Sito Eratostenesa

Zbiór liczb naturalnych bez liczby 1 składa się z dwóch rodzajów liczb:

1. Liczb złożonych, które są podzielne przez liczby mniejsze z tego zbioru.
2. Liczb pierwszych, które nie są podzielne przez liczby mniejsze z tego zbioru.

Pierwszy rodzaj liczby to wielokrotności liczb mniejszych. Jeśli zatem ze zbioru usuniemy wielokrotności kolejnych liczb, to pozostaną w nim tylko liczby pierwsze.

Około roku 200 p.n.e grecki matematyk Eratostenes podał algorytm na znajdowanie liczb pierwszych. Nazwa pochodzi od sposobu w jaki są one znajdowane. Wszystkie liczby po kolei przesiewa się - usuwane są spośród nich wszystkie wielokrotności danej liczby.

Przykład:

Mamy zbiór liczb:

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

Rozpoczynamy od liczby 2. Wyrzucamy wszystkie jej wielokrotności:

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

Po tej operacji w zbiorze pozostaje liczba 2 oraz liczby nieparzyste. Żadna z pozostałych liczb, oprócz 2, nie dzieli się już przez 2. Teraz to samo wykonujemy z liczbą 3. Wyrzucamy ze zbioru wszystkie jej wielokrotności:

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

Operację kontynuujemy z pozostałymi liczbami:

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

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

...

I ostatecznie otrzymujemy zbiór, z którego usunięto wszystkie liczby złożone:

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

Pozostałe w zbiorze liczby są liczbami pierwszymi: 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47.

Realizując ten algorytm na komputerze musimy wybrać odpowiednią reprezentację zbioru liczb, z którego można usuwać w prosty sposób wielokrotności liczb. Użyjemy do tego celu tablicy. Poszczególne elementy tej tablicy będą reprezentowały liczby zbioru. Natomiast zawartość tych elementów będzie informacją, czy dany element pozostaje w zbiorze, czy też został z niego usunięty.

Najlepszym typem danych dla tablicy będzie typ logiczny bool. Jeśli element tablicy będzie miał wartość true, to znaczy, że reprezentowana przez jego indeks liczba znajduje się w zbiorze. Jeśli będzie miał wartość false, to ta liczba została ze zbioru usunięta. Na początku algorytmu umieszczamy true we wszystkich elementach tablicy - odpowiada to pełnemu zbiorowi liczb. Następnie do elementów, których indeksy są wielokrotnościami początkowych liczb, będziemy wpisywać false. Na końcu algorytmu wystarczy przeglądnąć tablicę i wyprowadzić indeksy tych elementów, które zachowały wartość true - nie są one wielokrotnościami żadnych wcześniejszych liczb, a zatem są liczbami pierwszymi.

Algorytm Sita Eratostenesa

Wejście:

n - określa górny kraniec przedziału <2,*n*>, w którym poszukujemy liczb pierwszych

Wyjście:

Liczby pierwsze z przedziału <2,*n*>

Dane pomocnicze:

T[ ] - tablica o elementach logicznych, których indeksy obejmują przedział <2,*n*>

i - kolejne liczby, których wielokrotności usuwamy

w - wielokrotności liczb *i*

Krok 1:	Ustaw wszystkie elementy *T*[ ] na true	
Krok 2:	*i* ← 2	*; rozpoczynamy od liczby 2*
Krok 3:	**Jeśli** *i* ≥ *n*, **to idź do** kroku 11	*; sprawdzamy, czy liczby osiągnęły koniec przedziału <2,n>*
Krok 4:	*w* ← *i* + *i*	*; liczymy pierwszą wielokrotność liczby i*
Krok 5:	**Jeśli** *w* > *n*, **to idź do** kroku 9	*; sprawdzamy, czy wielokrotność wpada w przedział <2,n>*
Krok 6:	*T*[*w*] ← false	*; jeśli tak, to usuwamy ją ze zbioru liczb*
Krok 7:	*w* ← *w* + *i*	*; obliczamy następną wielokrotność*
Krok 8:	**Idź do** kroku 5	*; i kontynuujemy usuwanie wielokrotności*
Krok 9:	*i* ← *i* + 1	*; wyznaczamy następną liczbę*
Krok 10:	**Idź do** kroku 3	*; i kontynuujemy*
Krok 11:	*i* ← 2	*; przeglądamy tablicę T[ ]*
Krok 12:	**Jeśli** *i* > *n*. **to zakończ**	
Krok 13:	**Jeśli** *T*[*i*] = true, **to pisz** *i*	*; wyprowadzamy liczby, które pozostały w T[ ]*

Krok 14: $i \leftarrow i + 1$

Krok 15: **Idź do** kroku 12