

Zadanie: ZER

Zera i jedyнки



Olimpiada Informatyczna, dzień próbny. Plik źródłowy zer.* Dostępna pamięć: 256 MB.

Jest sobie pewien ciąg $a_0, \dots, a_{n-1}$ złożony z liczb 0 i 1. Nie znasz elementów tego ciągu, możesz jednak zapytać o sumę dowolnej pary różnych elementów ciągu. Twoim zadaniem jest odgadnąć ciąg za pomocą małej liczby takich pytań.

Komunikacja

Zaimplementuj program, który odgadnie ciąg zer i jedynek, korzystając z dostarczonej biblioteki odpowiadającej na podane pytania. Twój program nie może używać standardowego wejścia i wyjścia.

Uwaga: Podane ograniczenia pamięci i czasu dotyczą tylko Twojego rozwiązania (nie wliczają działania biblioteki).

C++

Na początku programu należy napisać:

- C++: `#include "zerlib.h"`

Biblioteka udostępnia następujące funkcje:

- `int daj_n()` – Funkcja daje w wyniku liczbę całkowitą n , oznaczającą długość nieznanego ciągu zer i jedynek. Funkcję tę można wywołać dowolnie wiele razy.
- `int suma(int i, int j)` – Funkcja daje w wyniku $a_i + a_j$, jeśli $0 \leq i, j < n$ oraz $i \neq j$. W przeciwnym razie jej wywołanie spowoduje błędną odpowiedź.
- `void odpowiedz(std::vector<int> a)` – Ta funkcja pozwala zgłosić wynikowy ciąg. Należy ją wykonać dokładnie raz. Przyjmuje ona tablicę $a[0..n-1]$ daną jako wektor `std::vector<int>` reprezentującą ciąg wynikowy $a[0], \dots, a[n-1]$. Jeśli podany w funkcji ciąg będzie błędny lub jego długość będzie inna niż n , spowoduje to błędną odpowiedź. **Po wykonaniu tej funkcji należy zakończyć program.**

Python

Na początku programu należy napisać:

- Python: `from zerlib import daj_n, suma, odpowiedz`

Biblioteka udostępnia następujące funkcje:

- `daj_n() -> int` – Funkcja daje w wyniku liczbę całkowitą n , oznaczającą długość nieznanego ciągu zer i jedynek. Funkcję tę można wywołać dowolnie wiele razy.
- `suma(i : int, j : int) -> int` – Funkcja daje w wyniku $a_i + a_j$, jeśli $0 \leq i, j < n$ oraz $i \neq j$. W przeciwnym razie jej wywołanie spowoduje błędną odpowiedź.
- `odpowiedz(a : list[int])` – Ta funkcja pozwala zgłosić wynikowy ciąg. Należy ją wykonać dokładnie raz. Przyjmuje ona tablicę $a[0..n-1]$ daną jako listę, reprezentującą ciąg wynikowy $a[0], \dots, a[n-1]$. Jeśli podany w funkcji ciąg będzie błędny lub jego długość będzie inna niż n , spowoduje to błędną odpowiedź. **Po wykonaniu tej funkcji należy zakończyć program.**

Ocenianie

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

Podzadanie	Warunki	Liczba punktów
1	$3 \leq n \leq 1000$	50
2	$3 \leq n \leq 200\,000$	50

Biblioteka **nie musi** generować ciągu $a_0, \dots, a_{n-1}$ na początku interakcji z Twoim programem. Może ona w trakcie interakcji zmieniać elementy ciągu, o ile nowe elementy są nadal spójne z wynikami zwróconymi przez dotychczasowe wywołania funkcji `suma`.

Jeśli m to maksymalna liczba wywołań funkcji `suma`, które Twój program wykonał w jednym przypadku testowym, to Twoje rozwiązanie otrzyma następujący procent punktów za test (odpowiednio przeskalowany w przypadku przekroczenia połowy limitu czasowego):

Liczba wywołań	Procent punktów
$m \leq n$	100% punktów za test
$m = n + 1$	80% punktów za test
$m \leq n^2 - n$	50% punktów za test
$m > n^2 - n$	0% punktów za test (werdykt <i>Błędna odpowiedź</i>)

Przykładowy przebieg programu

Poniżej przedstawiono przykładowy przebieg programu dla testu przykładowego.

Akcja	Parametry	Wynik	Opis
daj n	–	5	$n = 5$
suma	$i = 0, j = 1$	1	$a_0 + a_1 = 1$
suma	$i = 1, j = 2$	1	$a_1 + a_2 = 1$
suma	$i = 3, j = 4$	2	$a_3 + a_4 = 2$, a więc $a_3 = a_4 = 1$
suma	$i = 0, j = 3$	2	$a_0 + a_3 = 2$, a więc $a_0 = 1$, skąd wynika też, że $a_1 = 0$ i $a_2 = 1$
odpowiedź	$a = 1, 0, 1, 1, 1$	–	Prawidłowa odpowiedź z użyciem $m = 4 \leq n = 5$ pytań, 100% punktów za test

Testy przykładowe. Test **0a** to test z przykładu powyżej. Poza tym:

0b: $n = 1000$, $a = \underbrace{00\dots0}_{500} \underbrace{11\dots1}_{500}$;

0c: $n = 200\,000$, $a = \underbrace{(01)(01)\dots(01)}_{100\,000}$.

Dla zawodnika

W katalogu `dlazaw` znajdują się przykładowe (błędne) rozwiązania w C++ oraz Pythonie, w plikach `zer.cpp` i `zer.py`. Są też dostępne testy przykładowe (`zer0a.in`, `zer0b.in`, `zer0c.in`) oraz przykładowe biblioteki do komunikacji. Testy przykładowe są w następującym formacie:

- w pierwszym wierszu: liczba całkowita n ,
- w drugim wierszu: ciąg liczb $a_0, \dots, a_{n-1}$, rozdzielonych odstępami.

Przykładowe biblioteki są inne od tych używanych w SIO. W szczególności ciąg $a_0, \dots, a_{n-1}$ jest ustalony na początku interakcji z Twoim programem. Ponadto biblioteki te mogą nie sprawdzać poprawności wejścia ani argumentów wywołania funkcji. Gdy wyślesz rozwiązanie do SIO, zostanie ono sprawdzone na testach przykładowych za pomocą właściwej biblioteki.

Aby skompilować przykładowe rozwiązanie w C++, możesz użyć następującego polecenia, które tworzy plik `zer.e`:

- `g++ -O3 -static -std=c++23 zerlib.cpp zer.cpp -o zer.e`

Rozwiązanie można uruchomić np. na teście przykładowym `zer0a.in` za pomocą polecenia:

- C++: `./zer.e <zer0a.in`
- Python: `python3 zer.py <zer0a.in`

Uwaga: W tym zadaniu nie są dostępne uruchomienia próbne w SIO ani skrypt `ocen` na komputerach.