Zajęcia 12

Temat: Sortowanie 1

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, 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.

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

Zadania do wykonania na zajęciach	Treści programowe
1. Latarnie	M.3, P.2.14, A.3.4
2. Najczęściej występujący (najc2)	M.3, P.2.14, A.3.4
3. Ciąg Farey'a (fare)	M.3, P.2.14, A.3.4

Materiały do zajęć:

https://www.main2.edu.pl/main2/courses/show/7/13/

Zadania do wykonania w domu:

Minimalna liczba (II OIG):

https://szkopul.edu.pl/problemset/problem/8-Z3rWfeUiuDVT7E9Yc3LTFt/site/

Deski kontratakują (XIV OIG):

https://szkopul.edu.pl/problemset/problem/2hydlzVm5wFFvOONGbTbm0Ow/site/

ZADANIA I ROZWIĄZANIA:

Zadanie 1. Latarnie

Dostępna pamięć: 32MB

Samotne długie noce i ostre noże w kieszeniach
Wskazówki zegarka świecą w ciemnościach
nie, nie, nie
Kroki na schodach, w bezruchu zamierasz
Tu czai się strach, tak ciemno tu teraz
nie, nie, nie
Właśnie tak, tak wygląda moje miasto nocą,
tak wygląda nocą świat
nie, nie, nie
tak, tak wygląda moje miasto nocą,
tak wygląda nocą świat

De Mono: Moje miasto nocą

Nie każdy lubi chodzić nocą ulicami, na których latarnie dają zbyt mało światła. Ciemności boi się również pan Integer. Wzdłuż ulicy, na której mieszka, postawiono k latarni. Pan Integer uważa, że jest ich stanowczo za mało. Albo za słabo świecą. Ponieważ zaplanowano właśnie wymianę wszystkich żarówek na nowe (energooszczędne), pan Integer postanowił zwrócić się do miejscowego zarządu dróg z nietypową prośbą. Chciałby, aby każda latarnia świeciła światłem z odpowiednią jasnością co najmniej o promieniu r tak, aby cała ulica była oświetlona. Nie chce przy tym narażać miasta na dodatkowe koszty, dlatego chce wyznaczyć r tak, by było jak najmniejsze. Pomożesz mu?

Wejście

W pierwszym wierszu standardowego wejścia znajdują się dwie liczby całkowite d oraz k – odpowiednio długość ulicy, na której mieszka pan Integer oraz liczba latarni ($1 \le k \le d \le 10^6$). W kolejnej linii znajduje się k liczb całkowitych x_i – odległości i-tej latarni od początku ulicy ($0 \le x_i \le d$). Latanie mogą stać w dowolnym miejscu ulicy. W jednym miejscu znajduje się tylko jedna latarnia.

Wyjście

Minimalny promień światła r, dla którego każdy punkt ulicy zostanie rozświetlony. Wynik wypisz z dokładnością do jednego miejsca po przecinku.

Przykład

Wejście	Wyjście
`10 3` `1 5 9`	`4.0`

Rozwiązanie

Dane należy wczytać do tablicy, posortować ją, a następnie znaleźć największą różnicę pomiędzy dwoma sąsiednimi elementami. Wynik (podzielony przez 2) należy przyrównać do odległości pierwszego elementu od początku oraz ostatniego od końca i wybrać największy z nich.

Do sortowania można użyć polecenia `sort` z biblioteki STL.

Zadanie 2. Najczęściej występujący

Dostępna pamięć: 32MB

Wyznacz najpopularniejszy element ciągu, to znaczy taki, który występuje w nim największą liczbę razy. W przypadku, gdy w ciągu jest więcej niż jeden najpopularniejszy element, podaj największy z nich.

Wejście

W jednej linii znajduje się ciąg liczb całkowitych zakończony liczbą 0. Liczb jest nie więcej niż 1000000, ich wartość mieści się w przedziale $[-10^{18}, 10^{18}]$.

Wyjście

Najpopularniejszy element ciągu.

Przykład

Wejście	Wyjście
1 2 -5 7 1 3 3 7 3 1 0	3

Rozwiązanie

Ze względu na zakres danych wejściowych nie będzie możliwe zliczenie w tablicy wystąpień każdego z elementów. Dane należy wczytać do tablicy `t[]` i posortować ją. Elementy o tej samej wartość znajdują się obok siebie. Wystarczy je zliczyć i wybrać wartość maksymalną.

```
sortuj(t,t+n)
m ← 0, ile ← 1
dla i=1,2,…,n-1
      jeżeli t[i]=t[i-1]
            ile ← ile + 1
      przeciwnie
            ile ← 1
      jeżeli m < ile
            m ← ile
wypisz m
```

Zadanie 3. Ciąg Farey'a

Dostępna pamięć: 32MB

Ciąg Farey'a dla liczby naturalnej N (N > 1) to ciąg ułamków, których licznik i mianownik są liczbami naturalnymi nieprzekraczającymi liczby N oraz licznik nie jest większy od mianownika. Ułamki powinny być skrócone, posortowane rosnąco i nie mogą się powtarzać. Oto przykład ciągu Farey'a dla N = 4:

$$\frac{0}{1}\ \frac{1}{4}\ \frac{1}{3}\ \frac{1}{2}\ \frac{2}{3}\ \frac{3}{4}\ \frac{1}{1}$$

Napisz program, który czyta liczbę N i wypisuje odpowiadający jej ciąg Farey'a.

Wejście

Pierwszy i jedyny wiersz danych zawiera jedną liczbę naturalną N (3 ≤ N ≤ 1000).

Wyjście

Program powinien wypisać w jednym wierszu ciąg ułamków zapisanych przy użyciu znaku „/” (np. „1/2”) oddzielonych pojedynczymi odstępami, stanowiącymi ciąg Farey’a dla liczby N.

Przykład

Wejście	Wyjście
4	0/1 1/4 1/3 1/2 2/3 3/4 1/1

Rozwiązanie

Najprostszym sposobem rozwiązania zadania będzie wygenerowanie wszystkich możliwych ułamków z podanego zakresu oraz zapamiętanie w tablicy tych nieskracalnych(funkcja `__gcd` z STL zwraca największy wspólny dzielnik).

W językach C/C++ stosunkowo proste wydaje się skorzystanie ze struktury, która będzie przechowywać liczniki i mianowniki.

```
struct ulamek { int l, m;};
ulamek u[1000000];
```

Aby skorzystać z sortowania z biblioteki STL, konieczne jest w tym wypadku skorzystanie z własnej funkcji porównującej (z porównania usunęliśmy dzielenie):

```
bool operator < (const ulamek &a, const ulamek &b) {
     return (a.l*b.m)<(b.l*a.m);
}
```

Główna część programu:

```
k=1;
for (int i=1; i<n;i++)
     for (int j=n; j>=2; j--)
          if (__gdc(i,j)==1 && j>i) {
               u[k].l=i; u[k].m=j;
               k++;

sort(u,u+k);
for(int i=0; i<k;i++) printf("%d/%d ", u[i].l, u[i].m);
```

Pamiętajmy o wypisaniu pierwszego i ostatniego elementu ciągu!