4a. Teoria liczb: dzielenie całkowite i liczby pierwsze

Grzegorz Kosiorowski

Uniwersytet Ekonomiczny w Krakowie

Teoria liczb - wstęp

Teoria liczb to „matematyka matematyki” zajmująca się badaniem własności liczb - przede wszystkim całkowitych (w tym rozdziale, jeśli nie będzie wyraźnie powiedziane inaczej, wszystkie liczby są całkowite - i w czasie rozwiązywania zadań tylko takie mogą występować). Kiedyś wydawało się, że to dziedzina najbardziej abstrakcyjna z możliwych i praktycznych zastosowań mieć nie będzie. Dziś już wiemy, że to nieprawda.

Teoria liczb - zastosowania

- Współczesne systemy szyfrowania i deszyfrowania informacji (opierają się na trudności faktoryzacji liczb całkowitych - dokładniej omówimy to w kolejnej części kursu).
- Efektywne metody kompresji danych i generowania liczb pseudolosowych.
- Dyskretna Transformata Fouriera: analiza sygnałów i danych.
- Kodowanie informacji z autowykrywaniem błędów (np. numery kont bankowych i kart kredytowych, cyfrowy zapis muzyki).
- Muzykologia i automatyczne generowanie (dającej się słuchać) muzyki (Brian Eno).
- Zastosowania teorii liczb we współczesnej fizyce teoretycznej: http://empslocal.ex.ac.uk/people/staff/mrwatkin/zeta/physics.htm

Złożoność obliczeniowa podstawowych operacji

- Czas zapisu liczby n na komputerze jest $O(\log n)$.
- Złożoność dodawania liczb a i b to $O(\log a + \log b)$.
- Złożoność mnożenia liczb a i b to $O(\log a \log b)$.

Dzielenie w zbiorze liczb całkowitych

Dodawanie, odejmowanie i mnożenie w ramach liczb całkowitych nie ma szczególnie zaskakujących własności. Jednak ciekawe problemy pojawiają się przy dzieleniu, gdyż wynik dzielenia dwu liczb całkowitych w zbiorze $\mathbb{R}$ nie musi być całkowity.

Iloraz i reszta z dzielenia

Jeśli dla pewnych liczb całkowitych a, b istnieją liczby q, r takie, że $a = bq + r$ i $0 \leqslant r < b$, to q nazywamy *ilorazem* liczb a i b, a r - *resztą* z dzielenia a przez b. Zapisujemy $r = a \mod b$.

$$17 = 5 \cdot 3 + 2$$

$$2 = 17 \mod 5$$

Podzielność

Podzielność

Mówimy, że b *dzieli* a (lub a *jest podzielne* przez b, b jest *dzielnikiem* a, a jest *wielokrotnością* b), jeśli istnieje q takie, że $a = bq$, czyli, gdy $a \mod b = 0$. Zapisujemy $b|a$.

Przykład: $3|6$.

Twierdzenia o podzielności

Twierdzenia o podzielności

Dla dowolnych liczb $a, b, c \neq 0$ zachodzi:
a) jeśli $a|b$ to $a|bc$,
b) jeśli $a|b$ i $b|c$ to $a|c$,
c) jeśli $a|b$ i $a|c$ to $a|(b+c)$.

Największy wspólny dzielnik

Największy wspólny dzielnik

Największy wspólny dzielnik niezerowych liczb a i b (zapisywany jako NWD(a, b)) to największa liczba d taka, że $d|a$ i $d|b$.

Twierdzonko

Dla niezerowych liczb a, b zachodzi:
$NWD(a, 1) = 1$, $NWD(a, a) = a$, $NWD(a, b) = NWD(b, a)$.

Przykłady: $NWD(9, 6) = 3$, $NWD(36, 12) = 12$, $NWD(33, 26) = 1$,
Jednak dla większych liczb znajdowanie największego wspólnego dzielnika wymaga systematycznego podejścia.

Algorytm Euklidesa

Poniżej algorytm, który ma już około 2300 lat, pochodzący z „Elementów” Euklidesa. Służy do znajdowania największego wspólnego dzielnika liczb całkowitych.

Algorytm Euklidesa

Dane: Liczby całkowite dodatnie a, b.
Zmienne: r - liczba całkowita.

- I. Dopóki $b \neq 0$ wykonuj:
- Ia. $r := a \mod b$.
- Ib. $a := b$, $b := r$.
- **Rezultat**: Na końcu działania algorytmu a jest największym wspólnym dzielnikiem danych na początku liczb.

Czas działania tego algorytmu jest $O(\log^3 a)$, gdy $a \geqslant b$.

Algorytm Euklidesa - przykład

Przykład

Znaleźć $NWD(888, 1104)$.

Rozpoczynamy od podstawienia $a = 888$, $b = 1104$.

Wyniki algorytmu będziemy zapisywać w poniższej tabeli:

Nr kroku	r	a	b
1			
2			
...			

Algorytm Euklidesa - przykład

Przykład

Znaleźć $NWD(888, 1104)$.

Skoro $b \neq 0$, to w pierwszym kroku $r = a \mod b = 888$ (bo $a = 0 \cdot 1104 + 888$)
Następnie $a := b = 1104$, a $b := r = 888$. Zatem pierwszy krok jedynie zamienił nam kolejnością a i b (dlatego warto zaczynać algorytm Euklidesa tak, by $a > b$)

Nr kroku	r	a	b
1	888	1104	888
2			

Algorytm Euklidesa - przykład

Przykład

Znaleźć $NWD(888, 1104)$.

Nadal $b \neq 0$, więc $r = 1104 \mod 888 = 216$ (bo $1104 = 1 \cdot 888 + 216$)

Następnie $a := b = 888$, a $b := r = 216$.

Nr kroku	r	a	b
1	888	1104	888
2	216	888	216

Algorytm Euklidesa - przykład

Przykład

Znaleźć $NWD(888, 1104)$.

Nadal $b \neq 0$, więc $r = 888 \mod 216 = 24$ (bo $888 = 4 \cdot 216 + 24$)
Następnie $a := b = 216$, a $b := r = 24$.

Nr kroku	r	a	b
1	888	1104	888
2	216	888	216
3	24	216	24

Algorytm Euklidesa - przykład

Przykład

Znaleźć $NWD(888, 1104)$.

Nadal $b \neq 0$, więc $r = 216 \mod 24 = 0$ (bo $216 = 9 \cdot 24$)
Następnie $a := b = 24$, a $b := r = 0$. $b = 0$, więc algorytm jest zakończony i $NWD(888, 1104) = a = 24$.

Nr kroku	r	a	b
1	888	1104	888
2	216	888	216
3	24	216	24
4	0	24	0

Rozszerzony algorytm Euklidesa

W kryptologii bardzo przydatny będzie tak zwany rozszerzony algorytm Euklidesa, który znajduje tak zwane współczynniki Bezouta, czyli liczby x i y z poniższego twierdzenia.

Tożsamość Bezouta

Dla dowolnych liczb całkowitych dodatnich a i b istnieją liczby całkowite x i y takie, że

$$ax + by = NWD(a, b).$$

Rozszerzony algorytm Euklidesa

Rozszerzony algorytm Euklidesa - część I

Dane: Liczby całkowite dodatnie a, b.

Zmienne: r_i, q_i, x_i,y_i - ciągi liczb całkowitych, i-licznik pętli.

- I. $r_0 := a$, $r_1 := b$,$i := 1$.
- II. Dopóki $r_i \neq 0$ wykonuj:
- IIa. $i := i + 1$.
- IIb. $r_i := r_{i-2} \mod r_{i-1}$, $q_{i-1} := (r_{i-2} - r_i)/r_{i-1}$.
- Zauważmy, że w momencie zakończenia działania tej pętli obliczyliśmy już r_{i-1} =NWD(a, b) i dla wszystkich $k < i - 1$ zachodzi $r_k - q_{k+1} r_{k+1} = r_{k+2}$.

Rozszerzony algorytm Euklidesa

Rozszerzony algorytm Euklidesa - część II

- III. $i := i - 1$, $x_i := 0$, $y_i := 1$.
- IV. Dopóki $i > 1$ wykonuj:
- IVa. $i := i - 1$.
- IVb. $x_i := y_{i+1}$, $y_i := x_{i+1} - q_i x_i$.
- **Rezultat**: (x_1, y_1) są współczynnikami Bezouta. NWD jest obliczone wcześniej.

Rozszerzony algorytm Euklidesa - przykład

Przykład

Znaleźć NWD(234, 123) oraz liczby całkowite x, y takie, że $234x + 123y = NWD(234, 123)$.

Rozpoczynamy od podstawienia $a = 234$, $b = 123$.
Wyniki algorytmu będziemy zapisywać w poniższej tabeli:

i	r_i	q_i	x_i	y_i
0	?			
1	?	?	?	?
2	?	?	?	?
...	...	...	...	...

Rozszerzony algorytm Euklidesa - przykład

Przykład

Znaleźć NWD(234, 123) oraz liczby całkowite x, y takie, że $234x + 123y = NWD(234, 123)$.

Wpisujemy do tabeli w odpowiednie miejsca liczb a i b (i ustawiamy licznik $i = 1$):

i	r_i	q_i	x_i	y_i
0	234			
1	123	?	?	?
2	?	?	?	?
...	...	...	...	...

Rozszerzony algorytm Euklidesa - przykład

i	r_i	q_i	x_i	y_i
0	234			
1	123	?	?	?
2	?	?	?	?
...	...	...	...	...

$r_i = r_1 \neq 0$, więc kontynuujemy etap II: $i = 2$, $234 = 1 \cdot 123 + 111$, stąd $r_2 = 111$ i $q_1 = 1$.

Rozszerzony algorytm Euklidesa - przykład

$r_i = r_1 \neq 0$, więc kontynuujemy etap II: $i = 2$, $234 = 1 \cdot 123 + 111$, stąd $r_2 = 111$ i $q_1 = 1$.

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	?	?	?
3	?	?	?	?
...	...	...	...	...

Rozszerzony algorytm Euklidesa - przykład

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	?	?	?
3	?	?	?	?
...	...	...	...	...

$r_i = r_2 \neq 0$, więc kontynuujemy etap II: $i = 3$, $123 = 1 \cdot 111 + 12$, stąd $r_3 = 12$ i $q_2 = 1$.

Rozszerzony algorytm Euklidesa - przykład

$r_i = r_2 \neq 0$, więc kontynuujemy etap II: $i = 3$, $123 = 1 \cdot 111 + 12$, stąd $r_3 = 12$ i $q_2 = 1$.

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	1	?	?
3	12	?	?	?
4	?	?	?	?

Rozszerzony algorytm Euklidesa - przykład

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	1	?	?
3	12	?	?	?
4	?	?	?	?

$r_i = r_3 \neq 0$, więc kontynuujemy etap II: $i = 4$, $111 = 9 \cdot 12 + 3$, stąd $r_4 = 3$ i $q_3 = 9$.

Rozszerzony algorytm Euklidesa - przykład

$r_i = r_3 \neq 0$, więc kontynuujemy etap II: $i = 4$, $111 = 9 \cdot 12 + 3$, stąd $r_4 = 3$ i $q_3 = 9$.

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	1	?	?
3	12	9	?	?
4	3	?	?	?

Rozszerzony algorytm Euklidesa - przykład

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	1	?	?
3	12	9	?	?
4	3	?	?	?

$r_i = r_4 \neq 0$, więc kontynuujemy etap II: $i = 5$, $12 = 4 \cdot 3 + 0$, stąd $r_5 = 0$ i $q_3 = 4$.

Rozszerzony algorytm Euklidesa - przykład

... stąd $r_5 = 0$ i $q_3 = 4$. $r_i = 0$, więc etap II algorytmu się kończy.
$r_{i-1} = r_4 = 3$ to $NWD(234, 123)$.

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	1	?	?
3	12	9	?	?
4	3	4	?	?
5	0	-	-	-

Rozszerzony algorytm Euklidesa - przykład

W etapie III znów ustawiamy licznik $i = 4$ i wpisujemy $x_i = x_4 = 0$ oraz $y_i = y_4 = 1$. Odtąd poruszamy się „w górę tabeli".

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	1	?	?
3	12	9	?	?
4	3	4	0	1
5	0	-	-	-

Rozszerzony algorytm Euklidesa - przykład

$x_i := y_{i+1}$, $y_i := x_{i+1} - q_i x_i$.

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	1	?	?
3	12	9	?	?
4	3	4	0	1
5	0	-	-	-

$i > 1$, więc zmniejszamy licznik pętli o 1 ($i = 3$) i obliczamy:

$x_3 := y_4 = 1$; $y_3 := x_4 - q_3 x_3 = 0 - 9 \cdot 1 = -9$.

Rozszerzony algorytm Euklidesa - przykład

...$(i = 3)$ i obliczamy: $x_3 := y_4 = 1$; $y_3 := x_4 - q_3 x_3 = 0 - 9 \cdot 1 = -9$.

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	1	?	?
3	12	9	1	-9
4	3	4	0	1
5	0	-	-	-

Rozszerzony algorytm Euklidesa - przykład

$x_i := y_{i+1}$, $y_i := x_{i+1} - q_i x_i$.

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	1	?	?
3	12	9	1	-9
4	3	4	0	1
5	0	-	-	-

$i > 1$, więc zmniejszamy licznik pętli o 1 ($i = 2$) i obliczamy:

$x_2 := y_3 = -9$; $y_2 := x_3 - q_2 x_2 = 1 - 1 \cdot (-9) = 10$.

Rozszerzony algorytm Euklidesa - przykład

...$(i = 2)$ i obliczamy: $x_2 := y_3 = -9$;
$y_2 := x_3 - q_2 x_2 = 1 - 1 \cdot (-9) = 10$.

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	1	-9	10
3	12	9	1	-9
4	3	4	0	1
5	0	-	-	-

Rozszerzony algorytm Euklidesa - przykład

$x_i := y_{i+1}$, $y_i := x_{i+1} - q_i x_i$.

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	?	?
2	111	1	-9	10
3	12	9	1	-9
4	3	4	0	1
5	0	-	-	-

$i > 1$, więc zmniejszamy licznik pętli o 1 ($i = 1$) i obliczamy:

$x_1 := y_2 = 10$; $y_1 := x_2 - q_1 x_1 = -9 - 1 \cdot 10 = -19$.

Rozszerzony algorytm Euklidesa - przykład

...$(i = 1)$ i obliczamy: $x_1 := y_2 = 10$;
$y_1 := x_2 - q_1 x_1 = -9 - 1 \cdot 10 = -19$.

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	10	-19
2	111	1	-9	10
3	12	9	1	-9
4	3	4	0	1
5	0	-	-	-

Rozszerzony algorytm Euklidesa - przykład

i	r_i	q_i	x_i	y_i
0	234			
1	123	1	10	-19
2	111	1	-9	10
3	12	9	1	-9
4	3	4	0	1
5	0	-	-	-

W tym momencie $i = 1$, więc algorytm się kończy. Z tabeli można spisać wyniki: $NWD(234, 123) = 3$ oraz $234 \cdot 10 + 123 \cdot (-19) = 3$.

Najmniejsza wspólna wielokrotność

Najmniejsza wspólna wielokrotność

Najmniejsza wspólna wielokrotność dodatnich liczb a i b (zapisywana jako NWW(a, b)) to najmniejsza liczba dodatnia w taka, że $a|w$ i $b|w$.

Twierdzonko

Dla dodatnich liczb a, b zachodzi:
$NWW(a, 1) = a$, $NWW(a, a) = a$, $NWW(a, b) = NWW(b, a)$.

Przykłady: $NWW(7, 8) = 56$, $NWW(9, 6) = 18$.
Jednak dla większych liczb znajdowanie NWW wymaga systematycznego podejścia.

Najmniejsza wspólna wielokrotność - algorytmiczne wyznaczanie

Zależność NWD i NWW

Dla dodatnich liczb a, b zachodzi:

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

W szczególności zachodzi: $NWD(a, b) = \frac{ab}{NWW(a,b)}$ i $NWW(a, b) = \frac{ab}{NWD(a,b)}$.

Dzięki powyższemu twierdzeniu możemy wyznaczać NWW algorytmicznie: wystarczy wyznaczyć $NWD(a, b)$ z algorytmu Euklidesa i zastosować wzór: $NWW(a, b) = \frac{ab}{NWD(a,b)}$.

Najmniejsza wspólna wielokrotność - podzielność przez wiele liczb

W rozdziale dotyczącym zliczania elementów zbioru (kombinatoryki) przyda się twierdzenie:

Podzielność przez dwie liczby

Dla dodatnich liczb a, b zachodzi:

$$a|c \text{ i } b|c \Leftrightarrow NWW(a,b)|c.$$

Oczywiście, twierdzenie to działa również dla większej ilości liczb niż dwie.

Liczby pierwsze - wstęp

Każda liczba $a > 1$ ma **przynajmniej** 2 dzielniki: 1 i samą siebie. Jednak niektóre z liczb są pod tym względem wyjątkowe:

Liczba pierwsza

Liczba pierwsza to liczba naturalna posiadająca dokładnie 2 różne dzielniki. Liczbę naturalną większą od 1 nazywamy *złożoną*, gdy nie jest pierwsza.

Liczby 0 i 1 nie są uważane ani za pierwsze, ani za złożone. Zbiór liczb pierwszych oznaczamy $\mathcal{P}$.

Liczby względnie pierwsze

Jeśli $\mathrm{NWD}(a, b) = 1$ to a i b nazywamy *liczbami względnie pierwszymi* do siebie. Zapisujemy $a \perp b$.

Względną pierwszość możemy sprawdzić algorytmem Euklidesa.

Liczby pierwsze - znaczenie

Liczby pierwsze są kluczowe dla teorii liczb, gdyż każdą liczbę można rozłożyć na iloczyn liczb pierwszych w dokładnie jeden sposób (z dokładnością do przestawienia kolejności). Można to porównać do cząsteczek chemicznych, które może utworzyć tylko jeden układ atomów (choć oczywiście to porównanie nie jest idealne).

Rozkład (faktoryzacja) liczby całkowitej na czynniki pierwsze

Rozkład liczby całkowitej dodatniej na czynniki pierwsze to zapisanie jej w postaci iloczynu $n = p_1^{a_1} p_2^{a_2} \cdot \ldots \cdot p_k^{a_k}$, gdzie $p_1, \ldots p_k$ są różnymi liczbami pierwszymi, a $a_1, \ldots, a_k$ są liczbami całkowitymi dodatnimi.

$$600 = 2^3 \cdot 3 \cdot 5^2 = 5^2 \cdot 2^3 \cdot 3 \ldots$$

Fundamentalne twierdzenie arytmetyki

Fundamentalne twierdzenie arytmetyki

Każda liczba naturalna $n > 1$ ma jednoznaczny (z dokładnością do kolejności mnożenia) rozkład (czyli faktoryzację) na iloczyn liczb pierwszych.

Najważniejszy dla informatyki jest fakt, że to twierdzenie nie jest w żaden sposób konstruktywne tj. nie mówi, jak ten rozkład można wyliczyć. Obecnie nie jest znany żaden efektywny algorytm faktoryzujący liczby naturalne, tzn. znajdujący rozkład na iloczyn liczb pierwszych, co jest sednem współczesnych systemów kryptologicznych i innych „algorytmów zapadkowych". Oczywiście, niektóre liczby można rozłożyć stosunkowo łatwo - najtrudniejsze do rozłożenia zaś są te, które są iloczynami dwu liczb pierwszych podobnej wielkości - za znalezienie takich rozkładów wiele firm wypłaca wysokie nagrody.

Złożoność obliczeniowa

- Nie jest znany algorytm możliwy do wykonania na nie-kwantowym sprzęcie, faktoryzujący wszystkie liczby naturalne w czasie wielomianowym (choć nie istnieje dowód, że nie ma takiego).
- Na dziś (2024) najlepszym rezultatem dla faktoryzacji dowolnego n jest

$$O\left(e^{\left(\left(\frac{8}{3}\right)^{\frac{2}{3}}+o(1)\right)(\log n)^{\frac{1}{3}}(\log\log n)^{\frac{2}{3}}}\right)$$

  realizowane przez algorytm GNFS z 1993 roku. Istnieje wiele algorytmów działających w czasie „subekspotencjalnym", czyli lepszym niż $O((1+\varepsilon)^n)$ dla każego $\varepsilon > 0$.
- Istnieje algorytm Shora rozkładający liczby naturalne w czasie $O(n^3)$, ale wymaga wykorzystania komputerów kwantowych.

Testowanie pierwszości

- Choć rozkład liczby na czynniki pierwsze jest algorytmicznie nieosiągalny w sensownym czasie, to sprawdzenie, czy jakaś liczba jest pierwsza jest dużo prostsze (ważne dla „algorytmów zapadkowych").
- Teoretycznie najszybszy deterministyczny algorytm testujący pierwszość liczby to algorytm AKS (Agrawala-Kayala-Saxena) z 2002 roku, ktory działa w czasie $O(\log^6 n)$, gdzie n jest sprawdzaną liczbą. Jest natomiast niepraktyczny w zastosowaniach (zajmuje bardzo dużo czasu dla liczb „rozsądnej wielkości").
- W praktyce stosowane są testy probabilistyczne (teoretycznie nie dające 100% pewności dla każdego wyniku, ale albo dające wystarczającą praktycznie pewność, albo pewne dla „liczb rozsądnej wielkości"), ale o mniejszym czasie wykonania (np. testy Fermata, Millera-Rabina, Sollovaya-Strassena, Baille-PSW).

Twierdzenie o liczbach pierwszych

Nie istnieje „największa” liczba pierwsza, gdyż zachodzi twierdzenie (również udowodnione przez Euklidesa):

Twierdzenie o liczbach pierwszych

Liczb pierwszych jest nieskończenie wiele.

Dowód tego twierdzenia będzie zapewne jedynym dowodem w ramach tego kursu wymaganym na egzaminie (bo jest ładny).

Dowód twierdzenia o liczbach pierwszych

Twierdzenie o liczbach pierwszych

Liczb pierwszych jest nieskończenie wiele.

Dowód nie wprost: Zakładamy, że liczb pierwszych jest skończenie wiele i są to: $p_1, \ldots, p_k$. Niech $n = p_1 p_2 \ldots p_k + 1$. n jest pierwsza lub złożona. Jednak n nie może być pierwsza, bo jest większa od każdej z liczb $p_1, \ldots, p_k$, a to są wszystkie liczby pierwsze. Jeśli n jest złożona, to ma rozkład na czynniki pierwsze i w szczególności jest podzielna przez jakąś liczbę pierwszą. Ale dla danego i, wynik dzielenia n przez p_i w liczbach całkowitych to $n = p_i \cdot q + 1$, gdzie $q = p_1 p_2 \ldots p_{i-1} \cdot p_{i+1} \ldots p_k$ (iloczyn wszystkich p poza p_i), więc reszta z tego dzielenia to 1. Dlatego, n nie jest podzielna przez żadną liczbę pierwszą, więc nie może być złożona. W każdym przypadku założenie o skończoności zbioru liczb pierwszych prowadzi do sprzeczności, więc musi być fałszywe. QED.

Największa znana liczba pierwsza

Jako czystą ciekawostkę (tak wielkie liczby nie są używane nawet w kryptologii) podam, że największą **znaną** (tj. taką, której pierwszość udowodniono) liczbą pierwszą (na 1 I 2025) jest $2^{136279841} - 1$. Liczy sobie 41024320 cyfr w zapisie dziesiętnym. Jej pierwszość udowodnił w październiku 2024 roku Luke Durant z projektu GIMPS (Great Internet Mersenne Prime Search).