**Wojciech Nadara**
Treść zadania, Opracowanie

**Michał Glapa, Marek Sokołowski**
Program

**Dostępna pamięć: 256 MB.** *OI, etap III, dzień pierwszy, 13.04.2016*

# Posłaniec

*Bajtazar po długim okresie swojego panowania w królestwie Bajtocji stwierdził, że to zajęcie bardzo go wyczerpało, i ustąpił z tronu. Jednak przywykł on do życia w wyższych sferach i chciałby pozostać na bieżąco z najważniejszymi wiadomościami dotyczącymi królestwa i dworu. Dlatego postanowił zostać królewskim posłańcem.*

*Już pierwszego dnia na nowym stanowisku zlecono mu dostarczenie pilnej wiadomości pomiędzy dwoma miastami królestwa. Stwierdził on jednak, że w trakcie pracy zrobi sobie małą wycieczkę krajoznawczą i niekoniecznie pojedzie najkrótszą możliwą trasą. Oczywiście nie może dopuścić, aby nowy król się o tym dowiedział – w końcu posłaniec powinien dostarczać wiadomości tak szybko, jak to tylko możliwe!*

*Wszystkie połączenia drogowe w Bajtocji są jednokierunkowe. Bajtazar zna doskonale całe królestwo i wie, między którymi miastami istnieją połączenia drogowe. Zadeklarował on liczbę połączeń, których chciałby użyć, przejeżdżając pomiędzy miastami, i planuje on przejechać pomiędzy nimi dowolną ścieżką wymagającą dokładnie tylu połączeń (nie zważając na to, ilu połączeń tak naprawdę wymaga taka podróż). W trakcie swojej podróży Bajtazar nie może jednak pojawić się w początkowym ani końcowym mieście więcej niż raz, gdyż wtedy wzbudziłby podejrzenia u królewskich oficjeli. Może on jednak wielokrotnie pojawiać się w innych miastach, jak też wielokrotnie korzystać z tych samych połączeń drogowych.*

*Pomóż naszemu bohaterowi i napisz program, który obliczy dla niego, na ile sposobów może on zrealizować swoją wycieczkę krajoznawczą. Innymi słowy, program ma wyznaczyć liczbę różnych ścieżek o zadanej długości pomiędzy dwoma wybranymi miastami królestwa (przy czym w mieście początkowym i końcowym można pojawić się tylko raz). Ponieważ wynik zapytania może być całkiem duży, wystarczy, że program poda resztę z dzielenia wyniku przez pewną wybraną przez Bajtazara liczbę.*

## Wejście

*W pierwszym wierszu standardowego wejścia znajdują się trzy liczby całkowite $n$, $m$ i $z$ ($n \geqslant 2$, $0 \leqslant m \leqslant n(n-1)$, $2 \leqslant z \leqslant 1\,000\,000\,000$) pooddzielane pojedynczymi odstępami, oznaczające odpowiednio: liczbę miast w Bajtocji, liczbę jednokierunkowych połączeń między nimi oraz liczbę wybraną przez Bajtazara. Miasta numerujemy liczbami od 1 do $n$.*

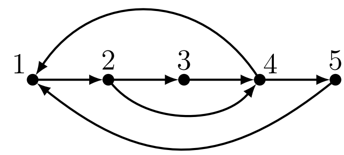
*Dalej następuje $m$ wierszy; każdy zawiera parę liczb całkowitych $a$, $b$ ($1 \leqslant a, b \leqslant n$, $a \neq b$) oddzielonych pojedynczym odstępem, opisującą jednokierunkowe połączenie z miasta o numerze $a$ do miasta o numerze $b$. Żadne połączenie nie jest podane na wejściu wielokrotnie.*

*W kolejnym wierszu wejścia znajduje się dodatnia liczba całkowita $q$ oznaczająca liczbę zapytań Bajtazara. W każdym z następnych $q$ wierszy znajduje się opis jednego zapytania. Opis taki składa się z trzech liczb całkowitych $u_i$, $v_i$ i $d_i$ ($1 \leqslant u_i, v_i \leqslant n$, $u_i \neq v_i$, $1 \leqslant d_i \leqslant 50$) pooddzielanych pojedynczymi odstępami, oznaczających, że Bajtazar ma przejechać z miasta o numerze $u_i$ do miasta o numerze $v_i$, używając dokładnie $d_i$ połączeń.*

## Wyjście

*Na standardowe wyjście należy wypisać dokładnie $q$ wierszy. W $i$-tym wierszu należy wypisać resztę z dzielenia przez $z$ liczby ścieżek z $i$-tego zapytania.*

## Przykład




*Dla danych wejściowych:*

```
5 7 10
1 2
2 3
3 4
4 5
5 1
2 4
4 1
2
2 1 3
5 3 6
```

*poprawnym wynikiem jest:*

```
2
1
```

**Wyjaśnienie do przykładu:** *Dla pierwszego zapytania mamy dwie możliwe ścieżki:* $2 \to 3 \to 4 \to 1$ *oraz* $2 \to 4 \to 5 \to 1$*; dla drugiego zapytania tylko jedną:* $5 \to 1 \to 2 \to 4 \to 1 \to 2 \to 3$.

**Testy „ocen”:**

`1ocen`: $n = 6$, $q = 10$, *pięć miast połączonych ze sobą bezpośrednio w obie strony; szóste miasto niepołączone z żadnym innym;*

`2ocen`: $n = 20$, $q = 100$, *miasta w Bajtocji położone są na okręgu; każde dwa sąsiednie miasta na tym okręgu są połączone ze sobą bezpośrednio w obie strony;*

`3ocen`: $n = 100$, $q = 500\,000$, *mapa Bajtocji ma kształt trójzębu.*

## Ocenianie

*Zestaw testów dzieli się na następujące podzadania. Testy do każdego podzadania składają się z jednej lub większej liczby osobnych grup testów.*

| **Podzadanie** | **Warunki** | **Liczba punktów** |
|---|---|---|
| *1* | $n \leqslant 20$, $q \leqslant 100$ | *12* |
| *2* | $n \leqslant 100$, $m \leqslant 500$, $q \leqslant 100$ | *20* |
| *3* | $n \leqslant 100$, $q \leqslant 10\,000$ | *38* |
| *4* | $n \leqslant 100$, $q \leqslant 500\,000$ | *30* |

# Rozwiązanie

Sytuację opisaną w treści zadania można w naturalny sposób przedstawić za pomocą grafu skierowanego. Miasta Bajtocji odpowiadają wierzchołkom grafu, a połączenia – krawędziom. Zbiory wierzchołków i krawędzi grafu oznaczymy jako $V$ oraz $E$ (mamy $|V| = n$, $|E| = m$).

W opisie rozwiązania będziemy używali dwóch sposobów reprezentacji grafu. Pierwszy z nich to macierz sąsiedztwa, czyli tablica $M$ rozmiaru $n \times n$, taka że $M[a][b] = 1$, jeżeli istnieje krawędź z wierzchołka $a$ do wierzchołka $b$, natomiast w przeciwnym przypadku $M[a][b] = 0$. Drugi natomiast to listy sąsiedztwa, czyli w naszym przypadku tablica $n$ list określających zbiory krawędzi wchodzących do poszczególnych wierzchołków grafu.

## Początkowe spostrzeżenia

W opisie rozwiązania będziemy wielokrotnie posługiwali się wartościami typu „liczba sposobów dojścia od wierzchołka $a$ do wierzchołka $b$ za pomocą dokładnie $k$ krawędzi". Oznaczmy taką wartość przez $ways(a, b, k)$. Zauważmy, że $ways(a, b, 0)$ jest równe 0 dla $a \neq b$ i 1 dla $a = b$. Ponadto mamy $ways(a, b, 1) = M[a][b]$. Zastanówmy się, jak można efektywnie obliczać wartości $ways(a, b, k)$ dla $k > 1$. Załóżmy, że chcemy przejść od wierzchołka $a$ do wierzchołka $b$ w $k$ krokach. Wówczas któryś wierzchołek $c$ odwiedzimy bezpośrednio przed wierzchołkiem $b$. Rozpatrując wszystkie takie wierzchołki $c$, że istnieje krawędź z $c$ do $b$, jesteśmy w stanie stwierdzić, że

$$ways(a, b, k) = \sum_{cb \in E} ways(a, c, k-1). \tag{1}$$

Wzór ten pozwala obliczyć wszystkie wartości $ways(a, b, k)$ dla $a, b \in V$, $0 \leqslant k \leqslant K$ za pomocą programowania dynamicznego, w kolejności rosnących wartości $k$. Mamy $O(n^2 K)$ stanów. Aby oszacować liczbę przejść, zauważmy, że dla ustalonych $a$ oraz $k$ we wzorze rozważamy wszystkie wierzchołki $b$ i dla każdego z nich wszystkie krawędzie wchodzące – dla ustalonych $a$ i $k$ mamy więc $m$ przejść. Tak więc obliczenia wykonamy za pomocą list sąsiedztwa w złożoności $O(nmK)$.

Wyrażając (1) w terminach macierzy sąsiedztwa, możemy także napisać

$$ways(a, b, k) = \sum_{c \in V} ways(a, c, k-1) \cdot M[c][b]. \tag{2}$$

W tym przypadku suma jest po wszystkich wierzchołkach $c$, a nie tylko po poprzednikach wierzchołka $b$, a fakt, czy istnieje krawędź od $c$ do $b$, wyrażamy za pomocą domnożenia składnika przez $M[c][b]$. Można stąd wysnuć wniosek, że $ways(a, b, k) = M^k[a][b]$, gdzie $M^k$ to macierz sąsiedztwa podniesiona do $k$-tej potęgi. Warto znać ten fakt, jednak w takiej konkretnie postaci nie będzie on nam dalej potrzebny.

Od tego momentu będziemy zakładać, że potrzebne nam wartości funkcji *ways* zostały obliczone na początku algorytmu.

## Rozwiązanie brutalne

Jedno z możliwych rozwiązań zadania polega na użyciu programowania dynamicznego do każdego z zapytań oddzielnie. Ustalmy jedno z nich, z parametrami $u$, $v$, $d$. Będziemy obliczać wartości $dp[w][k]$ oznaczające, na ile sposobów można dojść od wierzchołka $u$ do wierzchołka $w$ za pomocą $k$ krawędzi, nie odwiedzając w trakcie drogi ani $u$, ani $v$. Robimy to tak samo, jakbyśmy obliczali $ways(u, w, k)$, z jedyną różnicą, że pomijamy $w = u$ dla wszystkich wartości $k > 0$ oraz $w = v$ dla wszystkich wartości $k < d$. Takie rozwiązanie odpowiada na konkretne zapytanie w złożoności $O(md)$. Jeżeli zatem przez $D$ oznaczymy największą wartość $d_i$ występującą w zapytaniach, to możemy stwierdzić, że rozwiązanie działa w złożoności $O(mqD)$. Rozwiązanie o takiej złożoności wystarczało do przejścia pierwszych dwóch podzadań.

## Rozwiązanie wzorcowe

Wprowadźmy najpierw kilka terminów związanych z grafami. Ścieżki w grafie, w których krawędzie i wierzchołki mogą się powtarzać (czyli takie, jakie interesują nas w tym zadaniu), nazywamy *marszrutami*. Ponadto *wnętrzem marszruty* $v_1 v_2 \ldots v_{k-1} v_k$ nazwijmy zbiór wierzchołków $\{v_2, \ldots, v_{k-1}\}$. Będziemy mówili, że marszruta ma parametry $u, v, d$, jeżeli zaczyna się ona w wierzchołku $u$, kończy w wierzchołku $v$ oraz przechodzi przez $d$ krawędzi. Powiemy także, że zapytanie ma parametry $u$, $v$, $d$, jeżeli jesteśmy w nim proszeni o wyznaczenie liczby marszrut z takimi parametrami, które w swoim wnętrzu nie zawierają $u$ ani $v$.

Główny pomysł rozwiązania wzorcowego jest dość ogólną metodą liczenia *dobrych obiektów* jako różnicy liczby *wszystkich obiektów* oraz *złych obiektów*. Ustalmy konkretne zapytanie z parametrami $u, v, d$. *Dobrymi obiektami* są w naszym przypadku marszruty z $u$ do $v$ o długości $d$ odwiedzające $u$ i $v$ tylko na początku i końcu (odpowiednio). *Wszystkimi obiektami* będą marszruty z $u$ do $v$ o długości $d$, a *złymi obiektami* będą marszruty z $u$ do $v$ o długości $d$, których wnętrze zawiera $u$ lub $v$. Niech zatem $good(a, b, k)$ i $bad(a, b, k)$ oznaczają odpowiednio liczbę dobrych i złych marszrut o długości $k$ zaczynających się w $a$ i kończących w $b$. Będziemy rozważać tylko $a, b \in \{u, v\}$ (przy czym potencjalnie może zachodzić $a = b$ lub $a = v$, $b = u$).

*Złym wystąpieniem* wierzchołka na marszrucie o parametrach $a, b, k$ nazwijmy każde wystąpienie $u$ lub $v$ w jej wnętrzu. Przyjmijmy, że marszruta od $a$ do $b$ długości $k$ jest zła. Rozważmy ostatnie złe wystąpienie wierzchołka na tej marszrucie. Załóżmy, że było nim odwiedzenie wierzchołka $w \in \{u, v\}$ po $l$ krokach. Taka zła marszruta na odcinku swoich pierwszych $l$ kroków mogła być dowolną marszrutą o parametrach $a, w, l$, natomiast na kolejnym odcinku o długości $k - l$ była dobrą marszrutą o parametrach $w, b, k - l$. Zatem sumarycznie takich złych marszrut jest $ways(a, w, l) \cdot good(w, b, k - l)$. Ostatnie złe wystąpienie mogło być wystąpieniem zarówno wierzchołka $u$ jak i $v$ oraz $l$ mogło przyjąć jakąkolwiek wartość od 1 do $k - 1$. Stąd wniosek, że

$$\begin{aligned} bad(a, b, k) = {} & (ways(a, u, 1) \cdot good(u, b, k - 1) + \ldots + ways(a, u, k - 1) \cdot good(u, b, 1)) \\ & + (ways(a, v, 1) \cdot good(v, b, k - 1) + \ldots + ways(a, v, k - 1) \cdot good(v, b, 1)). \end{aligned}$$

Natomiast $good(a,b,k) = ways(a,b,k) - bad(a,b,k)$. Dzięki tym wzorom, dla zapytania o parametrach $u$, $v$, $d$ możemy napisać krótki pseudokod realizujący programowanie dynamiczne liczące wszystkie interesujące nas wartości $good(a,b,k)$, gdzie $a,b \in \{u,v\}$, $1 \leqslant k \leqslant d$.

```
for k := 1 to d do
  foreach a in {u, v} do
    foreach b in {u, v} do
      good[a][b][k] := ways(a, b, k);
      foreach last in {u, v} do
        for l := 1 to k − 1 do
          good[a][b][k] := good[a][b][k] − ways(a, last, l) · good[last][b][k − l];
```

Interesująca nas odpowiedź będzie obliczona w polu $good[u][v][d]$.

Na całe rozwiązanie składa się preprocessing tablicujący wszystkie wartości funkcji $ways(a,b,k)$ dla $k \leqslant D$ w złożoności $O(nmD)$ oraz odpowiadanie na zapytania w złożoności $O(qd^2)$, co daje rozwiązanie w złożoności $O(nmD + qD^2)$.