Zadanie: SON
Sondy i rebelianci

Olimpiada Informatyczna

ONTAK 2017, dzień ósmy. Plik źródłowy `son.*` Dostępna pamięć: 256 MB. *5.7.2017*

Darth Vader przydzielił Ci zadanie odnalezienia bazy rebeliantów ukrytej w pasie asteroid. Wiesz już, że należy jej szukać na jednej spośród n planetek leżących wzdłuż pasa, którym nadałeś numery 1, 2, ..., n ($n \leq 10^{12}$).

Do dyspozycji masz specjalnie skonstruowane, naszpikowane nowoczesną technologią sondy, które na milę wyczują rebeliancką krew. Niestety, przez swoją nowoczesność sondy są naprawdę duże i w dodatku jasno świecą. Rebelianci zatem po prostu zestrzelą każdą sondę, która doleci do ich planety... mimo to, postanowiłeś spróbować. W jednej misji wysyłasz sondę wzdłuż pasa w kierunku pewnej planety p. Sonda mija kolejno planety 1, 2,... aż do p. Jeśli na żadnej z nich nie ma bazy, sonda wraca i informuje Cię o tym. Jeśli na którejś z mijanych planet czają się rebelianci, sonda zostaje zestrzelona i permanentnie stracona.

Masz do dyspozycji $k \leq 500$ sond, a Twój czas pozwala Ci na wykonanie co najwyżej $q \leq 500$ misji. Potem Lord Vader straci cierpliwość, a Ty nie dożyjesz zawodów drużynowych ONTAKa. Ustal strategię, która pozwoli Ci na jednoznaczne wyznaczenie bazy rebeliantów przy podanych ograniczeniach.

Gwarantujemy, że dla danych parametrów wejściowych istnieje strategia, dla której na pewno znajdziemy bazę rebeliantów.

Komunikacja

To jest zadanie interaktywne. Twój program, zamiast czytać z wejścia dane i pisać odpowiedź na wyjście, powinien komunikować się z dostarczoną biblioteką. Za jej pomocą możesz zarządzić wysłanie sondy, dowiedzieć się o wyniku misji, oraz podać ostateczną odpowiedź. Aby użyć biblioteki, należy wpisać na początku programu:

- **C/C++**: `#include "csonlib.h"`

Biblioteka udostępnia następujące funkcje i procedury:

- `dajDlugosc()`, `dajLiczbeSond()`, `dajLimitCzasowy()`;
  Pierwsza funkcja daje w wyniku liczbę całkowitą n, oznaczającą długość pasa asteroid (czyli liczbę możliwych planet),
  druga liczbę całkowitą k, oznaczającą liczbę sond, które masz do dypozycji, zaś
  trzecia liczbę całkowitą q, oznaczającą liczbę misji, które można przeprowadzić.
  - **C/C++:**
    ```
    long long dajDlugosc();
    int dajLiczbeSond();
    int dajLimitCzasowy();
    ```

- `sprawdz(k)`;
  Funkcja przeprowadza misję i w przypadku jej niepowodzenia zmniejsza dostępną liczbę sond o 1. Funkcja daje w wyniku `true`, jeśli sonda została zestrzelona, `false` w przeciwnym wypadku.
  - **C/C++:** `bool sprawdz(long long k);`
- `odpowiedz(wynik)`;
  Funkcja podaje bibliotece liczbę *wynik* jako odpowiedź – położenie bazy rebeliantów. Wywołanie tej funkcji **kończy działanie Twojego programu**.
  - **C/C++:** `void odpowiedz(long long wynik);`

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.

Podzadania

zadanie	punkty	ograniczenia
1	7	$n = q$, $k = 1$
2	33	$k = q = 60$
3	40	$k = 2$, $n \leq 60000$, $q = 500$
4	20	*brak*

Eksperymenty

W zakładce *Pliki* dostępna jest przykładowa biblioteka, która pozwoli Ci przetestować poprawność formalną rozwiązania. Biblioteka zakłada, że $q = 100$, $k = 2$ oraz $n = 100$, a baza znajduje się na planecie 50. Po wywołaniu procedury `odpowiedz`, biblioteka wypisuje na standardowe wyjście informację o udzielonej odpowiedzi.

Dostępne jest również przykładowe rozwiązanie `son.cpp` korzystające z biblioteki. Rozwiązanie to nie jest poprawne i zawsze wykonuje operację `odpowiedz(1)`.

Do kompilacji rozwiązania wraz z biblioteką powinieneś użyć polecenia:

- **C++**: `g++ csonlib.cpp Twoj_plik.cpp -oTwoj_plik_wynikowy --std=c++11`

Plik z rozwiązaniem i biblioteka powinny znajdować się w tym samym katalogu.