Zadanie: SKA
Skarbiec

Olimpiada Informatyczna

Eliminacje do IOI, dzień drugi. Plik źródłowy `ska.*` Dostępna pamięć: 256 MB. *13.08.2020*

To miała być spokojna noc. Bajzek siedział przed monitorami, pilnując budynku. Pracował w Bajtockim Banku Centralnym. Praca niezbyt ciekawa, ale płacili przyzwoicie, więc Bajzek nie narzekał. Nagle spostrzegł, że sygnał z jednej kamery jest zakłócony. Postanowił to sprawdzić. Gdy dotarł na miejsce, ujrzał płomienie buchające z jednego z pomieszczeń.

– Muszę sprawdzić co ze skarbcem... – pomyślał Bajzek i pobiegł na piętro wyżej, gdzie mieściło się pomieszczenie z pieniędzmi. – Trzeba je przenieść w bezpieczne miejsce – wymamrotał pod nosem, ale po chwili zorientował się, że nie może tak po prostu zapakować wszystkich banknotów i ich wynieść...

Pieniądze w skarbcu ułożone są w n stosach. Każdy stos zawiera pewną liczbę plików z banknotami. Jeden ze stosów jest jednak inny: zawiera on fałszywe banknoty wypełnione farbą, która wybucha, gdy opuści skarbiec. Jest to jedno ze słynnych zabezpieczeń Bajtockiego Banku Centralnego. Fałszywy plik banknotów jest nieco cięższy od prawdziwego: waży on 101 gram, podczas gdy prawdziwy waży 100 gram. Tylko jeden stos zawiera fałszywe pliki banknotów; stos ten nie zawiera prawdziwych plików.

W rogu skarbca stoi bardzo precyzyjna waga elektroniczna, umożliwiająca poznanie wagi dowolnego zbioru plików. Niestety działa ona bardzo wolno, a czasu nie jest za dużo.

Napisz program komunikujący się z biblioteką służącą do obsługi wagi, który znajdzie fałszywy stos banknotów, wykonując minimalną liczbę ważeń.

Komunikacja

Aby użyć biblioteki, należy wpisać na początku programu:

```
#include "skalib.h"
```

Biblioteka udostępnia następujące funkcje:

- `inicjuj()`
  Ta funkcja udostępnia informację o zawartości skarbca. Zwraca wektor zawierający n liczb całkowitych $a_1, a_2, \ldots, a_n$ ($1 \le n \le 1\,000\,000$, $1 \le a_i \le 10^9$). Liczba a_i oznacza liczbę plików z banknotami w i-tym stosie.
  - **C++:** `vector<int> inicjuj();`
- `waz(p)`
  Ta funkcja wykonuje ważenie. Jej argumentem jest wektor zawierający dokładnie n liczb całkowitych $p_1, p_2, \ldots, p_n$ ($0 \le p_i \le a_i$). Liczba p_i oznacza liczbę plików z banknotami z i-tego stosu, które Bajzek ma położyć na wagę. Funkcja zwraca wagę położonych pieniędzy w gramach.
  - **C++:** `long long waz(const vector<int>& p);`
- `odpowiedz(k)`
  Ta funkcja odpowiada bibliotece, że stos o numerze k ($1 \le k \le n$) zawiera fałszywe pliki banknotów. Jej wywołanie kończy działanie Twojego programu.
  - **C++:** `void odpowiedz(int k);`

Twój program nie może czytać żadnych danych (ani ze standardowego wejścia, ani z plików). Nie może również nic wypisywać do plików ani na standardowe wyjście. Może pisać na standardowe wyjście diagnostyczne (`stderr`), pamiętaj jednak, że zużywa to cenny czas.

Przykładowy przebieg programu

Poniższa tabela przedstawia przykładowy ciąg wywołań funkcji.

Wywołanie funkcji	Wynik	Wyjaśnienie
`inicjuj()`	`[1,2]`	$n = 2$, $a_1 = 1$, $a_2 = 2$
`waz([1,0])`	`101`	$p_1 = 1$, $p_2 = 0$; Bajzek kładzie na wagę jeden plik z pierwszego stosu; jego waga to 101, zatem jest on fałszywy (gdyby stos drugi zawierał fałszywe pliki, waga wyniosłaby 100)
`odpowiedz(1)`	–	udzielamy odpowiedzi; program kończy działanie

Powyższy przebieg programu jest poprawny. Używa on minimalnej liczby wywołań funkcji `waz`, dostanie więc pełne punkty.

Testy „ocen”:

`1ocen:` $n = 3$, wszystkie stosy mają rozmiar 1 poza jednym; wystarczy jedno ważenie;

`2ocen:` $n = 10$, wszystkie stosy mają rozmiar 1; wystarczą cztery ważenia;

`3ocen:` $n = 1000$, stosy mają rozmiar co najwyżej 1000; wystarczą dwa ważenia;

`4ocen:` $n = 1\,000\,000$, test ze stosami rozmiaru co najwyżej 1000; wystarczy dziesięć ważeń.

Ocenianie

Niech W oznacza minimalną liczbę ważeń potrzebnych na znalezienie fałszywego stosu. Jeśli Twój program poprawnie znajdzie stos, używając co najwyżej W wywołań funkcji `waz`, dostanie 100% punktów za test. Jeśli Twój program poprawnie znajdzie stos, wywołując tę funkcję $W+1$ razy, dostanie 50% punktów za test, a jeśli wywoła ją $W+2$ razy, dostanie 25% punktów za test.

Jeśli Twój program nie znajdzie fałszywego stosu lub użyje funkcji `waz` co najmniej $W+3$ razy, dostanie 0 punktów za test.

Zestaw testów dzieli się na następujące podzadania. Testy do każdego podzadania składają się z jednej lub większej liczby osobnych grup testów.

Podzadanie	Warunki	Punkty
1	wszystkie stosy mają rozmiar 1	8
2	wystarczy co najwyżej jedno ważenie ($W \leq 1$)	8
3	$n \leq 1000$; wystarczą co najwyżej dwa ważenia ($W \leq 2$)	28
4	wszystkie stosy mają ten sam rozmiar	12
5	$n \leq 100\,000$	32
6	brak dodatkowych warunków	12

Eksperymenty

Przykładowe **błędne** rozwiązanie wraz z przykładową biblioteką znajdują się w folderze `dlazaw`. Biblioteka wczytuje ze standardowego wejścia dane w następującym formacie:

- pierwszy wiersz: liczba stosów n oraz numer k fałszywego stosu ($1 \leq k \leq n$),
- drugi wiersz: rozmiary stosów $a_1, \ldots, a_n$.

W momencie wywołania funkcji `odpowiedz` biblioteka wypisuje informację, czy program zawodnika poprawnie wykrył fałszywy stos, oraz liczbę wykonanych ważeń. W szczególności przykładowa biblioteka nie sprawdza, czy liczba wykonanych ważeń była minimalna.

Przy kompilacji należy zadbać o to, aby pliki `skalib.h` i `skalib.cpp` znajdowały się w tym samym folderze co rozwiązanie. Polecenie kompilacji wygląda wówczas następująco:

```
g++ -O3 -static skalib.cpp ska.cpp -std=c++17
```

Dostarczony jest też przykładowy plik `makefile` generujący plik wykonywalny `ska.e` z pliku `ska.cpp` za pomocą komendy:

```
make
```