Zadanie: OSA
Osady i warownie [A]

POTYCZKI ALGORYTMICZNE

PA 2017, runda 5. Dostępna pamięć: 512 MB. Limit czasu: 3 s. *24.11.2017*

Nie wszyscy o tym wiedzą, ale w granicach administracyjnych Bajtocji znajduje się Wyspa Bitocka. Z powodu bardzo niskich podatków stała się ona mekką dla kupców zjeżdżających z całego świata. Niestety, przez swoją strategiczną lokalizację na Morzu Gigabajtowym, wyspa jest łakomym kąskiem dla wrogich państw. Powoduje to konieczność organizacji zarówno infrastruktury osad kupieckich, jak również imponującej sieci warowni i innych budowli obronnych.

Wyspa ma kształt prostokąta o bokach długości n oraz m kilometrów. Rząd bajtocki postanowił podzielić więc wyspę na nm obszarów administracyjnych; każdy w kształcie kwadratu o boku jednego kilometra. Wiersze w tym podziale ponumerowano liczbami naturalnymi od 1 do n włącznie, zaś kolumny – liczbami od 1 do m. W każdym obszarze może znajdować się albo osada kupiecka, albo warownia, albo lasy i tereny rolnicze. Kupcy nie mają prawa wstępu do budowli obronnych, mają jednak pełną dowolność w poruszaniu się po pozostałych terenach. Możliwe jest przejście jedynie pomiędzy obszarami sąsiadującymi ze sobą bokiem.

Ponieważ istotnym jest zachowanie ciągłości handlu, rząd Bajtocji wydał dekret, zgodnie z którym zawsze musi istnieć możliwość przejścia (być może korzystającego z innych osad lub terenów rolniczych) pomiędzy dowolnymi dwoma istniejącymi osadami. Utrzymywanie tego dekretu jest jednak niełatwe. Co pewien czas Ministerstwo Obrony Bajtocji prosi o budowę nowej warowni na terenie rolniczym. Rząd w takiej sytuacji sprawdza, czy nowo stworzony obiekt nie stanie w sprzeczności z dekretem. Jeśli nie ma przeszkód, wniosek zostaje zaakceptowany i wdrażany w życie. Ponadto, czasami mieszkańcy osady uznają swoją lokalizację na niekorzystną i składają wniosek o przeniesienie jej do sąsiedniego obszaru administracyjnego. Taki wniosek jest natychmiast wdrażany w życie.

Dostałeś zlecenie stworzenia biblioteki, która będzie w stanie efektywnie oceniać, czy akceptacja kolejno przychodzących wniosków nie będzie stała w sprzeczności z dekretem rządu Bajtocji.

Komunikacja

Jako rozwiązanie zadania powinieneś dostarczyć bibliotekę udostępniającą opisane poniżej funkcje wykorzystywane potem przez program sprawdzający. Aby móc tego dokonać, należy wpisać w swoim programie wiersz

```
#include "osalib.h"
```

Powinieneś zaimplementować następujące funkcje:

- `void NowaWyspa(int n, int m, char **board)` – funkcja wywoływana przez program sprawdzający dokładnie raz na początku działania programu. Informuje Twoją bibliotekę o wymiarach wyspy ($n \times m$) oraz podaje obecne wykorzystanie wszystkich jednostek administracyjnych. Dla $1 \leq i \leq n$, $1 \leq j \leq m$, znak `board[i - 1][j - 1]` opisuje typ zagospodarowania pozycji (i, j) i jest równy:
  - `.` (kropka), jeśli w danym miejscu znajdują się tereny rolne;
  - `W`, jeśli znajduje się tam warownia;
  - `K`, jeśli znajduje się tam osada kupiecka.
- `int NowaWarownia(int r, int c)` – prosi o budowę nowej warowni na obszarze o współrzędnych (r, c). Możesz założyć, że we wskazanym polu znajdują się obecnie tereny uprawne. Funkcja powinna zwrócić `1`, gdy w obecnej chwili można wybudować warownię zgodnie z zasadami. W tym przypadku budowla ta jest natychmiast wznoszona. W przeciwnym razie funkcja powinna zwrócić `0`.
- `void PrzeniesOsade(int r1, int c1, int r2, int c2)` – wnioskuje o przeniesienie osady kupieckiej z pola (r_1, c_1) na pole (r_2, c_2). Możesz założyć, że na polu (r_1, c_1) do tej pory była osada, pole (r_2, c_2) było niezagospodarowane oraz oba pola są sąsiednie, zatem $|r_1 - r_2| + |c_1 - c_2| = 1$. W wyniku tej operacji pole (r_1, c_1) staje się terenem rolnym, zaś (r_2, c_2) – osadą.

Ograniczenia

We wszystkich testach zachodzi $1 \leq n, m \leq 1000$. Ponadto, żadna z funkcji `NowaWarownia`, `PrzeniesOsade` nie będzie wywołana więcej niż 1 000 000 razy.

Eksperymenty

W dziale *Pliki* możesz znaleźć plik nagłówkowy biblioteki `osalib.h` oraz przykładowy program `osarunner.c` uruchamiający bibliotekę na teście przykładowym. Aby skompilować rozwiązanie (`osalib.c` w języku C lub `osalib.cpp` w języku C++), wszystkie pliki należy umieścić w jednym katalogu, a następnie wykonać następujące polecenie:

- C: `gcc -std=c11 -static -O2 osalib.c osarunner.c -lm`
- C++: `g++ -std=c++11 -static -O2 osalib.cpp osarunner.c`

Wygenerowany plik wykonywalny dla każdej odpowiedzi na zapytanie biblioteczne `NowaWarownia` wypisze `TAK` lub `NIE` w zależności od tego, czy biblioteka zgodziła się na budowę budowli, czy nie.

Ocenianie

Biblioteka otrzyma punkty za dany test, gdy odpowie poprawnie na wszystkie zapytania postaci `NowaWarownia`.

Twoja biblioteka nie powinna implementować funkcji `main` – dokona tego program sprawdzający. Ponadto, biblioteka nie może wczytywać nic ze standardowego wejścia ani wypisywać żadnych danych na standardowe wyjście. W tym przypadku należy liczyć się z błędem wykonania lub zgłoszeniem błędnej odpowiedzi przez program sprawdzający. Może za to wypisywać informacje diagnostyczne na standardowe wyjście błędów. Pamiętaj jednak, że zużywa to czas procesora.

Przykładowy przebieg programu

Wywołanie	**Wynik**	**Uwagi**

`NowaWyspa(3, 4, board)`	—	Zmienna `board` jest tablicą znaków o wymiarach 3×4:
		`..WK`
		`WK..`
		`...K`

`NowaWarownia(3, 2)`	1	Wybudowaliśmy warownię w trzecim wierszu i drugiej kolumnie.
`NowaWarownia(2, 3)`	0	Budowa tej warowni odetnie osadzie $(2, 2)$ dostęp do osady $(1, 4)$.
`PrzeniesOsade(2, 2, 2, 3)`	—	
`PrzeniesOsade(2, 3, 2, 4)`	—	
`NowaWarownia(2, 3)`	1	Budowa warowni jest już możliwa.