Deski (rozwiązanie)

XIV OIJ, zawody I stopnia, tura otwarta
30 września – 16 grudnia 2019

Autorzy zadania: **Bartosz Łukaszewicz, Jacek Tomasiewicz**
Opracowanie: **Dominik Klemba, Michał Niedziółka**

Olimpiada Informatyczna Juniorów

W tym zadaniu musimy spośród danych N desek wybrać 4, z których możemy zbudować kwadratową piaskownicę o jak największym polu. Deski można przy tym skracać.

Jeżeli liczba desek jest mniejsza niż 4 to piaskownicy oczywiście nie da się zbudować, ponieważ każdą deskę możemy wykorzystać tylko raz.

Jeśli zaś desek jest co najmniej 4, musimy wybrać pewne cztery z nich i skrócić tak, aby wszystkie były równe – czyli do najmniejszej wybranej deski. Aby więc uzyskać największą możliwą piaskownicę, wybieramy po prostu cztery największe deski. Pole takiej piaskownicy będzie kwadratem długości czwartej z nich.

Zadanie zatem sprowadza się do znalezienia czwartej największej deski. Jedną z najszybszych dróg do tego celu jest posortowanie wszystkich długości (używając funkcji `sort`) i wypisanie odpowiedniej długości podniesionej do kwadratu.

des.cpp

```cpp
#include "bits/stdc++.h"

using namespace std;

// Dobrym zwyczajem algorytmicznym jest ustawienie
// rozmiaru tablicy na trochę wiekszy niz wymagany.
const int MAX_N = 1000 * 1000 + 5;

int L[MAX_N];

int main() {
  ios_base::sync_with_stdio(0);

  // Deklarujemy i wczytujemy zmienną N - liczbę desek,
  int N;
  cin >> N;

  // a następnie same długości desek do tablicy L.
  for (int i = 0; i < N; ++i) {
    cin >> L[i];
  }

  // Sortujemy długości desek używając funkcji z biblioteki standardowej.
  sort(L, L + N);

  // Jeżeli desek jest mniej niz 4, to nie da się zbudować piaskownicy.
  // W przeciwnym przypadku budowa zawsze jest możliwa.
  if (N < 4) {
    cout << "0\n";
  } else {
    // Tablica L jest typu int, ale wynik może byc typu long long.
    // Dlatego musimy przed mnozeniem wykonać rzutowanie inta na long longa.
    cout << (long long)L[N - 4] * L[N - 4] << "\n";
  }

  return 0;
}
```

Warto zwrócić uwagę, że w języku C++ wynik nie zmieści się w typie `int`, dlatego trzeba go najpierw przekonwertować na typ `long long`, przed wykonaniem mnożenia.

Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

des.py

```python
# Wczytujemy liczbę desek.
N = int(input())

# Wczytujemy długości desek.  Wszystkie elementy konwertujemy
# z napisów na liczby.
L = [int(l) for l in input().split()]

# Sortujemy długości desek używając funkcji bibliotecznej.
L.sort()

# Jeżeli desek jest mniej niz 4, to nie da sie zbudować piaskownicy.
if N < 4:
    print(0)
# W przeciwnym przypadku budowa zawsze jest możliwa.
else:
    print(L[N-4] ** 2)
```

Oczywiście można było, zamiast używać gotowej funkcji `sort`, samemu napisać wybrany algorytm sortowania, trzeba jednak zwrócić uwagę na jego złożoność – liczba desek n może być równa nawet $1\,000\,000$. Przy takiej liczbie wystarczająca złożoność to $O(n \log n)$, którą można było uzyskać implementując algorytm szybkiego sortowania (*quick sort*), bądź sortowania przez scalanie (*merge sort*). Algorytmy wolniejsze, o złożoności kwadratowej, takie jak sortowanie przez wstawianie (*insert sort*), wybieranie (*selection sort*), bądź sortowanie bąbelkowe (*bubble sort*) nie przechodziły największych testów i otrzymywały 63 punktów.

Istnieje inne, szybsze rozwiązanie tego zadania, korzystające z faktu, że największy element tablicy potrafimy za pomocą jednej pętli, w czasie liniowym. Rozwiązanie takie przechodzi trzykrotnie po tablicy i przy każdym przejściu największy element zamienia na zero (czyli na element mniejszy od dopuszczalnych).

Po tej operacji przejdziemy po tablicy jeszcze raz i znajdziemy element największy, który po podniesieniu do kwadratu będzie odpowiedzią. Taki algortm działa w czasie liniowym względem długości tablicy.

des_alt.py

```python
# Pomocnicza funkcja, która zwraca indeks maksymalnego elementu tablicy L.
def znajdzMax(L, N):
  indeks_maxa = 0
  # Przechodzimy przez całą tablicę,
  for i in range(1, N):
    # jeżeli znajdziemy element tablicy większy od aktualnego maksimum,
    if L[indeks_maxa] < L[i]:
      # to aktualizujemy indeks na maksimum.
      indeks_maxa = i
  return indeks_maxa

# Wczytujemy liczbę desek.
N = int(input())

# Wczytujemy długości desek.  Wszystkie elementy konwertujemy
# z napisów na liczby.
L = [int(l) for l in input().split()]

# Trzykrotnie znajdujemy maksymalny element i zamieniamy go na zero.
for i in range(3):
    indeks_maxa = znajdzMax(L, N)
    L[indeks_maxa] = 0

# Znajdujemy czwarty największy element i wypisujemy go.
# Jeżeli elementów tablicy L jest mniej niż 4, to wtedy wszystkie są
# zerami, więc odpowiedzią będzie 0, jak jest wymagane.
indeks_czwartego = znajdzMax(L, N)
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".

```python
print(L[indeks_czwartego] * L[indeks_czwartego])
```

des_alt.cpp

```cpp
#include "bits/stdc++.h"

using namespace std;

// Dobrym zwyczajem algorytmicznym jest ustawienie
// rozmiaru tablicy na troche wiekszy niz wymagany.
const int MAX_N = 1000 * 1000 + 5;

int L[MAX_N];

// Pomocnicza funkcja, która zwraca indeks maksymalnego elementu tablicy L.
int znajdzMax(int N) {
  int indeks_maxa = 0;
  // Przechodzimy przez całą tablicę,
  for (int i=1; i<N; i++) {
    // jeżeli znajdziemy element tablicy większy od aktualnego maksimum,
    if (L[indeks_maxa] < L[i]) {
      // to aktualizujemy indeks na maksimum.
      indeks_maxa = i;
    }
  }
  return indeks_maxa;
}

int main() {
  ios_base::sync_with_stdio(0);

  // Deklarujemy i wczytujemy zmienną N - liczbę desek,
  int N;
  cin >> N;

  // a następnie same długości desek do tablicy L.
  for (int i = 0; i < N; ++i) {
    cin >> L[i];
  }

  // Trzykrotnie znajdujemy maksymalny element i zamieniamy go na zero.
  for (int i=0; i<3; i++) {
    int indeks_maxa = znajdzMax(N);
    L[indeks_maxa] = 0;
  }

  // Znajdujemy czwarty największy element i wypisujemy go.
  // Jeżeli elementów tablicy L jest mniej niż 4, to wtedy wszystkie są
  // zerami, więc odpowiedzią będzie 0, jak jest wymagane.
  int indeks_czwartego = znajdzMax(N);
  cout << (long long)L[indeks_czwartego] * L[indeks_czwartego] << "\n";
}
```


Olimpiada finansowana jest ze środków Ministerstwa Edukacji Narodowej w ramach zadania publicznego „Organizacja i przeprowadzenie olimpiad i turniejów w latach szkolnych 2019/2020, 2020/2021, 2021/2022".