nie zachodzi, ponieważ $8^{24} \equiv 21 \ (mod\ 25)$.
4. Liczba 25 jest liczbą złożoną.

Implementacja testu pierwszości Fermata w programie Matlab zawarta została w skrypcie 3.7. W tabeli 2.3 przedstawione są wyniki działania tego kodu dla różnych liczb pierwszych i złożonych oraz czas działania algorytmu.

Lp.	Liczba p	Liczba powtórzeń n	Wynik	Czas działania funkcji $[s]$
1	5	2	pierwsza (100%)	0.069366
2	15	8	złożona (100%)	0.097156
3	47	18	pierwsza (100%)	0.102425
4	199	28	pierwsza (100%)	0.090658
5	1999	68	pierwsza (100%)	0.092615
6	6899	190	pierwsza (100%)	0.095254
7	94 427	19	pierwsza (100%)	0.080347
8	94 428	19	złożona (100%)	0.070723
9	15 472 147	30	pierwsza (100%)	0.071718
10	15 472 147	5530	pierwsza (100%)	0.733975
11	15 472 148	5530	złożona (100%)	0.575719

Tabela 2.3. Wyniki działania testu pierwszości Fermata dla liczb o różnej wielkości.

Łatwo zauważyć, że algorytm działa niezwykle szybko i zwraca stuprocentowo pewny wynik nawet dla bardzo dużych liczb. Czas działania zależy głównie od wyboru liczby powtórzeń n - im liczba n jest większa, tym czas działania algorytmu się wydłuża. Jednak już dla niewielkich n wynik jest podawany z prawdopodobieństwem równym 1.

Trzeba zaznaczyć, że kongruencja 2.3 może zachodzić także w pewnych przypadkach, gdy liczba p nie jest liczbą pierwszą.

Definicja 2.1. [5, s. 49] *Liczby Carmichaela* to złożone liczby naturalne, dla których teza małego twierdzenia Fermata 1.8 jest prawdziwa. Liczba naturalna n jest liczbą Carmichaela wtedy i tylko wtedy, gdy jest liczbą złożoną i dla każdej liczby naturalnej a z przedziału $1 < a < n$, względnie pierwszej z n, liczba $a^{n-1} - 1$ jest podzielna przez n.

Najmniejszą liczbą Carmichaela jest $561 = 3 \cdot 11 \cdot 17$. Innymi z nich są na przykład $1105 = 5 \cdot 13 \cdot 17$, $1729 = 7 \cdot 13 \cdot 19$ lub $2465 = 5 \cdot 17 \cdot 29$.

Przykład 2.6. Używając testu pierwszości Fermata sprawdzimy, czy liczba 561 jest liczbą pierwszą.

1. Losujemy $a \in [2, 559]$. Niech $a = 5$.
2. $d = NWD(5, 561) = 1$.
3. Kongruencja $a^{p-1} \equiv 1 \ (mod\ p)$ zachodzi, gdyż $5^{560} \equiv 1 \ (mod\ 561)$.
4. Liczba 561 może być liczbą pierwszą.

Po zastosowaniu algorytmu liczba 561 została błędnie wskazana jako liczba prawdopodobnie pierwsza. Stało się tak, ponieważ dla liczby Carmichaela równość z małego twierdzenia Fermata 1.8 zachodzi dla wszystkich a takich, że $NWD(a, p) = 1$.

Test Fermata zastosowany do liczby Carmichaela jest nieskuteczny, co przedstawione zostało w tabeli 2.4.

Lp.	L. Carmichaela	Liczba powtórzeń n	Wynik	Czas działania $[s]$
1	561	5	pierwsza (60%)	0.064905
2	561	15	pierwsza (73%)	0.097156
3	561	150	pierwsza (58%)	0.102425
4	1105	5	pierwsza (100%)	0.067776
5	1105	15	pierwsza (73%)	0.068222
6	1105	150	pierwsza (70.6%)	0.088589
7	1729	5	pierwsza (60%)	0.069128
8	1729	15	złożona (66.67%)	0.078923
9	1729	150	pierwsza (76.67%)	0.082983
10	2465	5	pierwsza (80%)	0.064879
11	2465	15	pierwsza (60%)	0.070719
12	2465	150	pierwsza (75.3%)	0.090890

Tabela 2.4. Wyniki działania testu pierwszości Fermata dla liczb Carmichaela.

Jak wynika z powyższej tabeli, liczby Carmichaela przechodzą test pierwszości Fermata z dużym prawdopodobieństwem.

2.2.2. Test pierwszości Lehmana

Test Lehmana [1, s. 60] jest kolejnym probabilistycznym testem pierwszości liczb. Algorytm testu Lehmana dla dużej liczby p wygląda następująco:

1. Losujemy dużą liczbę $a < p$.
2. Obliczamy $b \equiv a^{(p-1)/2} \ (mod\ p)$.
3. Jeśli $b \equiv 1 \ (mod\ p)$ lub $b \equiv -1 \ (mod\ p)$ to p jest liczbą pierwszą z prawdopodobieństwem większym lub równym 50%. W przeciwnym wypadku p jest liczbą złożoną.

Przykład 2.7. Stosując test pierwszości Lehmana sprawdzimy, czy 31 jest liczbą pierwszą.

1. Losujemy dużą liczbę $a < 31$. Niech $a = 21$.
2. Obliczamy $b = a^{(31-1)/2} = a^{15} \equiv 30 \ (mod\ 31)$.
3. $b = 30 \equiv -1 \ (mod\ 31)$, stąd 31 może być liczbą pierwszą z prawdopodobieństwem większym lub równym 50%.

Skrypt 3.8 przedstawia algorytm Lehmana w programie Matlab. Wyniki działania programu przedstawione zostały w tabeli 2.5.

Lp.	Liczba p	Wynik	Czas działania funkcji $[s]$
1	5	pierwsza ($\geq 50\%$)	0.023871
2	47	pierwsza ($\geq 50\%$)	0.032829
3	199	pierwsza ($\geq 50\%$)	0.031084
4	1999	pierwsza ($\geq 50\%$)	0.026158
5	6899	pierwsza ($\geq 50\%$)	0.029798
6	94 427	pierwsza ($\geq 50\%$)	0.023563
7	94 428	złożona (100%)	0.015226
8	15 472 145	złożona (100%)	0.027043
9	15 472 147	pierwsza ($\geq 50\%$)	0.026316
10	15 485 857	pierwsza ($\geq 50\%$)	0.024070

Tabela 2.5. Wyniki działania testu pierwszości Lehmana dla liczb o różnej wielkości.

Algorytm Lehmana jest bardzo szybki i zwraca prawidłowe wyniki dla liczb o różnej wielkości.

2.2.3. Test pierwszości Solovaya-Strassena

Kolejnym probabilistycznym testem pierwszości jest test Solovaya-Strassena, opracowany przez dwóch matematyków - amerykańskiego profesora Roberta M. Solovaya i profesora Volkera Strassena z Niemiec. Do wprowadzenia algorytmu testowania niezbędna jest definicja symbolu Legendre'a.

Definicja 2.2. Niech p będzie liczbą pierwszą większą od 2, a a liczbą całkowitą. *Symbol Legendre'a* $\left(\frac{a}{p}\right)$ definiuje się następująco

$$\left(\frac{a}{p}\right) = \begin{cases} 0, & \text{jeśli } a \text{ jest wielokrotnością liczby} p, \\ 1, & \text{jeśli istnieje } b \text{ takie, że } b^2 = a \ (mod\ p), \\ -1, & \text{jeśli nie istnieje żadne } b \text{ takie, że } b^2 = a \ (mod\ p). \end{cases}$$

Symbol Legendre'a można obliczyć następującym wzorem

$$\left(\frac{a}{p}\right) \equiv a^{\frac{p-1}{2}} \ (mod\ p).$$

Symbol Legendre'a może być uogólniony do *symbolu Jacobiego* $\left(\frac{a}{n}\right)$, gdzie n jest dowolną liczbą nieparzystą, która ma rozkład na czynniki pierwsze

$$n = p_1^{m_1} \cdot p_2^{m_2} \cdot ... \cdot p_k^{m_k}.$$

Wtedy symbol Jacobiego można zapisać w postaci

$$\left(\frac{a}{n}\right) = \left(\frac{a}{p_1}\right)^{m_1} \cdot \left(\frac{a}{p_2}\right)^{m_2} \cdot ... \cdot \left(\frac{a}{p_k}\right)^{m_k}.$$

Jeżeli n jest liczbą pierwszą to symbol Jacobiego i symbol Legendre'a są sobie równe.

Definicja 2.3. Jeśli n jest liczbą złożoną nieparzystą, a liczba b, względnie pierwsza z n, spełnia warunek

$$b^{\frac{n-1}{2}} \equiv \left(\frac{b}{n}\right) \ (mod\ n), \tag{2.4}$$

to n jest *pseudopierwszą liczbą Eulera* przy podstawie b.

Test pierwszości liczby n algorytmem Solovaya-Strassena przebiega następująco

1. Losujemy $b \in \{2, ..., n-2\}$.
2. Obliczamy $r = b^{\frac{n-1}{2}}\ (mod\ n)$. Jeśli $r \neq \pm 1$, to n jest liczbą złożoną.
3. Obliczamy $s = \left(\frac{b}{n}\right)$. Jeśli $r \neq s$, to n jest złożona.

Jeżeli po zakończeniu algorytmu nie ma odpowiedzi "n jest złożona", to test należy powtórzyć. Po k-krotnym powtórzeniu testu i stwierdzeniu, że n jest liczbą pseudopierwszą Eulera przy każdej z wybranych podstaw b, prawdopodobieństwo, że liczba n jest jednak złożona wynosi $\frac{1}{2^k}$.

Skrypt jednorazowego przejścia algorytmu testu pierwszości Solovaya-Strassena w programie Matlab został przedstawiony na końcu pracy (skrypt 3.9). Tabela 2.6 zawiera wyniki otrzymane za pomocą powyższego algorytmu i czas działania funkcji dla każdej operacji.

Lp.	Liczba n	Wynik	Czas działania funkcji $[s]$
1	5	pierwsza	0.034387
2	47	pierwsza	0.032534
3	199	pierwsza	0.039209
4	1999	pierwsza	0.032661
5	6899	pierwsza	0.036613
6	94 427	pierwsza	0.034677
7	94 428	złożona	0.017758
8	15 472 145	złożona	0.027369
9	15 472 147	pierwsza	0.056256
10	15 485 857	pierwsza	0.033962

Tabela 2.6. Wyniki działania testu pierwszości Solovaya-Strassena dla liczb o różnej wielkości.

2.2.4. Test pierwszości Millera-Rabina

Test pierwszości Millera-Rabina jest kolejnym testem probabilistycznym, który z pewnym prawdopodobieństwem określa, czy dana liczba jest pierwsza, czy złożona. Oryginalna wersja tego testu, stworzona przez profesora Gary'ego L. Millera, jest testem deterministycznym, lecz jego prawidłowość bazuje na uogólnionej hipotezie Riemanna, która nie została dotychczas udowodniona. W 1975 roku kryptolog Michael O. Rabin przekształcił algorytm Millera do algorytmu probabilistycznego.

Fakt 2.1. Niech n będzie nieparzystą liczbą pierwszą i niech $n-1 = 2^s \cdot t$, gdzie $2 \nmid t$. Niech $b \in \phi(n)$. Wtedy albo $b^t \equiv 1 \pmod{n}$, albo $b^{2^r t} \equiv -1 \pmod{n}$ dla pewnego $r \in \{0, 1, ..., s-1\}$.

Definicja 2.4. Niech n będzie nieparzystą liczbą złożoną i niech $n-1 = 2^s \cdot t$, gdzie $2 \nmid t$. Niech $b \in \phi(n)$. Jeżeli

- $b^t \equiv 1 \pmod{n}$ lub
- $b^{2^r t} \equiv -1 \pmod{n}$ dla pewnego $r \in \{0, 1, ..., s-1\}$

to liczbę n nazywamy *liczbą silnie pseudopierwszą* przy podstawie b.

Fakt 2.2. Jeśli n jest nieparzystą liczbą złożoną, to jest silnie pseudopierwsza przy podstawie b dla co najwyżej 25% wszystkich podstaw b takich, że $0 < b < n$.

Algorytm działania testu pierwszości Millera-Rabina przebiega następująco:

1. Obliczamy $n-1=2^s \cdot t$, gdzie t jest liczbą nieparzystą.
2. Losujemy $b \in \{2, ..., n-2\}$.
3. Obliczamy $a = b^t \pmod{n}$. Jeśli $a = \pm 1$, to n może być pierwsza.
4. Obliczamy $a = b^{2^r t} \pmod{n}$ dla $0 < r < s$. Jeśli dla pewnego r otrzymamy $a = -1$, to n może być pierwsza, w przeciwnym razie n jest złożona.

Po otrzymaniu odpowiedzi "n jest pierwsza" powtarzamy kroki algorytmu 2-4.

Skrypt testu pierwszości Millera-Rabina przedstawiony został w dalszej części pracy (skrypt 3.10). Tabela 2.7 zawiera wyniki otrzymane za pomocą powyższego algorytmu i czas działania funkcji dla każdej operacji.

Lp.	Liczba n	Wynik	Czas działania funkcji $[s]$
1	5	pierwsza	0.036212
2	47	pierwsza	0.036644
3	199	pierwsza	0.038669
4	1999	pierwsza	0.037960
5	6899	pierwsza	0.037959
6	94 427	pierwsza	0.040197
7	94 428	złożona	0.018025
8	15 472 145	złożona	0.049525
9	15 472 147	pierwsza	0.035766
10	15 485 857	pierwsza	0.045164

Tabela 2.7. Wyniki działania testu pierwszości Millera-Rabina dla liczb o różnej wielkości.

2.2.5. Chiński test pierwszości

3. Zastosowania w kryptografii

Skrypty z programu Matlab

Sito Eratostenesa

Listing 3.1. Skrypt funkcji Sito Eratostenesa.

```
function [n,p] = sitoEratostenesa(N)
   sito=1:N;
   a=2;
   while a<=sqrt(N)
      sito(a^2:a:N)=0;
      a=find(sito>a,1);
   end
   p = sito(sito>1);
   n = length(p);
end
```

Sito Sundarama

Listing 3.2. Skrypt funkcji Sito Sundarama.

```
function L = sitoSundarama(n)
P = 1:n;
for i = 1:n
     for j = i:(n-i)/(2*i+1)
          if i+j+2*i*j<=n
          P(i+j+2*i*j) = -1/2;
          end
     end
end
P = P(P>0);
P = [2, P*2+1];
L = length(P);
```

Funkcja Eulera

Listing 3.3. Skrypt funkcji Eulera.

```
function [m]=euler_phi(n)
m=0;
for i=1:n
    if gcd(i,n)==1
        m=m+1;
    end
end
end
```

Rząd multiplikatywny

Listing 3.4. Skrypt funkcji liczącej rząd multiplikatywny.

```
function [k] = mult_order(n,a)

if gcd(a,n)~=1
    disp('liczby nie są względnie pierwsze')
    return
end

k=1;
while powermod(a,k,n) ~= 1
    k=k+1;
end
end
```

Metoda naiwna

Listing 3.5. Algorytm metody naiwnej.

```
function MetodaNaiwna(N)
tic;
  X=['Liczba ', num2str(N), ' jest liczbą złożoną.'];
  Y=['Liczba ', num2str(N), ' jest liczbą pierwszą.'];
  Z=['Liczba ', num2str(N) ,...
      ' nie jest ani pierwsza, ani złożona.'];
  if (N==2 || N==3)
      disp(Y)
      return
  elseif (N==1)
      disp(Z)
      return
  else
  [n,p] = sitoEratostenesa(N);
  lp = p(p<=sqrt(N))
  l = length(lp);
 m = ones(l,1);
  end

  for i=1:l
      if mod(N, lp(i))==0
          disp(X)
          break
      else
          m(i)=0;
      end
  end
  if sum(m)==0
      disp(Y)
  end
 toc;
end
```

Test pierwszości AKS

Listing 3.6. Skrypt testu pierwszości AKS.

```
function MetodaAKS(n)

for b=2:n-1
    if round(log(n)/log(b))==log(n)/log(b)
        disp('n jest złożona')
        return
    end
end

r=2;

if gcd(r,n)~=1
    disp('n jest złożona')
    return
end

k=mult_order(r,n);

while k<=(log2(n))^2
    if gcd(r,n)~=1
        disp('n jest złożona')
         return
    end
    k=mult_order(r,n);
    r=r+1;
    if r>=n
         disp('n jest pierwsza')
         return
    end
end

for a=2:min(r,n-1)
```

```
33      if mod(n,a)==0
34          disp('n jest złożona')
35          return
36      end
37  end
38
39  if n<=r
40      disp('n jest pierwsza')
41      return
42  end
43
44  m=euler_phi(r);
45  syms x
46
47  v=sym2poly(x^r-1);
48
49  for a=1:floor(sqrt(m)*log2(n))
50      [q1,w1]=deconv(sym2poly((x+a)^n),v);
51      [q2,w2]=deconv(sym2poly(x^n+a),v);
52      w1=mod(w1,n); w2=mod(w2,n);
53      if w1~=w2
54          disp('n jest złożona')
55          return
56      end
57  end
58  disp('n jest pierwsza')
59
60  end
```

Test pierwszości Fermata

Listing 3.7. Skrypt testu pierwszości Fermata.

```
function testFermata(p,n)
    X=['Liczba ', num2str(p), ' jest liczbą złożoną.'];
    Y=['Liczba ', num2str(p), ' może być pierwsza.'];
    if (p<=0 || n<=0)
        disp('Liczby p i n muszą być większe od 0')
        return
    end
    if (p==2 || p==3)
        disp(Y)
        return
    end
    if n>(p-3)
        disp('Podaj mniejszą liczbę n')
        return
    end
    pierwsza=zeros(n,1);
    zlozona=zeros(n,1);
    a=randperm(p-3,n)+1;
    for i=1:n
        if gcd(a(i),p)>1
            zlozona(i)=1;
        elseif powermod(a(i),p-1,p)~=1
            zlozona(i)=1;
        else
            pierwsza(i)=1;
        end
    end
    if sum(pierwsza)>=sum(zlozona)
        disp(Y)
        disp('Prawdopodobienstwo wynosi:')
        disp(sum(pierwsza)/n*100)
    else
```

```
33          disp(X)
34      end
35  end
```

Test pierwszości Lehmana

Listing 3.8. Skrypt testu pierwszości Lehmana.

```
1  function testLehmana(p)
2  X = ['Liczba ', num2str(p), ' jest liczbą złożoną.'];
3  Y = ['Liczba ', num2str(p) ,...
4      ' może być pierwsza z prawdopodobieństwem >= 50%.'];
5  if mod(p,2)==0
6      disp(X)
7      return
8  end
9  a = randi([(p-1)/2,p-1]);
10 b = powermod(a,(p-1)/2,p);
11 if (mod(b,p)==1 || mod(b,p)-p==-1)
12     disp(Y)
13 else disp(X)
14 end
15 end
```

Test pierwszości Solovaya-Strassena

Listing 3.9. Skrypt testu pierwszości Solovaya-Strassena.

```
1  function testSolovaya(n)
2      X = ['Liczba ', num2str(n), ' jest liczbą złożoną.'];
3      Y = ['Liczba ', num2str(n), ' może być pierwsza.'];
4      if mod(n,2)==0
5          disp(X)
6          return
7      end
```

```
8       b = randi([2,n-2]);
9       r = powermod(b,(n-1)/2,n);
10      if (r ~= 1 && r-n ~= -1)
11          disp(X)
12          return
13      else
14          s = 1;
15          f = factor(n);
16          for i = 1:length(f)
17              s = s*powermod(b,(f(i)-1)/2,n);
18          end
19          if r~=s
20              disp(X)
21              return
22          end
23      end
24      disp(Y)
25  end
```

Test pierwszości Millera-Rabina

Listing 3.10. Skrypt testu pierwszości Millera-Rabina.

```
1   function TestMilleraRabina(n)
2       X=['Liczba ', num2str(n), ' jest liczbą złożoną.'];
3       Y=['Liczba ', num2str(n), ' może być pierwsza.'];
4       if (n==2 || n==3)
5           disp(Y)
6           return
7       end
8       if mod(n,2) == 0
9           disp(X)
10          return
11      end
12      f = factor(n-1);
```

```
    s = 0;
    for i = 1:length(f)
        if f(i) == 2
            s = s+1;
        end
    end
    t = 1;
    for j = s+1:length(f)
       t = t*f(j);
    end
    b = randi([2,n-2]);
    a1 = powermod(b,t,n);
    if (a1 == 1 || a1 == n-1)
        disp(Y)
        return
    end
    for r = 1:s
        a2(r) = powermod(b,2^(r-1)*t,n);
    end
    for i = 1:length(a2)
        if a2(i) == n-1
            disp(Y)
            return
        end
    end
    disp(X)
end
```

Bibliografia

[1] M. Karbowski, *Podstawy kryptografii*, Wydanie III, Helion, Gliwice, (2014).

[2] W. Marzantowicz, P. Zarzycki, *Elementarna teoria liczb*, Wydawnictwo Naukowe PWN, Warszawa, (2006).

[3] W. Narkiewicz, *Teoria liczb*, Wydawnictwo Naukowe PWN, Warszawa, (2003).

[4] W. Sierpiński, *Wstęp do teorii liczb*, Wydawnictwa Szkolne i Pedagogiczne, (1987).

[5] M. Zakrzewski, *Teoria liczb*, Wydanie I, Oficyna Wydawnicza GiS, Wrocław, (2017).

[6] `http://primes.utm.edu/largest.html` (dostęp 25.03.2018)

[7] `https://www.mersenne.org/primes/` (dostęp 25.03.2018)

[8] `http://mathworld.wolfram.com/PrimeSpiral.html`(dostęp 24.03.2018)

[9] `https://www.cse.iitk.ac.in/users/manindra/algebra/primality_v6.pdf` (dostęp 25.03.2018)

[10] `https://en.wikipedia.org/wiki/Multiplicative_order`

Skorowidz