Zajęcia 14

Temat: Wyszukiwanie binarne

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, omówienie, wykład, dyskusja

Zadania do wykonania na zajęciach	Treści programowe
1. Znaczki	M.3, P.2.14, A.3.2, A.3.4
2. Tunele	M.3, P.2.14, A.3.2, A.3.4
3. Bierki	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:

Statki kosmiczne (II OIG):

https://szkopul.edu.pl/problemset/problem/isVbAEInYlrdxQNvE6mEnS9i/site/?key=statement

Kalendarze (III OIG):

https://szkopul.edu.pl/problemset/problem/LWpMcXylQBa6wHzcJ6U7axzK/site/?key=statement

ZADANIA I ROZWIĄZANIA:

Zadanie 1. Znaczki

Limit pamięci: 32MB

Bitek i Bajtek zbierają znaczki. Bitek planuje wziąć udział w wystawie połączonej z konkursem na największy zbiór znaczków. Aby zwiększyć swoje szanse na odniesienie spektakularnego sukcesu, postanowił pożyczyć od Bajtka te znaczki, których sam nie posiada. Chciałby teraz wiedzieć, o ile znaczków zwiększy się jego kolekcja.

Wejście

W pierwszej linii wejścia znajdują się dwie liczby naturalne *n* (mniejsza niż 100 000) i *m* (mniejsza niż 10 000 000), oznaczające odpowiednio liczbę znaczków Bitka oraz znaczków Bajtka. W drugiej linii wejścia znajduje się *n* różnych liczb naturalnych mniejszych niż miliard – typy znaczków Bitka. W trzeciej linii wejścia znajduje się *m* różnych liczb naturalnych mniejszych niż miliard – typy znaczków Bajtka. Możesz założyć, że w Bitek i Bajtek podali każdy posiadany znaczek tylko raz.

Wyjście

W jedynym wierszu wyjścia znajduje się jedna liczba całkowita: liczba znaczków w kolekcji Bajtka, których nie posiada Bitek.

Przykład

Dla danych wejściowych:	poprawną odpowiedzią jest:
6 4 1 2 3 4 5 6 2 4 5 7	1

Rozwiązanie

Rozwiązanie poprawne: wystarczy posortować pierwszy zestaw *n* znaczków (`a[]`), a następnie dla każdego z *m* znaczków z drugiego zestawu sprawdzać wyszukiwaniem binarnym czy wystąpił on w pierwszym zestawie (funkcja sort w bibliotece STL).

```
znajdź (x)
      L ← 0, P ← n - 1
      dopóki (L≤P)
            k ← (L+P) div 2
            jeżeli a[k] = x
                  zwróć PRAWDA
            jeżeli a[k] < x
                  L ← k+1
            przeciwnie
                  P ← k-1
      zwróć  FAŁSZ
```

Główna część programu:

```
sort(a, a+n)
dla i=0,1,…,m
```

```
      wczytaj x
      jeżeli znajdź(x)=FAŁSZ
           licznik ← licznik + 1
wypisz licznik
```

Zadanie 2. Tunele

Limit pamięci: 32MB

Janek zarządza dużą firmą dostawczą. Właśnie zamierza opracować nowy plan dostaw do swoich kontrahentów. Do wszystkich można dojechać nowo wybudowaną autostradą. Niestety samochody na autostradzie muszą pokonywać również tunele. Każdy z samochodów oraz każdy z tuneli ma określoną wysokość. Aby ciężarówka Janka przejechała pod tunelem, musi być niższa niż wysokość tunelu. Janek zna wysokości ciężarówek i tuneli i zastanawia się, do którego miejsca autostrady dojedzie każde z jego aut.

Wejście

W pierwszej linii wejścia znajduje się dwie liczby całkowite a i b ($1 \le a, b \le 500\,000$), odpowiednio ilość tuneli i samochodów. W drugiej linii znajduje się a oddzielonych spacjami liczb całkowitych $t_1, t_2, \ldots, t_a$ ($1 \le t_i \le 1\,000\,000\,000$), gdzie t_i oznacza wysokość i-tego tunelu. W trzeciej linii znajduje się b oddzielonych spacjami liczb całkowitych $s_1, s_2, \ldots, s_b$ ($1 \le s_i \le 1\,000\,000\,000$), gdzie s_i oznacza wysokość i-tego samochodu.

Wyjście

Pierwszy i jedyny wiersz wyjścia zawiera b liczb całkowitych $d_1, d_2, \ldots, d_i$, gdzie d_i oznacza numer ostatniego tunelu, przez który może przejechać i-ty samochód lub 0, jeśli nie może przejechać przez żaden z tuneli.

Przykład

Dla danych wejściowych:	poprawną odpowiedzią jest:
5 3 8 6 5 7 9 4 7 9	5 1 0

Rozwiązanie

Rozwiązanie wolne: Możemy symulować dla każdego z samochodów jazdę przez kolejne tunele, dopóki wysokość tunelu jest wystarczająca. To rozwiązanie będzie działać w czasie O(a·b).

Rozwiązanie szybkie: Przez kolejny tunel może przejechać samochód równy jego wysokości (minus jeden), jeżeli po lewej jego stronie nie znajdzie się żaden niższy tunel. W przeciwnym wypadku ten niższy tunel zadecyduje o maksymalnej możliwej wysokości samochodu. Zapamiętajmy więc dla każdego tunelu najniższą możliwą wysokość prześwitu w tablicy `t[]`. Teraz możemy dla każdego samochodu wyszukać odpowiedni tunel binarnie. Możemy wykorzystać funkcje `reverse` oraz `upper_bound` z biblioteki STL w celu uproszczenia obliczeń:

```
min=1000000000;
```

```
scanf("%d %d",&a,&b);
for(int i=0; i<a; i++) {
    scanf("%d",&t[i]);
    if (t[i]<min) min=t[i];
    t[i]=min;
}
reverse(t,t+a);
for (int i=0; i<b; i++) {
    scanf("%d",&s);
    int *p = upper_bound(t, t+a, s);
    printf("%d ",a-(int)(p-t));
}
```

Zadanie 3. Bierki (OIG)

Limit pamięci: 32MB

Jaś lubi budować trójkąty z bierek. W tym celu trzyma je w worku, z którego wybiera trzy bierki na chybił-trafił. Bierki mogą mieć różne długości i nie zawsze Jaś może zbudować trójkąt, a wtedy wpada w histerię. Mama Jasia ma dość histerycznych napadów synka i dlatego poprosiła Ciebie o pomoc. Należy odrzucić niektóre bierki w taki sposób, aby z pozostałych zawsze dało się ułożyć trójkąt, jednocześnie zostawiając jak najwięcej bierek w worku.

Zadanie

Opracuj program, który:

- wczyta ze standardowego wejścia liczbę bierek w worku oraz ich długości,
- obliczy największą liczbę bierek, którą można pozostawić w worku, tak żeby z każdych trzech z nich można było utworzyć trójkąt,
- wypisze wynik na standardowe wyjście.

Wejście

W pierwszym wierszu zapisano liczbę n ($5 \leq n \leq 30\,000$), oznaczającą liczbę bierek w worku. W każdym z n następnych wierszy zapisano długość jednej bierki. Długość bierki jest liczbą całkowitą z przedziału [1, 500].

Wyjście

W pierwszym wierszu wypisz liczbę bierek, które powinny zostać w worku.

Przykład

Dla danych wejściowych:	poprawną odpowiedzią jest:
10 7 1 2 8 10 6	7

1 7 9 9	

Rozwiązanie

Rozwiązanie wolne: Dla każdej pary dwóch różnych bierek sprawdźmy, ile bierek jest mniejszych niż suma dwóch pierwszych. Takie rozwiązanie będzie działało w czasie $O(n^3)$.

Rozwiązanie szybkie: Posortujmy bierki niemalejąco. Dla każdej bierki i oraz i+1 znajdźmy (wyszukiwaniem binarnym) pierwszą bierkę, której długość jest równa lub większa od sumy bierek i oraz i+1. Największy otrzymany w ten sposób przedział stanowi rozwiązanie zadania.

```
sort(a,a+n);
for (int i=0;i<n-1;i++)
    {
       int* p = lower_bound(a,a+n,a[i]+a[i+1]);
       ilosc=(p-a)-i;
       if(ilosc > bierki)
            bierki=ilosc;
    }
```