Zajęcia 15

Temat: Wyszukiwanie binarne po wyniku

Czas trwania: 2x45 min

Cel zajęć:

projektuje i programuje proste problemy z różnych dziedzin, stosuje przy tym: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, tablice, rekurencję, pisze własne funkcje rekurencyjne, stosuje algorytmy sortowania, wyszukiwanie binarne wyznacza złożoność obliczeniową, testuje poprawność programów dla różnych danych, posługuje się zintegrowanym środowiskiem programistycznym przy pisaniu, uruchamianiu i testowaniu programów;

Efekty:

- umie uruchomić potrzebne oprogramowanie,
- umie napisać program z wykorzystaniem algorytmu sortującego,
- zna porządek częściowy i liniowy,
- zna algorytmy sortowania, wyszukiwanie binarne.

Formy i metody pracy: praca samodzielna, dyskusja, wykład, omówienie

Zadania do wykonania na zajęciach	Treści programowe
1. Zawody wędkarskie (zwezad)	M.3, P.2.14, A.3.2, A.3.4
2. Marchewki (marzad)	M.3, P.2.14, A.3.2, A.3.4

Materiały do zajęć:

https://www.main2.edu.pl/main2/courses/show/7/5/

Zadania do wykonania w domu:

Tort (V OIG):

https://szkopul.edu.pl/problemset/problem/G50OwjoYk8gUak-2HGHGll4o/site/?key=statement

ZADANIA I ROZWIĄZANIA:

Zadanie 1. Zawody wędkarskie

Dostępna pamięć: 32MB

Jasio organizuje zawody wędkarskie. Dokonał właśnie przeglądu wszystkich n stanowisk, jakimi dysponuje. Stanowiska umiejscowione są w linii prostej nad jeziorem w odległościach x_1, x_2, ..., x_n od początku jeziora. W czasie konkursu zawodnicy nie mogą sobie nawzajem przeszkadzać. Dlatego Jaś chciałby teraz tak rozmieścić w wędkarzy, aby najmniejsza odległość pomiędzy dwoma najbliższymi z nich była jak największa. Pomóż znaleźć Jasiowi taki rozkład wędkarzy.

Wejście

W pierwszej linii wejścia znajdują się dwie liczby całkowite: n oraz w ($2 \le w \le n \le 10^6$). W kolejnej linii znajduje się n liczb całkowitych x_i ($0 \le x_i \le 10^9, x_i - 1 < x_i$). Każda z wartości oznacza odpowiednio odległość i-tego stanowiska od początku jeziora.

Wyjście

Jedna liczba całkowita wyznaczająca najlepszą możliwą odległość pomiędzy dwoma najbliższymi stanowiskami.

Przykład

Dla danych wejściowych	poprawnym wynikiem jest
6 3 2 5 10 13 18 19	8

Ocenianie

Podzadanie	Ograniczenia	Liczba punktów
1	$n \le 10^4$	40
2	brak dodatkowych założeń	60

Rozwiązanie

Aby ułatwić rozwiązania zadania, w tablicy `a[]` będziemy pamiętać odległości pomiędzy kolejnymi stanowiskami zamiast odległości od początku jeziora oraz napiszemy funkcję `rozmieść(x)` sprawdzającą, czy jest możliwe rozmieszczenie wszystkich wędkarzy tak, aby odległość między nimi wynosiła co najmniej x. Sprawdzenie wykonamy zachłannie, usadzając wszystkich wędkarzy jak najbliżej początku jeziora.

```
oblicz_odległości()
//pamiętam największą odległość oraz odległości pomiędzy kolejnymi
//stanowiskami zamiast odległości od początku jeziora
     m ← a[n-1]
     dla i=n-1, n-2, …, 0
          a[i] ← a[i] - a[i-1];

rozmieść (x)
     suma ← 0
```

```
      //pierwszy wędkarz na pewno zajął miejsce na stanowisku 0
      //pozostało więc jeszcze W-1 wędkarzy do rozmieszczenia
      //począwszy od stanowiska 1
      j ← W-1, i ← 1
      //jeżeli nie doszedłeś do końca, a masz jeszcze wędkarzy
      dopóki i<n i j>0 wykonuj
            //dodawaj kolejne odległości aż osiągniesz x
            dopóki i<N i suma<x wykonuj
                  suma ← suma +a[i]
                  i ← i+1
            jeżeli suma ≥ x
                  j ← j-1
                  suma ← 0
      jeżeli j=0
            zwróć PRAWDA
      przeciwnie
            zwróć FAŁSZ
```

Rozwiązanie wolne: Korzystając z funkcji rozmieść będziemy próbowali rozmieścić wędkarzy o kolejne odległości. Ostatnia możliwa odległość jest rozwiązaniem zadania (wczytywanie jest pominięte).

```
oblicz_odległości()
i ← 1
dopóki rozmieść(i) = PRAWDA
      i ← i + 1
wypisz i - 1
```

Rozwiązanie szybkie: Zauważmy, że największym problemem jest wykonywanie funkcji rozmieść dla zbyt wielu różnych odległości i. Możemy znacząco zmniejszyć liczbę sprawdzanych odległości korzystając z wyszukiwania binarnego. Zauważmy, że najmniejszą możliwą do uzyskania odległością jest jeden, zaś największą – zapamiętana w funkcji `oblicz_odległości` odległość m.

Wyszukując binarnie ostatnią odległość, dla której funkcja `rozmieść(x)` zwraca prawdę musimy uważać, by w każdym kroku właściwie zawężać zakres poszukiwań. Zauważmy, że jeżeli jakieś x nie spełnia warunków zadania, to odpowiedź może wynosić co najwyżej x-1 (zmniejszamy prawy zakres). Jeżeli x spełnia warunki zadania, to również nadaje się na wynik. Aby jednak również lewy zakres zawsze ulegał zawężeniu, zaokrąglajmy wyznaczanie środka w górę (w innym wypadku program mógłby się nam „zapętlić” na dwóch ostatnich wartościach):

```
oblicz_odległości()
L ← 1, P ← m
dopóki L < P
      środek ← (L+P+1) div 2
      jeżeli rozmieść(środek) = PRAWDA
            L ← x
      przeciwnie
            P ← x - 1
wypisz środek
```

Zadanie dla uczniów: określ złożoność obliczeniową dla obu metod.

Zadanie 2. Marchewki

Dostępna pamięć: 64MB

Bitomir zasadził na swoim polu marchewki. Każda marchewka jest oddalona od innej marchewki o jeden bitometr (jednostka długości w Bajtocji) w poziomie i w pionie. Bitomir chce zakupić specjalny zbierak do marchewek. Zbierak zamontowany jest na długim ramieniu i potrafi zebrać wszystkie marchewki w promieniu swojego działania. Bitomir zastanawia się teraz, jakiej długości ramię (w bitometrach) musi posiadać zbierak, by jednorazowo zebrał co najmniej *m* marchewek. Cena zbieraka zależy od długości ramienia. Bitomir chce zakupić jak najtańsze urządzenie. Pomóż mu wyznaczyć najlepszy zbierak!

Wejście

W pierwszej linii wejścia znajduje się jedna liczba całkowita m ($2 \le m \le 5 \cdot 10^{12}$), oznaczająca liczbę marchewek, którą chcemy jednorazowo zebrać.

Wyjście

Na wyjściu powinna znaleźć się jedna liczba całkowita oznaczająca minimalną długość ramienia zbieraka.

Przykład

Dla danych wejściowych	poprawnym wynikiem jest
13	2

Ocenianie

Podzadanie	Ograniczenia	Liczba punktów
1	$n \le 10^5$	15
2	$n \le 10^8$	15
3	brak dodatkowych założeń	60

Rozwiązanie

Rozwiązanie wolne 1: W każdym kwadracie o boku 2r obliczamy liczbę marchewek wewnątrz koła o promieniu r. Wypisujemy takie r, dla którego udało zebrać się co najmniej m marchewek:

```
r ← 0
wykonuj
      r ← r + 1
      liczba_marchewek ← 0
      dla x=-r, -r+1, …, r wykonuj
            dla y=-r, -r+1, …, r wykonuj
                  jeżeli x^2 + y^2 ≤ r^2
                        liczba_marchewek ← liczba_marchewek + 1
dopóki liczba_marchewek < m
wypisz r
```

Rozwiązanie ulepszone: Spróbujmy znacząco przyśpieszyć obliczanie liczby marchewek wewnątrz koła o promieniu r. Zauważmy, że dla x równego 0 nad osią 0x znajduje się dokładnie r marchewek. Po zwiększeniu x o 1 liczba ta może wyłącznie zmaleć. Odnajdźmy to y znajdujące się wewnątrz koła o promieniu r. Następnie dalej zmniejszajmy x aż do r. Zauważmy, że w ten sposób każde x oraz każde y zmniejszamy dokładnie r razy. Działanie to zawrzyjmy w funkcji ile_marchewek(r).

```
ile_marchewek (r)
      ile ← 0
      y ← r
      dla x=0,1,…, r wykonuj
            dopóki x² + y² ≤ r² wykonuj
                  y ← y - 1
            ile ← ile + y
      //rozwiązanie dla pierwszej ćwiartki mnożymy przez 4
      //dla czterech ćwiartek i dodajemy punkt (0,0)
      zwróć 4·ile+1
```

Główna część programu:

```
r ← 0
dopóki ile_marchwek(r) < m wykonuj
      r ← r + 1
wypisz r
```

Nadal jednak nasz program działa za wolno.

Rozwiązanie szybkie: Ponownie największy problem stanowi sprawdzanie wszystkich możliwych r (obliczania liczby marchewek wewnątrz okręgu niestety nie da się już przyśpieszyć). Zauważmy jednak, że potrafimy założyć najmniejszy oraz największy rozmiar ramienia urządzenia. Możemy też wyznaczać liczby marchewek dla długości pośrednich. Z pomocą przyjdzie nam więc wyszukiwanie binarne.

Jako minimalny promień przyjmijmy 0, jako maksymalny 2 000 000. Wyznaczmy środek dla tych dwóch wartości i obliczmy liczbę marchewek wewnątrz (funkcja ile_marchewek(r) z poprzedniego rozwiązania). Jeżeli jest równa m, zakończmy wyszukiwanie, za duża - promień może być równy bądź mniejszy, jeżeli jest zbyt mała, promień na pewno jest większy. Zadbajmy przy tym o zmniejszanie przedziału przeszukiwań w każdym kroku zarówno z lewej, jak i z prawej strony:

```
L ← 0, P ← 2000000
dopóki L < P wykonuj
      r ← (L+P) div 1 // zaokrąglenie w dół
      ile ← ile_marchewek(r)
      jeżeli  ile = m
            przerwij pętlę
      jeżeli ile > m
            P ← r
      przeciwnie
            L ← r + 1
wypisz r
```

Zadanie dla uczniów: określić liczbę operacji wykonywaną przez funkcję ile_marchewek w zależności od r. Określić złożoność obliczeniową każdego z trzech rozwiązań.