Wiktor Kuropatwa
Treść zadania, Opracowanie

Mateusz Baranowski
Program

Dostępna pamięć: 128 MB. *OI, etap II, dzień drugi, 14.02.2013*

Morskie opowieści

Młody Bajtinson uwielbia przesiadywać w portowej tawernie. Często wysłuchuje tam opowieści o przygodach wilków morskich. Początkowo wierzył we wszystkie, nawet najbardziej nieprawdopodobne zasłyszane historie. Z czasem stał się jednak podejrzliwy. Postanowił napisać program, który będzie sprawdzał, czy usłyszane przez niego opowieści są w ogóle możliwe. Niestety, kiepski z niego programista. Pomóż mu!

Na wodach, po których żeglują marynarze spotykani przez Bajtinsona, znajduje się n portów oraz m szlaków żeglownych między nimi. Istnienie szlaku żeglownego łączącego dwa porty oznacza, iż możliwe jest wykonanie rejsu, który zaczyna się w jednym z nich, zaś kończy w drugim. Taki rejs jest możliwy **w obie strony**.

Bajtinson poznał k historii morskich przygód. W każdej z nich opisywany marynarz rozpoczynał podróż w jednym z portów, wykonywał pewną liczbę rejsów szlakami żeglownymi i kończył w pewnym, być może tym samym porcie. Marynarz ten mógł odbyć wiele rejsów tym samym szlakiem żeglownym, w obu kierunkach.

Wejście

W pierwszym wierszu standardowego wejścia znajdują się trzy liczby całkowite n, m oraz k ($2 \leqslant n \leqslant 5000$, $1 \leqslant m \leqslant 5000$, $1 \leqslant k \leqslant 1\,000\,000$). Oznaczają one kolejno: liczbę portów na wodach, po których żeglują marynarze spotkani przez Bajtinsona, liczbę szlaków żeglownych oraz liczbę poznanych opowieści.

Następne m wierszy zawiera opis istniejących szlaków żeglownych. Opis pojedynczego szlaku składa się z jednego wiersza zawierającego dwie liczby całkowite oddzielone pojedynczym odstępem, a oraz b ($1 \leqslant a, b \leqslant n$, $a \neq b$), oznaczające numery portów, które łączy dany szlak.

Kolejne k wierszy zawiera opis zasłyszanych przez Bajtinsona przygód. Opis pojedynczej przygody składa się z trzech liczb całkowitych pooddzielanych pojedynczymi odstępami: s, t oraz d ($1 \leqslant s, t \leqslant n$, $1 \leqslant d \leqslant 1\,000\,000\,000$). Opis taki oznacza, iż bohater danej przygody rozpoczął ją w porcie o numerze s, zakończył w porcie o numerze t oraz wykonał w jej trakcie **dokładnie** *d rejsów.*

W testach wartych łącznie 50% punktów zachodzi dodatkowy warunek $n \leqslant 800$.

Wyjście

Twój program powinien wypisać na standardowe wyjście k wierszy; i-ty wiersz powinien zawierać słowo `TAK`*, jeżeli i-ta zasłyszana przygoda (według kolejności z wejścia) była możliwa. W przeciwnym wypadku odpowiedni wiersz powinien zawierać słowo* `NIE`*.*

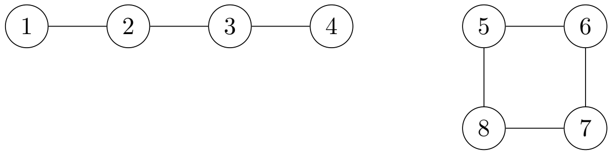
Przykład

Dla danych wejściowych:

```
8 7 4
1 2
2 3
3 4
5 6
6 7
7 8
8 5
2 3 1
1 4 1
5 5 8
1 8 10
```

poprawnym wynikiem jest:

```
TAK
NIE
TAK
NIE
```



Testy „ocen”:

`1ocen`: *n = 100, m = 4950, każdy port połączony z każdym, milion losowych przygód, wszystkie z odpowiedzią* `TAK`.

`2ocen`: *n = 100, m = 2450, rejsy możliwe między portami o numerach o tej samej parzystości, milion losowych przygód, połowa z odpowiedzią* `TAK`*, połowa z* `NIE`.

Rozwiązanie

Postawiony w zadaniu problem łatwo wyrazić w języku teorii grafów. Mamy dany graf nieskierowany, którego wierzchołki reprezentują porty, zaś krawędzie – szlaki żeglowne. Ponadto danych jest k zapytań postaci: *„czy z wierzchołka s do wierzchołka t istnieje ścieżka długości dokładnie d?”*. Naszym zadaniem jest odpowiedzenie na każde z tych zapytań.

Zauważmy, że na każde zapytanie, które dotyczy wierzchołka izolowanego (tzn. takiego, z którego nie wychodzą żadne krawędzie), odpowiadamy negatywnie. W dalszych rozważaniach zakładamy zatem, że w grafie nie ma wierzchołków izolowanych.

Kluczowe obserwacje

Patrząc na limity na długość ścieżek przedstawione w zadaniu, łatwo zauważyć, iż jakiekolwiek rozwiązanie przeglądające poszukiwane ścieżki w całości nie będzie miało szans zmieścić się w limitach czasowych. Musimy zatem umieć odpowiadać na zapytania o istnienie ścieżek bez wyznaczania ich *explicite*. Z pomocą przychodzi tu poniższa obserwacja:

Lemat 1. Jeśli z wierzchołka s do wierzchołka t istnieje ścieżka długości l, to istnieją również ścieżki długości $l+2$, $l+4$, $l+6$ itd.

Dowód: Rozważmy dowolną krawędź wychodzącą z wierzchołka t. Istniejącą ścieżkę z s do t długości l możemy przedłużyć o 2, przechodząc tą krawędzią raz w jedną, raz w drugą stronę. Tym samym, wykonując ten zabieg wielokrotnie, możemy przedłużyć długość danej ścieżki o dowolną parzystą wielkość. ■

Kluczową konsekwencją powyższego lematu jest to, że zbiór ścieżek między wierzchołkami s i t możemy bardzo łatwo opisać, znając dwie wartości:

- l_p – długość najkrótszej ścieżki długości parzystej między s i t oraz
- l_n – długość najkrótszej ścieżki długości nieparzystej między s i t.

Wtedy między wierzchołkami s i t istnieją ścieżki o długościach l_p, $l_p + 2$, $l_p + 4, \ldots$ oraz l_n, $l_n + 2$, $l_n + 4, \ldots$

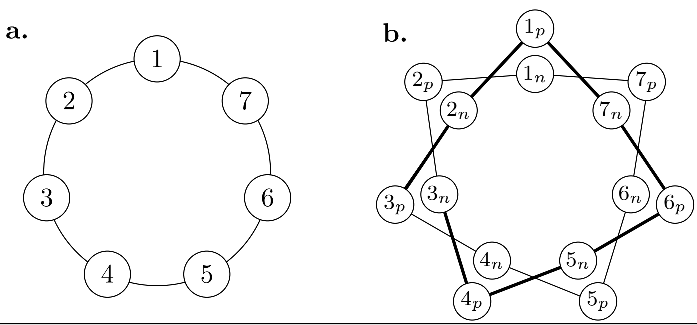
Przykładowo, na rys. 1a znajduje się graf G, będący cyklem o długości 7. Najkrótsza ścieżka o długości parzystej pomiędzy wierzchołkami 1 i 3 to $1 \to 2 \to 3$, a najkrótsza ścieżka o długości nieparzystej między nimi to $1 \to 7 \to 6 \to 5 \to 4 \to 3$. Między wierzchołkami 1 i 3 istnieją zatem ścieżki o długościach $2, 4, 6, \ldots$ oraz o długościach $5, 7, 9, \ldots$

Powyższe obserwacje są podstawą do sformułowania poniższego twierdzenia:

Twierdzenie 1. *Niech s i t będą wierzchołkami grafu, a l długością najkrótszej ścieżki o* **tej samej parzystości** *co liczba d. Odpowiedź na zapytanie „czy z wierzchołka s do wierzchołka t istnieje ścieżka długości dokładnie d?” jest twierdząca wtedy i tylko wtedy, gdy $d \geqslant l$.*

Dowód: Jeśli $d < l$, to oczywiste jest, że odpowiedź jest negatywna, gdyż l jest długością najkrótszej ścieżki o tej samej parzystości co d, więc krótsza ścieżka nie może istnieć. Jeśli zaś $d \geqslant l$, to możemy zapisać $d = l + 2k$ dla pewnej liczby naturalnej k. Między wierzchołkami s i t istnieje ścieżka długości l, zatem na mocy Lematu 1 istnieje również ścieżka długości d. ■

Twierdzenie to zachodzi też w przypadku, gdy ścieżka z s do t o szukanej parzystości długości nie istnieje. Wówczas przyjmujemy, że $l = \infty$.



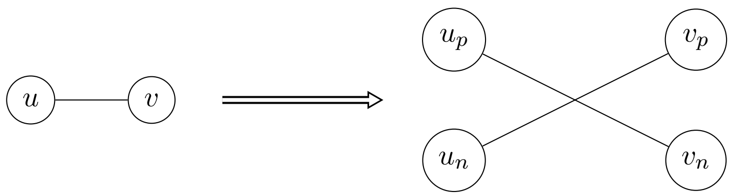
Rys. 1: **a.** Graf G będący cyklem długości 7. **b.** Graf G' uzyskany po transformacji grafu G z zaznaczonymi najkrótszymi ścieżkami długości parzystej i nieparzystej między wierzchołkami 1 i 3.

Tak więc, aby odpowiadać na zapytania z zadania, wystarczy obliczyć długość najkrótszej ścieżki długości parzystej bądź nieparzystej między wierzchołkami z zapytania i porównać ją z wartością d.

Najkrótsze ścieżki długości parzystej/nieparzystej

Potrzebujemy zatem rozwiązać następujący problem: mając dany graf nieskierowany $G = (V, E)$ i wyróżniony wierzchołek $s \in V$, należy wyznaczyć najkrótsze ścieżki długości parzystej/nieparzystej z wierzchołka s do wszystkich wierzchołków grafu.

W tym celu skonstruujmy graf nieskierowany $G' = (V', E')$. Dla każdego wierzchołka $u \in V$ dodajemy do V' dwa wierzchołki: u_p oraz u_n. Z kolei dla każdej krawędzi $uv \in E$ dodajemy do E' dwie krawędzie: $u_p v_n$ oraz $u_n v_p$; patrz też rys. 2.



Rys. 2: Transformacja krawędzi uv w grafie G w dwie krawędzie w grafie G'.

Zauważmy, iż każdej ścieżce długości parzystej z wierzchołka s do wierzchołka t w grafie G odpowiada ścieżka tej samej długości z wierzchołka s_p do wierzchołka t_p w grafie G'. Analogicznie, każdej ścieżce z wierzchołka s do wierzchołka t długości nieparzystej odpowiada ścieżka tej samej długości z wierzchołka s_p do wierzchołka t_n (patrz rys. 1b).

Ostatecznie, aby znaleźć najkrótsze ścieżki długości parzystej/nieparzystej z wierzchołka s do wszystkich pozostałych wierzchołków grafu G wystarczy skonstruować graf G', a następnie uruchomić na nim algorytm przeszukiwania wszerz (BFS) z wierzchołka s_p. Długość najkrótszej ścieżki o długości parzystej do wierzchołka t odczytujemy z odległości t_p od s_p, zaś nieparzystej z odległości t_n. Graf G' ma $2n$ wierzchołków i $2m$ krawędzi, zatem algorytm ten działa w czasie $O(n + m)$.

Rozwiązanie wzorcowe

Na początek wczytujemy wszystkie zapytania i dla każdego wierzchołka zapamiętujemy te, które go dotyczą. Teraz dla każdego wierzchołka:

(1) obliczamy wszystkie najkrótsze ścieżki długości parzystej/nieparzystej wychodzące z niego – wykorzystujemy w tym celu opisany powyżej algorytm,

(2) dysponując wyliczonymi w ten sposób wartościami, wyznaczamy odpowiedzi na zapytania dotyczące tego wierzchołka (wykorzystujemy tu Twierdzenie 1) i zapisujemy je w tablicy.

Na końcu wypisujemy odpowiedzi na zapytania w tej kolejności, w jakiej podane były na wejściu.

Zauważmy, że odpowiadając na zapytania dotyczące danego wierzchołka, potrzebujemy jedynie długości odpowiednich ścieżek wychodzących z tego wierzchołka. Jeśli spamiętamy wyniki przeszukiwań dotyczących tylko aktualnie rozważanego wierzchołka, możemy zaimplementować algorytm w złożoności pamięciowej $O(n+m+k)$.

Sumarycznie wykonamy $O(n)$ liniowych przeszukiwań grafu G', zaś odpowiedź na każde zapytanie zajmie czas stały. Ostatecznie, złożoność czasowa algorytmu wynosi więc $O(n(n+m)+k)$. Złożoność taka jest satysfakcjonująca przy limitach czasowych z zadania.

Rozwiązanie to znajduje się w plikach `mor.cpp` oraz `mor1.pas`, a także `mor2.cpp`.

Rozwiązania wolniejsze

Wykonując osobne przeszukiwanie dla każdego zapytania, otrzymujemy algorytm działający w złożoności $O(k(n+m))$. Implementacja tego rozwiązania znajduje się w plikach `mors1.cpp` oraz `mors2.pas`.

Rozwiązanie wykorzystujące potęgowanie macierzy do znalezienia ścieżki odpowiedniej długości nie wykorzystuje Twierdzenia 1 i działa w złożoności $O(kn^3 \log d)$, gdzie d to maksymalna długość ścieżki z zapytania. Zostało zaimplementowane w plikach `mors3.cpp` oraz `mors4.pas`.

Do wyznaczania długości najkrótszych ścieżek można też było użyć algorytmu Floyda-Warshalla. Uzyskane w ten sposób rozwiązanie działało w złożoności $O(n^3+k)$.

Rozwiązanie *online*

Wykonując wszystkie przeszukiwania przed wczytaniem zapytań i spamiętując ich wyniki, można było osiągnąć rozwiązanie o złożoności pamięciowej $O(n^2)$ zdolne do odpowiadania na zapytania *online*, czyli jedno zapytanie po drugim. Uważna implementacja pozwalała na zmieszczenie się w limitach pamięciowych i przejście wszystkich testów.

Testy

Rozwiązania sprawdzano na dziesięciu grupach testów. Testy a zawierały pojedynczą losowo wygenerowaną spójną składową. Testy b zawierały wiele losowych składowych, zaś c były zupełnie losowe. Ponadto, do części testów dodano wierzchołki izolowane.

Zawody III stopnia

opracowania zadań