Zadanie: DZI
Dzielniki [B]

POTYCZKI ALGORYTMICZNE

Potyczki Algorytmiczne 2024, runda piąta. Limity: 1024 MB, 15 s. *15.03.2024*

Jury wybrało pewną liczbę całkowitą x z przedziału $[1, n]$. Twoim zadaniem jest ją odgadnąć. Abyś nie musiał/musiała robić tego kompletnie w ciemno, możesz zadawać zapytania. W każdym zapytaniu możesz podać liczbą całkowitą y z przedziału $[0, c]$, a my w odpowiedzi zdradzimy Ci liczbę dzielników liczby $x + y$.

Aby nieco utrudnić Ci zadanie, w trakcie pojedynczego uruchomienia programu będziesz musiał/musiała rozwiązać t przypadków testowych. Sumaryczna liczba zapytań, które możesz w nich zadać, jest ograniczona przez q.

Komunikacja

To zadanie jest interaktywne. Należy napisać program, który będzie komunikował się z Jury, używając do tego dostarczonej biblioteki. Aby użyć biblioteki, należy wpisać w swoim programie:

- C++: `#include "dzilib.h"`
- Python: `from dzilib import GetT, GetN, GetQ, GetC, Ask, Answer`

Biblioteka udostępnia następujące funkcje:

- `GetT()` – Zwraca parametr t – liczbę przypadków testowych do rozwiązania.
- `GetN()` – Zwraca parametr n – ograniczenie na ukryte wartości x.
- `GetQ()` – Zwraca parametr q – ograniczenie na sumaryczną liczbę zapytań, które możesz zadać we wszystkich t przypadkach testowych.
- `GetC()` – Zwraca parametr c – ograniczenie na podawane przez Ciebie wartości y.
- `Ask(y)` – Zwraca liczbę dodatnich dzielników ukrytej liczby x powiększonej o y. Musi zachodzić $0 \leq y \leq c$.
- `Answer(z)` – Informuje bibliotekę, że Twoim zdaniem ukryta wartość x wynosi z. Nie zwraca nic (w języku C++ funkcja jest typu `void`).

W języku Python wszystkie parametry funkcji bibliotecznych oraz zwracane przez nie wartości są typu całkowitoliczbowego. W języku C++ typy przyjmowane oraz zwracane przez funkcje są takie same jak w przykładowej, dostarczonej bibliotece, o której więcej w sekcji *Eksperymenty*.

W każdym momencie działania programu (poza samym jego końcem) aktywny będzie jeden przypadek testowy. Pierwszy przypadek testowy zostaje aktywowany od razu po rozpoczęciu działania programu. Gdy przypadek testowy jest aktywny, możesz zadawać zapytania używając funkcji `Ask`. Gdy stwierdzisz, że znasz ukrytą wartość x, możesz podać ją za pomocą funkcji `Answer`, a biblioteka ją zweryfikuje. Jeśli podana wartość jest niepoprawna, to biblioteka zakończy działanie programu z werdyktem „błędna odpowiedź”. Jeśli wartość parametru będzie poprawna, to funkcja zakończy się; jeśli rozwiązany przypadek testowy nie był ostatni, to od razu aktywuje się kolejny. Po udzieleniu odpowiedzi na ostatni przypadek testowy powinnaś/powinieneś zakończyć działanie programu. Jeśli tego nie zrobisz i spróbujesz użyć funkcji `Ask` lub `Answer`, otrzymasz werdykt „błędna odpowiedź”.

Funkcji `GetT`, `GetN`, `GetQ` oraz `GetC` możesz używać w dowolnym momencie działania programu i zwracane przez nie wartości nie ulegną zmianie. Funkcje te nie liczą się do limitu zapytań, ale zużywają czas procesora. Wartość q ogranicza jedynie liczbę wywołań funkcji `Ask`. W momencie, gdy przekroczysz łączną dozwoloną liczbę zapytań, biblioteka zakończy działanie programu i otrzymasz werdykt „błędna odpowiedź”.

Nie należy wczytywać żadnych danych ze standardowego wejścia ani nic wypisywać na standardowe wyjście. Celowe próby wpłynięcia na wewnętrzne działanie biblioteki oceniającej są zakazane.

Biblioteka

Ukryte wartości x we wszystkich testach oraz ich kolejność są ustalone z góry. Oznacza to, że biblioteka, z którą komunikuje się Twój program, **nie będzie złośliwa** i nie będzie dostosowywać swojego zachowania do działania Twojego programu.

Limity podane w treści zadania dotyczą jedynie czasu i pamięci, które zużyje Twój program. Czas działania biblioteki oraz zużyta przez nią pamięć mogą jednak zależeć od testu oraz od dokładnego zachowania Twojego programu. Z tego względu limity czasu i pamięci w SIO_2 są nieco większe niż te podane w treści.

Przykładowy przebieg programu

W teście przykładowym zachodzi $t = 2$, $n = 10^6$, $q = 10^4$, $c = 10^{15}$, ukrytymi wartościami są kolejno 1000 i 1, a przykładowa komunikacja z biblioteką może wyglądać następująco:

Wywołana funkcja	**Wynik**	**Opis**
`GetT()`	2	$t = 2$.
`GetN()`	1 000 000	$n = 10^6$.
`GetQ()`	10 000	$q = 10^4$.
`GetC()`	1 000 000 000 000 000	$c = 10^{15}$.
`Ask(1)`	8	Liczba $x + 1$ ma 8 dzielników: 1, 7, 11, 13, 77, 91, 143 i 1001.
`Ask(24)`	11	Liczba $x + 24$ ma 11 dzielników: 1, 2, 4, 8, 16, 32, 64, 128, 256, 512 i 1024.
`Answer(1000)`	–	Poprawna odpowiedź, rozpoczęty zostaje kolejny przypadek testowy.
`GetT()`	2	Dozwolone zapytanie – zwrócona wartość nie zmienia się.
`Answer(1)`	–	Poprawny przykład komunikacji i podanie poprawnej odpowiedzi na ostatni przypadek testowy. Po udzieleniu odpowiedzi należy zakończyć działanie programu.

Zadane zostały 2 zapytania `Ask`, co mieści się w limicie 10 000 zapytań. Pamiętaj, że liczy się sumaryczna liczba zapytań dla wszystkich przypadków testowych w jednym teście.

Ocenianie

W obrębie grupy testów wszystkie testy mają te same wartości t, n, q oraz c, które prezentuje poniższa tabela:

Numer grupy	**t**	**n**	**q**	**c**
1	50	10^5	50 000	10^{12}
2	50	10^6	5 000	10^{12}
3	10	10^9	50 000	10^{12}
4	10	10^{14}	5 000	10^{17}
5	10	10^{14}	2 000	10^{17}
6	10	10^{14}	1 300	10^{17}
7	10	10^{14}	950	10^{17}
8	10	10^{14}	820	10^{17}
9	10	10^{14}	750	10^{17}
10	10	10^{14}	720	10^{17}

Test przykładowy nie należy do żadnej z grup. Możesz go rozwiązać, jednak zwróć uwagę, że możliwym jest zdobycie kompletu punktów bez rozwiązywania go.

Eksperymenty

Przykładowe **błędne** rozwiązania i przykładowe biblioteki w językach C++ i Python znajdują się w archiwach w dziale *Pliki* na SIO_2. Rozwiązania te i biblioteki obrazują typy zwracane oraz przyjmowane przez wszystkie funkcje. Przykładowe biblioteki mogą różnić się zachowaniem od tej, która zostanie użyta do oceny rozwiązań, i mogą nie spełniać założeń zadania. Mają one jedynie pokazać sposób interakcji z programem.

Rozwiązanie `dzi.cpp` w języku C++ można skompilować w następujący sposób:

```
g++ -O3 -static -o dzi dzi.cpp dzilib.cpp -std=c++20
```

Należy zadbać o to, aby pliki `dzilib.h` oraz `dzilib.cpp` znajdowały się w tym samym folderze, co rozwiązanie.

Rozwiązanie `dzi.py` można uruchomić poleceniem

```
python dzi.py
```

Należy zadbać o to, aby plik `dzilib.py` znajdował się w tym samym folderze, co rozwiązanie.

Po uruchomieniu, na samym początku działania programu biblioteka przyjmuje na standardowym wejściu kolejno wartości t, n, q oraz c, a następnie kolejne ukryte wartości x. Plik wejściowy odpowiadający testowi przykładowemu znajduje się w obu archiwach pod nazwą `dzi0.in`.