14. Sito Eratostenesa

W XIX w. w Kalifornii wybuchła gorączka złota. Jedną z najbardziej popularnych metod stosowanych przez poszukiwaczy cennego kruszcu było przesiewanie minerałów na sicie. Samorodki złota, które są cięższe, opadały na dno sita, a lżejsze piasek i muł były odsiewane. Okazuje się, że podobną metodę można zastosować do poszukiwania liczb pierwszych w zbiorze liczb całkowitych dodatnich. Z tego tematu dowiesz się, jak to zrobić.

Cele lekcji

- Zrozumiesz, jak działa algorytm sita Eratostenesa.
- Napiszesz program realizujący algorytm sita Eratostenesa.
- Poznasz złożoność obliczeniową algorytmu sita Eratostenesa.

Poznaliśmy wcześniej algorytmy, które pozwalają sprawdzić, czy dana liczba całkowita dodatnia jest liczbą pierwszą. Zajmiemy się teraz problemem wyznaczenia wszystkich liczb pierwszych mniejszych lub równych danej liczbie całkowitej większej od 1. Jednym z rozwiązań jest sprawdzanie, czy kolejne liczby z badanego zakresu są pierwsze, czyli wielokrotne wykorzystanie znanego już algorytmu.

Ćwiczenie 1

Oszacuj złożoność czasową algorytmu, który wyznaczałby liczby pierwsze mniejsze lub równe liczbie całkowitej n większej od 1 metodą naiwną, czyli sprawdzałby pierwszość kolejnych liczb całkowitych większych od 1 i mniejszych lub równych n.

14.1. Jak działa sito Eratostenesa?

Omówimy teraz metodę wyznaczania liczb pierwszych mniejszych lub równych liczbie n zaproponowaną już w starożytności przez greckiego matematyka Eratostenesa z Cyreny. Polega ona na usunięciu ze zbioru liczb całkowitych z zakresu od 2 do n wszystkich liczb złożonych. W ten sposób w zbiorze pozostaną tylko liczby pierwsze. Metodę tę nazwano **sitem Eratostenesa**.

Sito Eratostenesa

W praktyce algorytm polega na usuwaniu (odsiewaniu) ze zbioru liczb $\{2, \ldots, n\}$ wszystkich liczb złożonych, które są wielokrotnościami kolejnych analizowanych liczb ze zbioru, począwszy od 2.

Algorytm sita Eratostenesa

Prześledzimy teraz działanie tego algorytmu dla $n = 50$. Wyznaczymy zatem wszystkie liczby pierwsze z zakresu od 2 do 50.

Algorytm sita Eratostenesa

Zbiór wyjściowy składa się z liczb całkowitych z zakresu od 2 do 50:
{2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50}

Krok 1 Usuwamy wielokrotności liczby 2 większe od 2. Otrzymujemy zbiór:
{2, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49}

Krok 2 Usuwamy wielokrotności liczby 3 większe od 3. Otrzymujemy zbiór:
{2, 3, 5, 7, 11, 13, 17, 19, 23, 25, 29, 31, 35, 37, 41, 43, 47, 49}

Krok 3 Usuwamy wielokrotności liczby 5 większe od 5. Otrzymujemy zbiór:
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 49}

Krok 4 Usuwamy wielokrotności liczby 7 większe od 7. Otrzymujemy zbiór:
{2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47}

Otrzymany zbiór zawiera wszystkie liczby pierwsze z zakresu od 2 do 50.

Zwróć uwagę, że niektóre wielokrotności danej liczby zostały już usunięte w poprzednich krokach jako wielokrotności mniejszej liczby. Na przykład liczby 6 i 10 zostały usunięte jako wielokrotności liczby 2.

Dla liczb: 2, 3, 5, 7 pierwszymi usuwanymi wielokrotnościami były liczby: 4, 9, 25, 49, które są ich kwadratami. Jest to ogólna zasada – w każdym kroku algorytmu pierwszą usuwaną wielokrotnością jest kwadrat liczby, której krotności usuwamy. Algorytm kończy działanie, jeśli kwadrat rozpatrywanej liczby jest większy od prawego końca przedziału (liczby n). Zauważ, że każda liczba, której wielokrotności usuwamy, jest liczbą pierwszą.

Ćwiczenie 2

Prześledź działanie algorytmu sita Eratostenesa dla $n = 100$.

Zapamiętaj

Algorytm sita Eratostenesa służy do znajdowania liczb pierwszych z zakresu od 2 do n. Z podanego zbioru usuwamy wielokrotności kolejnych liczb, a pierwszą usuwaną krotnością jest kwadrat danej liczby. Kiedy kwadrat rozpatrywanej liczby jest większy od prawego końca przedziału, algorytm kończy działanie.

Warto wiedzieć

Eratostenes z Cyreny (276–194 p.n.e.) to pochodzący ze starożytnej Grecji matematyk, astronom, geograf, filozof i poeta. Jest autorem doświadczenia uznawanego za jeden z dziesięciu największych eksperymentów z fizyki – pomiaru obwodu Ziemi. Oszacował także odległość Ziemi od Księżyca i Słońca.

14.2. Implementacja algorytmu sita Eratostenesa

Określmy specyfikację problemu i zapiszmy w pseudokodzie algorytm znajdowania liczb pierwszych metodą sita Eratostenesa.

Specyfikacja

Dane: n – liczba całkowita, $2 \leqslant n \leqslant 1\,000\,000$.

Wynik: `Pierwsze` – zbiór liczb pierwszych mniejszych lub równych n.

```
Pierwsze ← {2, ..., n}
d ← 2
dopóki d * d ⩽ n wykonuj
  dla i ← d, d + 1, ..., n div d wykonuj
    Pierwsze ← Pierwsze - {i * d}
  d ← najmniejszy element zbioru Pierwsze, który jest
      większy od dotychczasowej wartości d
```

Dobra rada
Wewnętrzną pętlę możesz również zapisać następująco:

```
i ← d
dopóki i * d ⩽ n wykonuj
  Pierwsze ←
  Pierwsze - {i * d}
  i ← i + 1
```

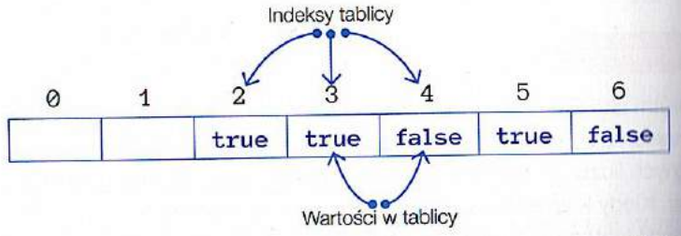
Po jednej iteracji zewnętrznej pętli zostają usunięte wielokrotności aktualnie rozpatrywanej liczby (zmienna `d`, wartość początkowa 2), następnie zmienna `d` przyjmuje kolejną wartość. Pętla jest powtarzana, dopóki kwadrat liczby jest mniejszy lub równy prawemu końcowi przedziału (`n`), w którym poszukujemy liczb pierwszych.

W wewnętrznej pętli usuwane są `i`-te wielokrotności wartości zmiennej `d`. Ponieważ iloczyn `i * d` musi być mniejszy lub równy `n`, największa wartość `i` jest równa wynikowi dzielenia całkowitego `n` przez `d` (`n div d`).

Zwróć uwagę, że następuje próba usunięcia wszystkich liczb postaci `i * d`, a niektóre z nich były już usunięte wcześniej jako wielokrotności mniejszej liczby (np. liczba 45 została usunięta jako wielokrotność liczby 3, ale algorytm wykonuje tę operację także dla liczby 5).

Do reprezentowania w programie zbioru liczb najprościej będzie wykorzystać tablicę. Zauważ, że nie musimy w niej przechowywać samych liczb. Wystarczy, że każdy element tablicy będzie zawierał informację logiczną (`true` lub `false`) określającą, czy liczba reprezentowana przez odpowiadający mu indeks tablicy jest liczbą pierwszą czy złożoną. Rysunek 14.1 przedstawia fragment tablicy `Pierwsze` z wartościami opisującymi początkowe liczby zbioru.

Dobra rada
Pamiętaj, że w języku C++ wartość `false` jest reprezentowana przez liczbę 0, a wartość `true` – przez liczbę 1.



Rys. 14.1. Fragment tablicy `Pierwsze` z zaznaczeniem liczb pierwszych i złożonych

Tablice w języku C++ są indeksowane zawsze od zera, więc zerowy i pierwszy element tablicy nie zostaną wykorzystane. Ostatni indeks w tablicy n-elementowej ma wartość $n - 1$, dlatego wygodniej będzie szukać liczb pierwszych mniejszych od n.

Oto zmodyfikowana specyfikacja problemu znajdowania liczb pierwszych, w którym wykorzystywana jest tablica wartości logicznych:

Specyfikacja

Dane: n – liczba całkowita, $2 \leqslant n \leqslant 1\,000\,000$.

Wynik: tablica wartości logicznych `Pierwsze[0..n-1]`, `Pierwsze[i]` ma wartość **`prawda`**, gdy `i` jest liczbą pierwszą, **`fałsz`** – w przeciwnym przypadku.

Ćwiczenie 3

Zapisz w pseudokodzie algorytm sita Eratostenesa zgodny z powyższą specyfikacją.

Kod źródłowy funkcji znajdującej liczby pierwsze metodą sita Eratostenesa zgodnie z powyższą specyfikacją może być taki jak poniżej. Rozmiar tablicy `Pierwsze` określa stała `N`, zdefiniowana przed funkcją.

```
1.  void SitoEratostenesa(bool Pierwsze[])
2.  {
3.      Pierwsze[2]=true;
4.      for (int i=3;i<N;i++)
5.          Pierwsze[i]=(i%2==1);
6.      int d=3;
7.      while (d*d<N)
8.      {
9.          for (int i=d;i*d<N;i+=2)
10.             Pierwsze[i*d]=false;
11.         do
12.             d+=2;
13.         while (!Pierwsze[d]);
14.     }
15. }
```

Fragment kodu źródłowego programu wyszukującego liczby pierwsze z zakresu od 2 do N – definicja funkcji realizującej algorytm sita Eratostenesa

W definicji funkcji wykorzystujemy fakt, że liczba 2 jest jedyną parzystą liczbą pierwszą, a kolejne liczby parzyste są złożone. Uwzględniamy to już podczas inicjowania tablicy `Pierwsze` (linia 1).

Elementom tablicy o parzystych indeksach większych od 2 przypisujemy wartość **false**, a pozostałym – wartość **true** (linie 4–5). Jedna iteracja głównej pętli **while** (linie 7–14) odpowiada za nadanie wartości **false** tym elementom tablicy, których indeksy są wielokrotnościami aktualnie rozpatrywanej liczby – wartości zmiennej d (linie 9–10). Aby wyznaczyć kolejną liczbę pierwszą, korzystamy z **instrukcji iteracji (pętli) do while**. Pętla ta najpierw wykonuje instrukcję, a potem sprawdza warunek powtarzania pętli. Jej składnia w języku C++ jest następująca:

Instrukcja iteracji (pętla) do while

```
do
    instrukcja;
while (warunek);
```

Warto wiedzieć

Zamiast pętli **do while** można użyć pętli **while**. Pętlę **do while** zastępujemy wówczas instrukcjami:
`instrukcja;`
`while (warunek) instrukcja;`

Instrukcja w pętli **do while** jest wykonywana, dopóki warunek przyjmuje wartość logiczną **prawda**. Ponieważ warunek jest sprawdzany po wykonaniu instrukcji, pętla wykona się co najmniej raz. Instrukcja wykonywana w pętli może być instrukcją złożoną, a więc blokiem instrukcji ujętych w nawiasy klamrowe.

Aby określić następną liczbę, której wielokrotności będziemy oznaczać przez **false**, najpierw określamy nową wartość zmiennej d, a potem sprawdzamy, czy jest ona liczbą pierwszą (linie 11–13). Co druga wielokrotność każdej liczby jest parzysta, więc usuwając krotności danej liczby, zwiększamy wartość zmiennej sterującej pętli **for** o 2 (linia 9). Analogicznie kiedy poszukujemy kolejnej liczby, której wielokrotności będziemy usuwać, zwiększamy wartość zmiennej d o 2. Pętla **do while** jest wykonywana, dopóki kolejne wartości zmiennej d są liczbami złożonymi (linie 11–13).

Dobra rada

Zapis `i+=2` odpowiada zapisowi `i=i+2`, który powoduje zwiększenie wartości zmiennej `i` o 2.

Pamiętaj, że tablice są **przekazywane do funkcji przez wskaźnik**. Wartości tablicy `Pierwsze` określone w funkcji `SitoEratostenesa` będą więc zachowane po wykonaniu funkcji. Pętla wypisująca liczby pierwsze po wywołaniu funkcji `SitoEratostenesa` może być taka jak poniżej. Wypisujemy te indeksy tablicy, które w tablicy mają wartość **true**.

Przekazanie parametru przez wskaźnik, s. 152

Fragment kodu źródłowego programu wyszukującego liczby pierwsze z zakresu od 2 do *N* – pętla odpowiadająca za wypisanie liczb pierwszych z zakresu od 2 do *N*

```
1.  for (int i=2;i<N;i++)
2.      if (Pierwsze[i]) cout<<i<<" ";
```

Ćwiczenie 4

Napisz i przetestuj program znajdujący liczby pierwsze z przedziału $[2; n]$ metodą sita Eratostenesa.

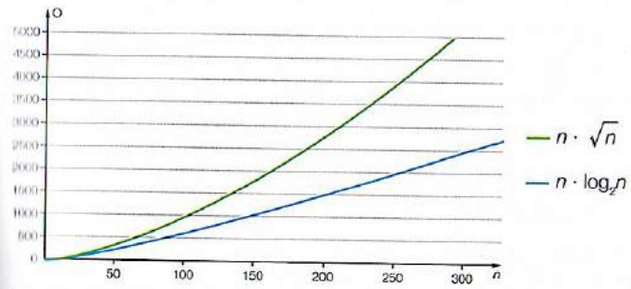
Ocena złożoności obliczeniowej algorytmu sita Eratostenesa

Zwróć uwagę, że w algorytmie sita Eratostenesa operacją dominującą jest usuwanie liczby (instrukcja `Pierwsze[i*d]=false` oraz inicjowanie tablicy). Można oszacować, że algorytm wykona tych operacji nie więcej niż $n \cdot \log_2 n$. Jest to liczba znacznie mniejsza niż $n \cdot \sqrt{n}$ prób

sprawdzania podzielności w algorytmie naiwnym, badającym pierwszość każdej liczby niezależnie (tabela 14.1 i rys. 14.2).

Liczba elementów zbioru (n)	Przybliżona liczba operacji	
	$n \cdot \log_2 n$	$n \cdot \sqrt{n}$
10	33	32
100	660	1000
1000	10000	31 600
10 000	133000	1 000 000
100 000	1660000	31620000

Tabela 14.1. Liczba operacji dla złożoności algorytmu rzędu $n \cdot \log_2 n$ i $n \cdot \sqrt{n}$



Rys. 14.2. Wykresy funkcji reprezentujących złożoności $n \cdot \log_2 n$ i $n \cdot \sqrt{n}$

Warto wiedzieć

Często obliczenia potrzebne do oszacowania złożoności algorytmu są obszerne i skomplikowane, dlatego ich nie przytaczamy.

Złożoność czasowa algorytmu sita Eratostenesa jest więc równa $O(n \cdot \log n)$. Złożoność pamięciowa wynosi $O(n)$, ponieważ algorytm sita Eratostenesa wykorzystuje tablicę n-elementową.

Rozmiar tablicy można znacznie zredukować, np. jeśli będziemy w niej przechowywać tylko informacje o liczbach nieparzystych. Wówczas i-ty element tablicy pamiętałby informację o liczbie $2i + 1$.

Warto wiedzieć

Złożoność czasową algorytmu sita Eratostenesa można oszacować dokładniej. Okazuje się, że wynosi ona $O(n \cdot \log(\log n))$.

A to ciekawe

W poszukiwaniu liczb pierwszych

Liczby pierwsze, które można zapisać w postaci $2^n - 1$, są nazywane liczbami pierwszymi Mersenne'a (od nazwiska francuskiego matematyka). W poszukiwanie kolejnych takich liczb zaangażowani są uczestnicy projektu GIMPS (ang. *Great Internet Mersenne Prime Search*), którzy użyczają w tym celu mocy obliczeniowej swoich komputerów. Dzięki zastosowaniu obliczeń rozproszonych uzyskuje się łączną moc, która odpowiada pracy dziesiątków tysięcy lat pojedynczego procesora. W czasie powstawania tego podręcznika największą liczbą pierwszą Mersenne'a jest $2^{82589933} - 1$.

Podsumowanie

- Algorytm sita Eratostenesa pozwala wyznaczyć liczby pierwsze z określonego przedziału.
- Algorytm nie bada podzielności liczb, lecz usuwa (oznacza jako liczby złożone) wielokrotności kolejnych liczb zbioru, większe od tych liczb.
- Najmniejszą wielokrotnością, którą algorytm usuwa, jest kwadrat rozpatrywanej liczby.
- Algorytm kończy działanie, gdy kwadrat liczby, której krotności ma usunąć, jest większy od prawego końca przedziału.
- Liczba operacji wykonywanych przez algorytm sita Eratostenesa jest znacznie mniejsza niż podczas sprawdzania pierwszości każdej liczby niezależnie.

Zadania

* **1** Napisz program, który wyznaczy liczby pierwsze w przedziale liczbowym podanym przez użytkownika.

* **2** Napisz program wypisujący pary liczb bliźniaczych nie większych od n, $2 \leqslant n \leqslant 1\,000\,000$. W rozwiązaniu wykorzystaj algorytm sita Eratostenesa.
Uwaga: Liczby bliźniacze to liczby pierwsze, których różnica wynosi 2. Liczbami bliźniaczymi są np. pary: 3 i 5, 5 i 7, 11 i 13.

** **3** Program utworzony w tym temacie zmodyfikuj tak, aby w tablicy była przechowywana wyłącznie informacja o liczbach nieparzystych (czyli rozmiar tablicy był dwukrotnie mniejszy).

** **4** Program wypisujący liczby półpierwsze nie większe od n, $2 \leqslant n \leqslant 1000$. Wypisywane liczby nie muszą być uporządkowane. W rozwiązaniu wykorzystaj algorytm sita Eratostenesa.
Uwaga: Liczba półpierwsza to liczba będąca iloczynem dwóch liczb pierwszych.
Liczbami półpierwszymi są np.: 4, 6, 9, 10, 14, 15.

*** **5** Napisz program realizujący algorytm sita Eratostenesa z użyciem szablonu struktury danych **set** z biblioteki STL. Struktura **set** reprezentuje w pamięci komputera zbiór.
Wskazówka: Żeby móc korzystać z typu **set**, należy dodać dyrektywę `#include <set>`.
Do dodania i usuwania elementów zbioru możesz wykorzystać metody **insert** i **erase**:

```
set<int> zbior;
zbior.insert(liczba);
zbior.erase(liczba);
```

Do pobrania kolejnego elementu zbioru możesz wykorzystać instrukcję:

```
liczba=*zbior.upper_bound(liczba);
```

Do wypisania elementów zbioru należy użyć zmiennej typu **iterator**:

```
set<int>::iterator it;
for(it=zbior.begin();it!=zbior.end();it++) cout<<*it<<" ";
```


15. Szukamy różnych podciągów

Czasami zachodzi potrzeba znalezienia fragmentu danych spełniającego pewne warunki. Na przykład trener zapisuje wyniki zawodnika osiągane podczas treningów, aby się dowiedzieć, kiedy rezultaty są coraz lepsze, a kiedy następuje spadek formy. Dużą popularnością cieszą się aplikacje, za pomocą których możemy monitorować swoją aktywność (sen, ruch, wysiłek fizyczny itp.), a potem np. sprawdzać, jaki był najdłuższy okres regularnych treningów albo kiedy spaliśmy najkrócej. W tym temacie zajmiemy się poszukiwaniem danych spełniających określone warunki.

Cele lekcji

- Obliczysz długość najdłuższego spójnego podciągu niemalejącego.
- Wyznaczysz najdłuższy spójny podciąg niemalejący.
- Poznasz i porównasz różne algorytmy znajdowania maksymalnej sumy elementów podciągu spójnego.
- Znajdziesz podciąg spójny o maksymalnej sumie elementów.

Podciąg to wybrane elementy ciągu wyjściowego zachowujące kolejność występowania. Na przykład liczby 3, 2, 5, 7 tworzą podciąg ciągu 3, 1, 2, 5, 7, 4. Jeśli elementy podciągu występują w ciągu wyjściowym obok siebie, to taki podciąg nazywamy **podciągiem spójnym**. Na przykład ciąg 1, 2, 5 jest podciągiem spójnym ciągu 3, 1, 2, 5, 7, 4. W tym temacie będziemy poszukiwać tylko podciągów spójnych.

- Podciąg
- Podciąg spójny

15.1. Szukamy długości najdłuższego spójnego podciągu niemalejącego

W **podciągu niemalejącym** każdy kolejny element jest większy lub równy poprzedniemu. Na przykład w ciągu 3, 1, 2, 5, 7, 4 możemy wyróżnić następujące spójne podciągi niemalejące:

1, 2
1, 2, 5
1, 2, 5, 7
2, 5
2, 5, 7
5, 7

W powyższym przykładzie najdłuższy spójny podciąg niemalejący tworzą liczby: 1, 2, 5, 7. Jego długość jest równa 4.
Zapiszmy specyfikację problemu znajdowania długości najdłuższego spójnego podciągu niemalejącego.

- Podciąg niemalejący