Zadanie: KOD
Kodowanie macierzami [B]

POTYCZKI ALGORYTMICZNE

Potyczki Algorytmiczne 2026, runda piąta. Limity: 1024 MB, 10 s. *2026-03-27*

Algosia i Bajtek są bardzo zajęci. Nie mają czasu na wymyślanie oryginalnych zadań, a już na pewno nie na wysyłanie sobie macierzy rozmiaru 1000×1000, co musieli robić, gdy brali udział w tegorocznym finale Olimpiady Informatycznej*.

Algosia chciałaby przekazać Bajtkowi pewną dodatnią liczbę całkowitą. Niestety, jak to ostatnio u nich bywa, oddziela ich bardzo zaawansowany system komputerowy. Algosia może wpisać do komputera macierz binarną rozmiaru 10×10, następnie system permutuje wiersze oraz kolumny macierzy i wysyła je Bajtkowi.

Napisz program, który pomoże Algosi i Bajtkowi t razy zakodować i odkodować liczbę.

Interakcja

To jest zadanie interaktywne. Twój program zostanie uruchomiony w dwóch kopiach – jednej dla Algosi, a drugiej dla Bajtka. Każde z uruchomień w pierwszym wierszu wejścia otrzyma słowo `Algosia` lub `Bajtek`, które określa za działanie której z osób odpowiada ta kopia programu.

Obydwa programy od razu po otrzymaniu swojego słowa dostaną też na wejściu dwie liczby całkowite n i t ($1 \le n \le 3 \cdot 10^{16}, 1 \le t \le 25$), oznaczające kolejno ograniczenie górne na kodowane liczby oraz liczbę przypadków testowych. Następnie t razy odbywa się interakcja.

Na początku i-tej interakcji Algosia otrzymuje jedną liczbę całkowitą n_i ($1 \le n_i \le n$). Na wyjściu powinna ona wypisać 10 wierszy zawierających po 10 znaków każdy. Każdy ze znaków powinien być równy `0` lub `1`. Wiersze i kolumny tej macierzy zostaną przepermutowane i wysłane na wejście procesu Bajtka w tym samym formacie. Po otrzymaniu macierzy proces Bajtka powinien na wyjście wypisać zakodowaną przez Algosię liczbę n_i. Po wypisaniu tej liczby zaczyna się kolejny przypadek testowy.

Po wypisaniu każdego fragmentu należy opróżnić bufor wyjścia, na przykład poprzez wywołanie polecenia `cout.flush()` lub `fflush(stdout)`.

Uwagi

- Opisana poniżej interakcja odpowiada pierwszemu testowi przykładowemu o nazwie `kod0a.in`.
- Obydwa programy rozpoczynają się w tym samym czasie. Czas działania programu mierzony jest jako czas rzeczywisty od rozpoczęcia do zakończenia interaktora.
- Algosia dostaje kolejną liczbę dopiero po odkodowaniu poprzedniej przez Bajtka.
- Proces Bajtka po wczytaniu liczb n i t zamiast blokować się na wejściu może wykorzystać czas, w trakcie którego proces Algosi koduje pierwszą liczbę na dowolne obliczenia. W skrajnych przypadkach może to dwukrotnie przyspieszyć działanie programu.

Przykładowa interakcja

Interaktor do Algosi	Algosia do interaktora	Interaktor do Bajtka	Bajtek do interaktora	Wyjaśnienie
`Algosia`		`Bajtek`		Interaktor informuje kopie programów Algosi i Bajtka o ich tożsamości.
`20 2`		`20 2`		Oboje się dowiadują, że będą dwa przypadki testowe i w każdym z nich maksymalna liczba do przekazania nie przekroczy 20.
`12`				Algosia dostaje do zakodowania liczbę 12.

*Mowa tu o zadaniu *Steganografia*, którego treść można znaleźć tutaj:
`https://szkopul.edu.pl/problemset/problem/Vc3EYISNd8DRaR6CC1g_4Dhy/site`,
a omówienie jego rozwiązania tutaj: `https://oi.edu.pl/l/oi33_3_ste`.
Nie gwarantujemy jednak, że informacje w nim zawarte są jakkolwiek przydatne przy rozwiązywaniu tego zadania.

	1110000000 1000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000 0000000000			Algosia wysyła macierz kodującą liczbę 12.
		0000000000 0000100000 0000000000 0000000000 0000000000 0000000000 0100100001 0000000000 0000000000 0000000000		Bajtek dostaje tę macierz z przepermutowanymi kolumnami i wierszami.
			12	Bajtek informuje, że odkodował z macierzy liczbę 12.
11				Algosia dostaje do zakodowania liczbę 11.
	1110000000 1000000000 1000000000 1000000000 1000000000 1000000000 1000000000 1000000000 1000000000 1000000000			Algosia wysyła macierz kodującą liczbę 11.
		0000000100 0000000100 1000000101 0000000100 0000000100 0000000100 0000000100 0000000100 0000000100 0000000100		Bajtek dostaje tę macierz z przepermutowanymi (potencjalnie w inny sposób niż poprzednio) kolumnami i wierszami.
			11	Bajtek informuje, że odkodował z macierzy liczbę 11.

Ocenianie

Zestaw testów składa się z 10 grup, za każdą z nich można dostać 0 lub 1 punkt. Ograniczenia na wartość n w poszczególnych grupach wyglądają następująco:

Numer grupy	$n \leq$
1	10^{10}
2	10^{11}
3	10^{12}
4	10^{13}
5	10^{14}
6	10^{15}
7	$5 \cdot 10^{15}$
8	10^{16}
9	$2 \cdot 10^{16}$
10	$3 \cdot 10^{16}$

Testowanie lokalne

W sekcji *Pliki* dostępny jest interaktor `kodsoc.cpp`. Interaktor dostarczony w plikach **nie permutuje** macierzy przed wysłaniem do Bajtka; poza tą różnicą interakcję z programami zawodników przeprowadza dokładnie tak samo jak ten sprawdzający zgłoszenia na SIO. `dlazaw`. Należy go uruchamiać komendą:

```
python3 kodrun.py [rozwiazanie] < [test]
```

przy czym pliki `kodsoc.cpp` oraz `oi.h` muszą znajdować się w tym samym folderze. Format, w którym interaktor przyjmuje wejście, jest opisany w komentarzu w pliku `kodsoc.cpp`. Dodatkowe opcje skryptu `kodrun.py` można poznać, uruchamiając komendę `python3 kodrun.py -h`.

Uruchomienia próbne

W tym zadaniu są dostępne. W uruchomieniu próbnym należy załączyć wejście dla interaktora. Uruchomienie próbne zwraca werdykt programu. **Interaktor używany w uruchomieniach próbnych jest taki sam, jak ten w `dlazaw`.**

Zasady fair play

Komunikowanie się pomiędzy programami w inny sposób niż za pomocą przebiegu rozgrywki, np. poprzez wysłanie ruchu przez jeden program z opóźnieniem i odczytanie zegara przez drugi, jest zabronione. W przypadku stwierdzenia przez Jury próby komunikowania się między programami w niedozwolony sposób, zgłoszenie może zostać usunięte, a w rażących przypadkach może zostać podjęta decyzja o dyskwalifikacji autora tego zgłoszenia z całego konkursu.