Zadanie: WIE
Wieżowce

POTYCZKI ALGORYTMICZNE

Potyczki Algorytmiczne 2018, runda próbna. Limity: 256 MB, 1 s. *12.01.2019*

Bajteusz jest ekscentrycznym miliarderem, który wykorzystuje swoją wiedzę i majątek, by znajdować odpowiedzi na nietypowe pytania. Niedawno chciał oszacować liczbę wielorybów na świecie (dokładne policzenie było zbyt kosztowne). Postanowił więc wielokrotnie złapać losowego wieloryba, oznaczyć go czerwoną wstążką i wypuścić na wolność. Bajteusz powtarzał tę operację do momentu, gdy złapany wieloryb był już oznaczony wstążką, czyli pojawiło się powtórzenie. Jeśli było k losowań, to rząd wielkości liczby wszystkich osobników to k^2. Metoda ta ma pewien problem, ale nie tym się dzisiaj zajmiemy.

Stolica Bajtocji, Bitowice, słynie ze swoich potężnych wieżowców, ponumerowanych od 1 do n. Jeśli wieżowiec ma x pięter, to są one ponumerowane od 1 do x. Problem w tym, że nikt nie zna dokładnej liczby pięter poszczególnych wieżowców. Nikt poza agentami nieruchomości, rzecz jasna.

Bajteusz postanowił poznać wysokość najwyższego wieżowca w Bitowicach, tzn. największą liczbę pięter. Żaden agent nieruchomości nie chce tego zdradzić, ale oczywiście każdy z chęcią sprzeda mieszkanie na dowolnym piętrze dowolnego wieżowca – lub powie, że takie piętro nie istnieje! Bajteusz może wybrać numer wieżowca i i numer piętra j, po czym zamówić mieszkanie na j-tym piętrze i-tego wieżowca. Dostanie wtedy odpowiedź `true` lub `false`, mówiącą, czy zakup się udał. Oszczędności Bajteusza pozwolą na 303030 zamówień (te z odpowiedzią `false` też się liczą, bo płacić trzeba z góry, a zwrot pieniędzy nastąpi dopiero za 30 dni).

Twoim zadaniem jest pomóc Bajteuszowi i znaleźć wysokość najwyższego wieżowca.

Interakcja

Twój program nie powinien nic czytać z wejścia ani wypisywać na wyjście. Zamiast tego zostanie on skompilowany z biblioteką oceniającą. Na początku programu umieść dyrektywę:

```
#include "wie.h"
```

Biblioteka ta udostępnia następujące funkcje:

- `int wezN()` – Funkcja ta zwraca liczbę n, oznaczającą liczbę wieżowców w Bitowicach.
- `bool zamowienie(int` i`, long long` j`)` – Funkcja ta zwraca wartość logiczną `true`, jeśli i-ty wieżowiec ma co najmniej j pięter, a `false` w przeciwnym przypadku. Dozwolone jest wielokrotne pytanie o to samo piętro tego samego wieżowca. Argumenty muszą spełniać następujące nierówności: $1 \leq i \leq n$ oraz $1 \leq j \leq 10^{18}$.
- `void odpowiedz(long long` X`)` – Użyj tej metody do udzielenia odpowiedzi, podając największą liczbę pięter X. Możliwe są remisy, więc być może wiele wieżowców będzie miało taką wysokość. Po wykonaniu tej funkcji Twój program zostanie zakończony funkcją `exit(0)`.

Biblioteka ta **nie** jest adaptacyjna. Oznacza to, że wysokości wieżowców są ustalone na początku uruchomienia programu i nie zmieniają się w zależności od pytań Twojego programu.

Ograniczenia

- $1 \leq n \leq 300\,000$,
- wysokości wieżowców są liczbami całkowitymi z przedziału $[1, 10^{18}]$,
- możesz użyć funkcji `zamowienie` co najwyżej 303030 razy.

Przykładowa biblioteka

Do Twojej dyspozycji w dziale Pliki w SIO jest dostępna paczka z przydatnymi plikami, zawierająca przykładową bibliotekę `wiezaw.cpp`, która implementuje podane wyżej trzy funkcje.

Aby skompilować Twój program (powiedzmy `wie.cpp`) z tą biblioteką, wpisz następującą komendę:

```
g++ wie.cpp wiezaw.cpp -o wie -std=c++11 -O2
```

Wejście dla przykładowej biblioteki

Biblioteka przyjmuje wejście w następującym formacie.

W pierwszym wierszu powinna znaleźć się liczba wieżowców n. W drugim wierszu powinno znaleźć się n liczb – i-ta z nich powinna określać wysokość i-tego wieżowca.

Przykład

Załóżmy, że mamy cztery wieżowce o kolejnych wysokościach: 2, 1, 7 oraz 3.

wywołanie	wynik
`wezN()`	4
`zamowienie(3, 2)`	`true`
`zamowienie(2, 1)`	`true`
`zamowienie(4, 4)`	`false`
`zamowienie(3, 5)`	`true`
`odpowiedz(7)`	

Takiemu testowi odpowiadałby następujący plik wejściowy dla biblioteki przykładowej (dostępny jako plik `wie0.in`):

```
4
2 1 7 3
```