Zgubiona kotka

Japoński Obóz Wiosenny 2020, dzień 3
22 marca 2020

Kod zadania: **stray**
Limit czasu: **2 s**
Limit pamięci: **537 MB**

Anthony jest mrówką i żyje w mieście JOI, w którym jest N skrzyżowań, ponumerowanych od 0 do $N-1$, Anthony mieszka przy skrzyżowaniu 0. W mieście jest także M dróg, z których droga o numerze i (dla $i = 0, \ldots, M-1$) łączy dwa skrzyżowania U_i oraz V_i. Dowolne dwa skrzyżowania łączy co najwyżej jedna droga. Drogi są dwukierunkowe i wiadomo, że z każdego skrzyżowania da się (jedną lub kilkoma drogami) dojechać do każdego innego.

Catherine jest z kolei kotem i planuje odwiedzić Anthony'ego. Niestety, dla przybyszów z zewnątrz wszystkie drogi i wszystkie skrzyżowania w mieście wyglądają tak samo, więc Catherine na pewno by się zgubiła. Aby ułatwić jej dotarcie do swojego domu, Anthony postanowił rozłożyć na ulicach miast kolorowe żetony – na każdej ulicy jeden żeton. Anthony ma A różnych kolorów żetonów, numerowanych od 0 do $A-1$. W każdym kolorze może przygotować dowolnie wiele żetonów, ale żetony tego samego koloru wszystkie będą identyczne i nierozróżnialne.

Catherine rozpoczyna wędrówkę w pewnym skrzyżowaniu i działa według następującej strategii:

- Jeśli znajduje się w skrzyżowaniu 0, rozpoznaje je i kończy swoją drogę.
- Jeśli nie jest w skrzyżowaniu 0, patrzy na wszystkie wychodzące z niego drogi i liczy dla każdego koloru, ile jest dróg oznaczonych tym kolorem, przy czym (uwaga!) pomija przy liczeniu drogę, którą przyszła (w końcu jest kotem, nikomu nie musi się tłumaczyć).
- Na podstawie tej informacji wybiera drogę, którą wyjdzie z tego skrzyżowania (może to być droga, którą przyszła; spośród dróg tego samego koloru może równie dobrze wybrać każdą).

Anthony i Catherine nie wiedzą jeszcze, z którego skrzyżowania zacznie Catherine. Chcieliby jednak tak rozłożyć żetony i zaplanować strategię znajdowania drogi, aby nie nadłożyła ona zbyt wiele czasu. Dokładnie rzecz biorąc, jeśli najkrótsza droga od skrzyżowania początkowego do skrzyżowania końcowego (0) miała długość d (wymagała przejścia d drogami), to Catherine chciałaby umieć znaleźć drogę długości co najwyżej $d + B$.

Twoim zadaniem jest napisać dwa programy, które to umożliwią:

- pierwszy program, mając dany układ miasta (skrzyżowań i dróg) podpowie Anthony'emu, jak rozstawić żetony;
- drugi program będzie podpowiadał Catherine, którą drogę ma wybrać na każdym skrzyżowaniu.

Implementacja

Do oceny przesyłasz dwa pliki. Pierwszy z nich powinien nazywać się `Anthony.cpp` i załączać na początku nagłówek `Anthony.h`. Musi implementować funkcję:

- `std::vector<int> Mark(int N, int M, int A, int B,`
  `std::vector<int> U, std::vector<int> V)`

  Funkcja ta zostanie wywołana dokładnie raz.

  - Parametr `N` to liczba skrzyżowań N.
  - Parametr `M` to liczba dróg M.
  - Parametr `A` to liczba kolorów żetonów A.
  - Parametr `B` to maksymalna dopuszczalna różnica między najkrótszą drogą i drogą Catherine.
  - Parametry `U` i `V` to wektory długości M, przy czym `U[i]` oraz `V[i]` to numery U_i oraz V_i skrzyżowań, które łączy i-ta droga $(0 \leq i \leq M-1)$.
  - Funkcja powinna zwrócić `x` – wektor o długości M. Jeśli długość będzie inna niż M, Twój program dostanie komunikat **Wrong Answer [1]**. Wartość `x[i]` $(0 \leq i \leq M-1)$ to kolor żetonu, który Anthony ma położyć na i-tej drodze. Powinna być spełniona nierówność $0 \leq$ `x[i]` $\leq A-1$ – jeśli nie będzie, Twój program dostanie komunikat **Wrong Answer [2]**.

Drugi plik powinien nazywać się `Catherine.cpp` i załączać na początku nagłówek `Catherine.h`. Program ten ma na celu podanie strategii Catherine, powinien w tym celu implementować następujące funkcje:

- `void Init(int A, int B)`

  Ta funkcja zostanie wywołana dokładnie raz, na początku działania programu.

  - Parametr `A` to liczba kolorów żetonów A.
  - Parametr `B` to maksymalna dopuszczalna różnica między najkrótszą drogą i drogą Catherine.

- `int Move(std::vector<int> y)`

  Ta funkcja będzie wywoływana za każdym razem, kiedy Catherine dochodzi do skrzyżowania (innego niż 0).

  - Parametr `y` to tablica długości A opisująca, co widzi Catherine na skrzyżowaniu: `y[j]` to liczba dróg wychodzących ze skrzyżowania, które mają żeton w kolrze j ($0 \le j \le A-1$), nie licząc drogi, którą Catherine weszła (o ile istnieje taka droga!).
  - Wartość `z` zwracana przez funkcję powinna spełniać warunek $-1 \le \texttt{z} \le A-1$. Jeśli warunek ten nie będzie spełniony, program dostanie komunikat **Wrong Answer [3]**. Wartość $\texttt{z} = -1$, oznacza, że Catherine powinna zawrócić i pójść drogą, którą przyszła do skrzyżowania. Wartość z taka, że $0 \le \texttt{z} \le A-1$ oznacza, że powinna wybrać drogę o kolorze `z`. Jeśli przy pierwszym wywołaniu funkcji `Move` Twój program odpowie wartością $\texttt{z} = -1$, dostatnie komunikat **Wrong Answer [4]**. Jeśli kiedykolwiek odpowie wartością `z` taką, że $0 \le \texttt{z} \le A-1$, ale $\texttt{y[z]} = 0$, dostanie komunikat **Wrong Answer [5]**.

Jeśli Catherine wybierze drogę inną niż ta, którą weszła, sprawdzarka poprowadzi ją jedną z dróg w kolorze wybranym przez wartość zwróconą przez `Move`.

Sprawdzarka może wybrać tę drogę dowolnie, w szczególności wcale nie musi jej uczciwie losować.

Jeśli Catherine nie dojdzie do miasta 0 w $d + B$ ruchach (czyli po $d + B$ wywołaniach funkcji `Move`), Twój program dostanie komunikat **Wrong Answer [6]**.

Uwagi

- Twój program może na własny użytek implementować też inne funkcje, może też używać globalnych zmiennych. Oba Twoje pliki zostaną skompilowane wspólnie ze sprawdzarką, tworząc jeden plik wykonywalny. Zmienne globalne i funkcje pomocnicze powinny być zadeklarowane w anonimowej przestrzeni nazw (**unnamed namespace**), żeby uniknąć konfliktu z innymi plikami. Podczas sprawdzania program zostanie osobno uruchomiony dla Anthony'ego, i osobno dla Catherine. Programy dla Anthony'ego i Catherine nie mogą używać wspólnych zmiennych globalnych.
- Twój program nie może używać standardowego wejścia i wyjścia, ani komunikować się z żadnymi innymi plikami poza wymienionymi. Wolno mu jednak wypisywać pomocnicze informacje na wyjście błędów (`stderr`).

Kompilacja, uruchomienie przykładowe

Ze strony konkursu możesz pobrać spakowane archiwum, które zawiera przykładową, pomocniczą sprawdzarkę dla Twojego rozwiązania. Archiwum zawiera też przykładowy plik źródłowy programu.

Sprawdzarka to plik `grader.cpp`. Aby przetestować swój program, umieść w jednym katalogu pliki `grader.cpp`, `Anthony.cpp`, `Catherine.cpp`, `Anthony.h` oraz `Catherine.h` i uruchom poniższą komendę, żeby skompilować swój program:

```
g++ -std=gnu++14 -O2 -o grader grader.cpp Anthony.cpp Catherine.cpp
```

Jeśli kompilacja się uda, stworzy się wykonywalny plik `grader`.

Pamiętaj, że właściwa sprawdzarka jest inna niż dostarczona przykładowa. Ta przykładowa uruchamia się jako pojedynczy proces, który czyta dane ze standardowego wejścia i wypisuje na standardowe wyjście.

Wejście przykładowej sprawdzarki

Przykładowa sprawdzarka czyta ze standardowego wejścia następujące dane:

$N\ M\ A\ B\ S$
$U_0\ V_0$
$\vdots$
$U_{M-1}\ V_{M-1}$

Wartość S to skrzyżowanie, na którym zaczyna Catherine, pozostałe wartości mają to samo znaczenie, co w opisach powyżej.

Wyjście przykładowej sprawdzarki

Jeśli wykonanie programu zakończy się bez błędu, przykładowa sprawdzarka wypisze na standardowe wyjście jeden z następujących komunikatów:

- Komunikat "`Wrong Answer [numer błędu]`", gdzie `numer błędu` to liczba od 1 do 5;
- Jeśli Catherine nie dojdzie do skrzyżowania 0 po $N + B$ przejściach drogami, przykładowa sprawdzarka wypisze "`Wrong Answer; Number of moves > N + B`".
- Jeśli Catherine dojdzie do skrzyżowania 0, przykładowa sprawdzarka wypisze "`Number of moves = liczba ruchów`", podając liczbę ruchów Catherine, jaką wykonał Twój program.

  Zauważ, że przykładowa sprawdzarka nie rozróżnia rozwiązania poprawnego od takiego, które dałoby **Wrong Answer [6]**.

Nawet jeśli Twój program zrobił więcej niż jeden z możliwych błędów, na wyjście wypisze się tylko jeden.

Kiedy Catherine ma do wyboru kilka dróg tego samego koloru, przykładowa sprawdzarka losuje jedną z nich z równym prawdopodobieństwem. Używa do tego liczb pseudolosowych z pewnym początkowym stanem generatora (ziarnem/*seed*). Możesz wywołać sprawdzarkę podając jej własny seed generatora, poleceniem:

```
./grader seed
```

gdzie *seed* jest liczbą oznaczającą ziarno.

Ograniczenia

- $2 \leq N \leq 20\,000$.
- $1 \leq M \leq 20\,000$.
- $1 \leq S \leq N - 1$ (S to numer miasta, w którym zaczyna Catherine).
- $0 \leq U_i < V_i \leq N - 1$ $(0 \leq i \leq M - 1)$.
- $(U_i, V_i) \neq (U_j, V_j)$ $(0 \leq i < j \leq M - 1)$.
- Z każdego miasta da się, jedną lub wieloma drogami, przejść do każdego innego.

Podzadania

1. (2 punkty) $A = 4,\ B = 0,\ M = N - 1$.
2. (2 punkty) $A = 4,\ B = 0$.
3. (2 punkty) $A = 3,\ B = 0,\ M = N - 1$.
4. (9 punktów) $A = 3,\ B = 0$.
5. (5 puunktów) $A = 2,\ B = 2N,\ M = N - 1,\ 6 \leq N \leq 500$.
6. (71 punktów) $A = 2,\ B = 12,\ M = N - 1$.
7. (9 punktów) $A = 2,\ B = 6,\ M = N - 1$.

Przykładowa komunikacja

Poniżej podane jest przykładowe wejście dla przykładowej sprawdzarki, i odpowiadające mu możliwe wywołania funkcji.

Przykładowe wejście 1
7 6 2 6 1 0 2 0 4 1 2 1 3 1 5 4 6

Anthony		Catherine	
Wywołanie	Zwraca	Wywołanie	Zwraca
`Mark(7,6,2,6,[0,0,1,1,1,4],[2,4,2,3,5,6])`	`[1,0,0,1,0,1]`		
		`Init(2,6)`	
		`Move([2,1])`	`0`
		`Move([0,0])`	`-1`
		`Move([1,1])`	`0`
		`Move([0,1])`	`1`

W tej przykładowej komunikacji, Catherine odwiedzi po kolei skrzyżowania 1, 5, 1, 2, 0. Wartość $d = 2$, a Catherine wykonuje 4 ruchy.

Ten przykładowy test spełnia założenia Podzadania 7.

Wśród plików, które możesz pobrać ze strony zawodów, `sample-02.txt` spełnia ograniczenia Podzadania 4.