Zadanie: GDZ
Gdzie jest jedynka? 2

POTYCZKI ALGORYTMICZNE

Potyczki Algorytmiczne 2019, finał. Limity: 256 MB, 3 s. *19.01.2020*

Bajtazar wymyślił sobie pewną permutację $(P_0, P_1, \ldots P_{n-1})$ liczb od 0 do $n-1$. Bajtazar prosi Cię o zgadnięcie, na którym miejscu tej permutacji znajduje się liczba 1. Abyś nie musiał(a) zgadywać w ciemno, Bajtazar będzie udzielać Ci odpowiedzi na pytania postaci: „Jaki jest największy wspólny dzielnik dwóch **różnych** liczb P_i oraz P_j?"

Aby nie było zbyt prosto, możesz zadać mu co najwyżej $\lceil \frac{5n}{2} \rceil$ zapytań. Dodatkowo, Bajtazar wciąż wymyśla sobie nowe permutacje, więc być może przyjdzie Ci szukać jedynki wiele razy.

Komunikacja

To zadanie jest interaktywne. Należy napisać program, który będzie komunikował się z Bajtazarem, używając do tego dostarczonej biblioteki. Należy w tym celu dołączyć nagłówek `gdzlib.h` za pomocą dyrektywy

```
#include "gdzlib.h"
```

Biblioteka udostępnia następujące funkcje:

- `int GetN()` – Zwraca parametr n ($3 \leq n \leq 500\,000$ lub $n = -1$). Jeśli $n \geq 3$, oznacza on długość aktualnie rozważanej permutacji. Jeśli $n = -1$, oznacza to, że Bajtazarowi skończyły się permutacje.
- `int Ask(int i, int j)` – Zwraca największy wspólny dzielnik liczb P_i oraz P_j ($0 \leq i, j < n, i \neq j$). W jednym przypadku testowym tę funkcję można wywołać co najwyżej $\lceil \frac{5n}{2} \rceil$ razy.
- `void Answer(int x)` – Udziela odpowiedzi, że $P_x = 1$ ($0 \leq x < n$).

Każde uruchomienie Twojego programu składa się z co najmniej jednego przypadku testowego. Każdy przypadek należy rozpocząć funkcją `GetN()` i zakończyć funkcją `Answer()`. Używanie funkcji `Ask()` lub `Answer()`, gdy żaden przypadek testowy nie jest aktywny, skutkować będzie werdyktem „błędna odpowiedź". Po przetworzeniu ostatniego przypadku testowego funkcja `GetN()` zwróci -1. Należy wtedy zakończyć działanie swojego programu.

Przypadki testowe w trakcie jednego uruchomienia programu mogą mieć różne parametry n. Suma długości permutacji w trakcie działania programu nie przekroczy jednak 500 000.

Użycie którejkolwiek funkcji z niepoprawnymi argumentami będzie skutkować otrzymaniem werdyktu „błędna odpowiedź". Celowe próby wpłynięcia na wewnętrzne działanie biblioteki oceniającej są zakazane.

Biblioteka

Biblioteka, z którą komunikuje się Twój program, może być *złośliwa*. Oznacza to, że biblioteka może dostosowywać ukryte permutacje w zależności od Twoich zapytań.

Czas działania biblioteki oraz zużyta przez nią pamięć mogą zależeć od testu oraz od dokładnego zachowania Twojego programu. Z tego względu limity na SIO są nieco większe niż te podane w treści. Jeśli biblioteka sprawdzająca zadziała szybciej lub zużyje mniej pamięci niż przewiduje przeznaczony dla niej zapas, Twój program będzie wtedy nieco mniej ograniczony.

Przykładowy przebieg programu

W teście przykładowym jest tylko jeden przypadek testowy. Wartość n jest równa 5, natomiast ukryta permutacja P to $(4, 2, 0, 3, 1)$.

Wywołana funkcja	Wynik	Opis
`GetN()`	5	$n = 5$.
`Ask(1, 4)`	1	$\mathrm{NWD}(P_1, P_4) = \mathrm{NWD}(2, 1) = 1$.
`Ask(1, 0)`	2	$\mathrm{NWD}(P_1, P_0) = \mathrm{NWD}(2, 4) = 2$.
`Ask(2, 3)`	3	$\mathrm{NWD}(P_2, P_3) = \mathrm{NWD}(0, 3) = 3$.
`Answer(4)`	—	Odpowiadamy, że $P_4 = 1$.
`GetN()`	-1	Należy zakończyć działanie programu.

Bajtazar zadał 3 zapytania `Ask()`, co mieści się w limicie $\lceil \frac{5n}{2} \rceil = 13$ zapytań. Pamiętaj, że zapytania są liczone oddzielnie dla każdego przypadku testowego.

Eksperymenty

Przykładowe **błędne** rozwiązanie i przykładowa biblioteka znajdują się w archiwum w dziale *Pliki* na SIO. Biblioteka może różnić się zachowaniem od tej, która zostanie użyta do oceny rozwiązań i nie spełniać założeń zadania. Ma ona jedynie pokazać sposób interakcji z programem.

Rozwiązanie `gdz.cpp` można skompilować w następujący sposób:

```
g++ -O3 -static -o gdz gdz.cpp gdzlib.cpp -std=c++17
```

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

Po skompilowaniu, biblioteka przyjmuje na standardowym wejściu opisy kolejnych przypadków testowych. Opis każdego testu powinien składań się z dwóch wierszy. Pierwszy z nich powinien zawierać pojedynczą liczbę n. Drugi z nich powinien zawierać permutację liczb od 0 do $n-1$. Po opisie wszystkich przypadków testowych powinien znaleźć się wiersz zawierający liczbę -1. Plik wejściowy odpowiadający testowi przykładowemu znajduje się w archiwum w pliku `gdz0.in`.