Zadanie Dzielniki – LOGIA 18 (2017/18), etap 3

Treść zadania

Napisz dwuparametrową funkcję **dzielniki** o całkowitych dodatnich parametrach a i b, której wynikiem jest liczba tych liczb całkowitych z przedziału **<a, b>** (tj. nie mniejszych niż **a** i nie większych niż **b**, zakładamy, że **a ≤ b ≤ 1000000**), które mają tylko trzy różne dzielniki naturalne.

Przykłady:

wynikiem **dzielniki(2, 6)** jest **1** (ww. własność ma tylko 4)

wynikiem **dzielniki(80000,90000)** jest **2**

Omówienie rozwiązania

Omówimy trzy algorytmy rozwiązania tego zadania.

Algorytm 1

Można, zgodnie z treścią zadania, przeglądać wszystkie liczby z przedziału <a,b>, dla każdej z nich policzyć liczbę dzielników i jeśli jest równa 3 powiększyć licznik liczb spełniających warunki zadania.

```
licznik = 0
dla liczba=a,...,b wykonuj
    jeśli liczba_dzielników(liczba) = 3 to licznik = licznik + 1
wynik licznik
```

Dla granicznych wartości przedziału <a,b> milion razy będziemy liczyć liczbę dzielników. Dlatego ważne jest, w jakim zakresie będą sprawdzane dzielniki. Warto zauważyć, że znajdując jeden dzielnik, oznaczmy go d1, mamy automatycznie drugi d2 = liczba/d1, ponieważ d1*d2 = liczba. Wyjątek stanowi sytuacja, gdy d1=d2, czyli liczba jest kwadratem. Wobec tego wystarczy sprawdzać podzielność do pierwiastka z liczby, gdyż w ten sposób wszystkie dzielniki będą wyszukane.

```
ile_dzielników = 0
dzielnik = 1
dopóki dzielnik * dzielnik < liczba wykonuj
    jeśli reszta z dzielenia liczba przez dzielnik = 0 to
           ile_dzielników = ile_dzielników + 2
    dzielnik = dzielnik + 1
jeśli dzielnik * dzielnik = liczba to
    ile_dzielników = ile_dzielników + 1
wynik ile_dzielników
```

Można także przerwać pętlę, gdy liczba dzielników przekroczy wartość 3. Warto jednak zastanowić się nad sprawniejszym algorytmem rozwiązania tego zadania, wykonującym wielokrotnie mniej operacji.

Algorytm 2

Zastanówmy się, jakie liczby mogą spełniać warunki zadania. Każda liczba (za wyjątkiem 1) ma co najmniej dwa różne dzielniki, 1 i samą siebie. Więc poszukujemy liczb, które mają dokładnie jeszcze jeden dzielnik. Są to kwadraty liczb pierwszych. Wówczas pierwiastek z liczby będzie tym trzecim dzielnikiem. Liczby spełniające warunki zadania są to więc kwadraty liczb pierwszych z przedziału `<a,b>`. Zadanie sprowadza się więc do znalezienia liczb pierwszych z przedziału $<\sqrt{a},\sqrt{b}>$. Ponieważ wartości pierwiastków mogą być niecałkowite, należy dolną granicę zaokrąglić w górę, a górną granicę przedziału w dół. W ten sposób znacznie został ograniczony przedział przeglądanych liczb.

```
licznik = 0
a = zaokrąglenie w górę pierwiastka kwadratowego z a
b = zaokrąglenie w dół pierwiastka kwadratowego z b
dla liczba=a,...,b wykonuj
    jeśli Pierwsza(liczba) to licznik = licznik + 1
wynik licznik
```

Sprawdzenie czy liczba jest liczbą pierwszą, można wykonać podobnie jak zliczanie dzielników. Należy testować dzielniki do pierwiastka z liczby. Warto zauważyć, że 2 jest jedyną liczbą pierwszą parzystą, więc w pętli wystarczy sprawdzać tylko liczby nieparzyste.

```
jeśli liczba = 1 to wynik fałsz i zakończ
jeśli liczba = 2 to wynik prawda i zakończ
dzielnik = 3
dopóki dzielnik * dzielnik ≤ liczba wykonuj
    jeśli reszta z dzielenia liczba przez dzielnik = 0 to
wynik fałsz i zakończ
    dzielnik = dzielnik + 2
wynik prawda
```

Algorytm 3

Ponieważ mamy znaleźć wszystkie liczby pierwsze z określonego przedziału warto zastosować szybszą metodę znajdowania liczb pierwszych – sito Eratostenesa. W tym celu generujemy zbiór wszystkich liczb nieparzystych nie większych od górnej granicy przedziału i usuwamy wielokrotności pierwszego elementu poczynając od jego kwadratu (pierwszym elementem będzie 3, a pierwszym usuwanym elementem 9). Następnie bierzemy kolejny element i usuwamy analogicznie jego wielokrotności. Jeśli kwadrat elementu będzie większy od górnej granicy, oznacza to, że liczby, które pozostały, są liczbami pierwszymi. Należy jeszcze dołączyć liczbę 2 jako jedyną liczbę pierwszą parzystą.

Mamy zbiór liczb pierwszych nie większych od $\sqrt{b}$, a interesują nas tylko te liczby, które są nie mniejsze od $\sqrt{a}$. Liczby mniejsze można odrzucić przeglądając je liniowo lub znaleźć miejsce pierwszej liczby nie mniejszej stosując algorytm przeszukiwania binarnego.

Rozwiązanie w języku Python

Do zapamiętania liczb pierwszych można użyć listy lub zbioru. W poniższym rozwiązaniu została użyta lista. Należy pamiętać, że elementy listy w Pythonie są numerowane od 0.

```
1.  from math import *
2.
3.  def dzielniki(a,b):
4.      if b < 4:
5.          return 0
6.      a = ceil(sqrt(a))
7.      b = floor(sqrt(b))
8.      # liczby nieparzyste od 3 do b
9.      pierwsze = [x for x in range(3,b+1,2)]
10.     # indeks liczby, której wielokrotności są usuwane
11.     i = 0
12.     # wartość liczby, której wielokrotności są usuwane
13.     d = 3
14.     # kwadrat liczby, której usuwane są wielokrotności jest w zakresie
15.     while d * d <= b:
16.         # usuwane są liczby złożone postaci j*d, poczynając od d*d, j*d musi być nie większe od b
17.         for j in range(d, b//d+1):
18.             # jeśli liczba j*d jest na liście, to należy ją usunąć
19.             if j * d in pierwsze:
20.                 pierwsze.remove(j*d)
21.         # indeks kolejnej liczby, której wielokrotności będą usuwane
22.         i += 1
23.         # wartość kolejnej liczby, której wielokrotności będą usuwane
24.         d = pierwsze[i]
25.     # dodanie na początku listy liczby 2
26.     pierwsze.insert(0,2)
27.     i = 0
28.     j = len(pierwsze)-1
29.     while i < j:
30.         s = (i+j)//2
31.         if pierwsze[s] < a: i=s+1
32.         else: j = s
33.     return len(pierwsze)-i
```

Liczba 4 jest najmniejszą mającą trzy różne dzielniki. Jeśli górna granica przedziału jest mniejsza od 4, to wynikiem jest 0. W wierszach 6 i 7 ograniczony jest przedział poszukiwań do $<\sqrt{a}, \sqrt{b}>$. W wierszach 8 – 26 realizowany jest algorytm sita Eratostenesa. W wierszu 9 generowana jest lista liczb nieparzystych nie większych od $\sqrt{b}$. W kolejnych wierszach usuwane są z listy liczby złożone, wielokrotności kolejnych początkowych liczb z listy. W wierszu 26 dodawana jest na początku listy liczba 2 jako jedyna parzysta liczba pierwsza. W wierszach 27 – 32 znajdowana jest pozycja pierwszej liczby nie mniejszej niż $\sqrt{a}$ na liście metodą przeszukiwania binarnego. Wynikiem funkcji (zadania) jest długość listy minus pozycja pierwszej liczby nie mniejszej od $\sqrt{a}$, którą wskazuje wartość zmiennej *i* (także *j*, ponieważ po zakończeniu algorytmu są równe).

Testy

Testy powinny obejmować zarówno przedziały krótkie, dla których łatwo jest ręcznie podać prawidłowy wynik, jak długie, łącznie z całym dopuszczalnym zakresem. Warto także uwzględnić przypadki szczególne: przedział, w którym nie ma żadnej liczby spełniającej warunki zadania, liczbę 1 oraz liczbę 4, jako jedyną parzystą posiadającą trzy różne dzielniki naturalne.

Wywołanie – Python	**Wynik**
dzielniki(4,25)	3
dzielniki(2,100)	4
dzielniki(111,999)	7
dzielniki(1,1000000)	168
dzielniki(1,3)	0
dzielniki(1,8)	1