Zadanie: SKU
Skup akcji [A]

POTYCZKI ALGORYTMICZNE

PA 2017, runda 4. Dostępna pamięć: 128 MB. Limit czasu: 12 s. *23.11.2017*

Uwaga! *To jest zadanie rozproszone. Zanim zaczniesz je rozwiązywać, zapoznaj się z informacjami dotyczącymi tego typu zadań dostępnymi w serwisie internetowym zawodów.*

Bajtazar, wielki entuzjasta handlu akcjami, jest właścicielem domu maklerskiego Share'EmAll. Ma on swoje oddziały w $N =$ `NumberOfNodes`() krajach. W każdym z tych krajów na giełdzie operuje pewna liczba firm.

Share'EmAll zdobył swoją pozycję na rynku dość nietypową strategią zakupu akcji: raz na jakiś czas kupuje je hurtowo. Oznaczmy przez M łączną liczbę wszystkich firm działających na giełdzie w państwach, w których Share'EmAll ma oddziały. Dom maklerski zakupi co najmniej jedną i co najwyżej M akcji każdej firmy – przy czym liczba kupionych akcji będzie *inna dla każdej firmy*. Bajtazar liczy na to, że jeśli kupi wszystkie akcje możliwie najtaniej, to korzystne zawirowania na rynku spowodują, iż będzie mógł je sprzedać dużo drożej.

Zazwyczaj wyznaczanie strategii zakupu jest bardzo prostym zadaniem – w końcu w tym Share'EmAll jest dobry. Niestety, informatycy z domu maklerskiego nie byli równie dobrzy w zabezpieczaniu połączeń pomiędzy swoimi siedzibami. W wyniku ataku hakerskiego infrastruktura pozwalająca na przesył danych na liniach pomiędzy niektórymi krajami została naruszona – w wielu miejscach infrastruktury zainstalowany został złośliwy program podmieniający wszystkie przesyłane liczby na zera. Poprawny przesył danych na takich liniach jest w ogóle niemożliwy, albo też możliwy tylko w jednym kierunku.

Bajtazar ma duży problem – musi natychmiast zmobilizować informatyków z całego świata, by pomogli mu usunąć skutki ataku hakerskiego! Optymistycznie zakłada on jednak, że nawet podczas ataku strategię zakupu akcji wciąż da się wyznaczyć. Czy jesteś w stanie mu w tym pomóc?

Komunikacja z biblioteką

Aby móc rozwiązywać zadanie, należy na początku programu dopisać dyrektywę

```
#include "skup.h"
```

Ponumerujmy wszystkie kraje liczbami całkowitymi od 0 do $N-1$. Wtedy i-ty komputer opisuje akcje wszystkich firm operujących w i-tym kraju.

Dodana biblioteka oferuje następujące funkcje:

- `long long NumberOfCompanies();` – uruchomiona na i-tej instancji programu, zwraca liczbę firm działających w i-tym kraju. Zachodzi $1 \leq$ `NumberOfCompanies`() ≤ 1000.
- `long long GetShareCost(int Id);` – uruchomiona na i-tej instancji programu, jeśli tylko zachodzi warunek $0 \leq$ `Id` $<$ `NumberOfCompanies`(), zwraca cenę (w bajtalarach) akcji firmy o numerze `Id`. Zakładamy, że firmy w każdym kraju ponumerowane są liczbami całkowitymi od 0 do `NumberOfCompanies`() $- 1$. Zachodzi $1 \leq$ `GetShareCost`(`Id`) $\leq 10^9$.

Ponadto, funkcje `PutChar`, `PutInt` oraz `PutLL` zadeklarowane w `message.h` zmieniają swoje działanie:

- `void PutChar(int target, char x);` – jeśli złośliwy program nie blokuje przesyłania danych z obecnego komputera do komputera `target`, do bufora do wysłania jest napisany znak `x`. W przeciwnym razie dopisany jest znak `(char)0`; to jest, liczba 8-bitowa równa 0.
- `void PutInt(int target, int x);` – zachowuje się analogicznie do `PutChar`, tylko dodaje do bufora `x` albo `(int)0`.
- `void PutLL(int target, long long x);` – zachowuje się analogicznie do `PutChar`, tylko dodaje do bufora `x` albo `(long long)0`.

Zwróć uwagę, że nie wiadomo, które połączenia zostały naruszone przez wirusa.

Wyjście

Pojedyncza instancja powinna wypisać pojedynczą liczbę całkowitą – optymalny koszt zakupu akcji zgodnie ze strategią Bajtazara.

Ograniczenia

- Niech N będzie liczbą państw (i zarazem liczbą instancji). Wtedy $10 \leq N \leq 100$. Można ponadto założyć, że w k-tej paczce testów, $1 \leq k \leq 10$, będzie zachodzić $N = 10k$.
- Limit wiadomości wychodzących z pojedynczej instancji: 1200 wiadomości.
- Limit łącznego rozmiaru wiadomości wychodzących z pojedynczej instancji: 8 MB.
- Wirus pozwala na rozwiązanie zadania. Oznacza to, że naruszone są takie (jednokierunkowe lub dwukierunkowe) połączenia, by było możliwe udzielenie odpowiedzi niezależnie od liczby firm działającej w każdym kraju i wartości ich akcji.
- Podany limit pamięci dotyczy pojedynczej instancji. Należy pamiętać, iż uwzględnia on pamięć potrzebną do przechowania jeszcze niewysłanych lub jeszcze nieprzeczytanych w całości wiadomości (nawet, jeśli składają się one z samych zer).
- Wywołanie funkcji `NumberOfCompanies`, `GetShareCost` trwa średnio nie dłużej niż 5 mikrosekund.

Przykładowe wywołanie programu

Przypuśćmy, że firma działa w trzech państwach (tj. $N = 3$). Dla poniższego przebiegu programu:

Numer instancji	Wywołanie funkcji	Zwrócona wartość
0	`NumberOfCompanies();`	**2**
	`GetShareCost(0);`	4
	`GetShareCost(1);`	7
1	`NumberOfCompanies();`	**1**
	`GetShareCost(0);`	1
2	`NumberOfCompanies();`	**2**
	`GetShareCost(0);`	5
	`GetShareCost(1);`	3

poprawnym wynikiem jest:

```
46
```

Wyjaśnienie do przykładu: wszystkie akcje mają kolejno koszty $4, 7, 1, 5, 3$. Optymalny koszt gwarantuje zakup odpowiednio $3, 1, 5, 2, 4$ akcji każdego typu. Uzyskany koszt wyniesie $4 \cdot 3 + 7 \cdot 1 + 1 \cdot 5 + 5 \cdot 2 + 3 \cdot 4 = 46$.

Testy przykładowe

Po wysłaniu rozwiązania od razu poznasz wyniki sprawdzania na poniższych testach przykładowych.

- `0a`: $N = 10$ krajów, można komunikować się pomiędzy każdą parą komputerów; i-ty kraj, $0 \leq i \leq 9$ zawiera $i + 1$ firm o akcjach o wartości $i + 1$ bajtalarów. Wynik to 8701.
- `0b`: $N = 50$ krajów. Dwa kraje mogą porozumieć się ze sobą bez zakłóceń, gdy ich numery są różnej parzystości. Każdy kraj zawiera po dwie firmy o akcjach o wartości 100 bajtalarów. Wynik to $\frac{100 \cdot 101}{2} \cdot 100 = 505\,000$.
- `0c`: $N = 100$ krajów; i-ty może przekazywać niezakłócone informacje tylko do kraju $i + 1$, zaś komputer o numerze $N - 1$ nie może przekazać niezakłóconej informacji nikomu. Ponadto, każdy kraj zawiera po 1000 firm o wartości 10^9 bajtalarów.

Eksperymenty

Pliki nagłówkowe `skup.1.h`, `skup.2.h`, `skup.3.h` odpowiadające powyższym testom przykładowym znajdują się w archiwum w systemie SIO. Zwróć uwagę, że te pliki nagłówkowe definiują makra `PutChar`, `PutInt`, `PutLL` symulujące działanie wirusa. Pamiętaj, że na systemie SIO wirus może być zaimplementowany inaczej.