Triv

PREOI 2025
Dzień 4 – 28 stycznia 2025

Kod zadania: **tri**
Limit pamięci: **512 MiB**

Ostrzegałem.

Pan A. i panna H. mają swoje zbiory liczb całkowitych. Każdy zbiór jest reprezentowany jako ciąg binary długości $M = 2^{21}$, w którym na i-tej pozycji jest 1, jeśli i-ty element występuje w zbiorze (oczywiście jak na prawdziwych informatyków przystało indeksują swoje zbiory od 0).

Planują oni n razy wyrzucić najmniejszy element, który razem posiadają. Jeśli zarówno pan A. jak i panna H. posiadają taki sam najmniejszy element, to elementu pozbywa się pan A. Zastanawiają się teraz, jaka będzie n-ta liczba, którą wyrzucą.

Ponieważ **bardzo** cenią sobie ciszę, chcą rozwiązać ten problem komunikując się przy tym jak najmniej razy.

Implementacja

Twój program będzie uruchamiany w dwóch procesach i za każdym razem będzie wywoływany z inną funkcją:

Uruchomienie dla pana A.:

- `void mainA(std::bitset<M>, int n, int subtask)`

Uruchomienie dla panny H.:

- `void mainH(std::bitset<M>, int n, int subtask)`

Obydwie funkcje mogą wywoływać następujące funkcje:

- `void send(bool)` – ta funkcja służy do przesyłania bitów drugiej osobie.
- `bool receive()` – ta funkcja służy do czytania bitów od drugiej osoby.
- `void answer(int)` – ta funkcja służy do zwrócenia wyniku. Może ją wywołać dowolny proces, po jej wywołaniu obydwa procesy zostają zatrzymane (w szczególności, drugi proces nie musi wiedzieć, że pierwszy już odpowiada).

Program **musi** zaczynać się od linijki `#include "trilib.h"`

Uwaga. Zmienna subtask przyjmie wartość 0,1 lub 2 w zależności od podzadania (0 dla testu przykładowego)

Ważne uwagi

Program może używać zmiennych globalnych, ale ponieważ jest on uruchamiany oddzielnie dla panny H. i pana A., uruchomienia te nie będą współdzielić zmiennych globalnych. Nie należy samodzielnie zakończyć wywołania programu, jedyne poprawne zakończenia to: wywołanie funkcji answer oraz koniec funkcji mainA/H.

Dla uczestników korzystających z systemu operacyjnego Linux (lub WSL itp.) - do czego bardzo gorąco zachęcamy - w zakładce pliki został umieszczony plik `dlazaw.zip` (w kategorii (tri)) znajduje się tam parę plików ale należy modyfikować jedynie plik `tri.cpp` oraz korzystać z pliku `compile.sh` (poniżej wyjaśnienie).

Dla uczestników nie mających dostępu do Linux'a udostępniamy środowisko w przeglądarce. Aby z niego skorzystać, należy wejść w link, założyć konto (jeśli takowego się jeszcze nie ma) i w prawym górnym rogu kliknąć `Use this template -> Open in codespaces`.

Do dyspozycji jest skrypt `compile.sh`, aby z niego skorzystać należy wpisać komendę w terminal: `./compile.sh 0` albo `./compile.sh 1` – odpowiednio bez/z pomocniczymi informacjami. Skrypt ten stworzy plik `tri.e`, który następnie należy odpalić (komendą `./tri.e`) i wpisać test.

Testy mają następującą strukturę:
`PREFIX SEED`

gdzie `PREFIX` to długość prefiksu na którym pojawią się niezerowe wartości w bitsetach, a `SEED` to ziarno do generatora liczb losowych.

Wyjście programu będzie dwojakie, albo zacznie się od `ERROR` po czym zostanie wskazany konkretny błąd (a w kolejnej linii wyjaśnienie błędu), albo zacznie się od `ANSWER` po czym nastąpi odpowiedź zwrócona przez program, następnie poprawna odpowiedź, a następnie liczba wykorzystanych bitów.

Przykłady

Wejście dla testu `tri0a`:

```
16 410907
```

Ograniczenia

Pan A. oraz panna H. mogą się porozumieć z pomocą maksymalnie 2^{11} bitów.

$n > 0$ oraz n-ty element zawsze istnieje.

Ocenianie

Podzadanie	Ograniczenia	Limit czasu	Liczba punktów
1	państwo A. i H. mają tylko elementy mniejsze niż 2^{11}	1 s	11
2	brak dodatkowych ograniczeń	1 s	89

W drugim podzadaniu liczba punktów jest zależna od maksymalnej ilości wysłanych bitów. Oznaczmy tą liczbę jako L.

- Jeśli $L \leq 2^{11}$ twój program dostanie za to podzadanie 18 punktów.
- Dla każdego i ze zbioru $2, 3, 4, 5, 6, 7, 8, 9, 10$ jeśli $L \leq 100 \cdot i$, twój program otrzyma dodatkowe 3 punkty.
- Dla każdego i ze zbioru $9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19$ jeśli $L \leq 10 \cdot i$, twój program otrzyma dodatkowe 4 punkty.