Bajtocja

Dostępna pamięć: 32 MB.

Za górami, za lasami, za rzekami, za morzami leży kraj potężny i bogaty zwany Bajtocją. Panuje tam dobrotliwy król Bajtazar I Wielki, słynny ze swej troski o infrastrukturę kraju.

W Bajtocji znajduje się n miast. Władca rozkazał swym nadwornym architektom przygotować projekty nowych ponaddźwiękowych traktów konnych. Jako odpowiedź otrzymał m propozycji, każda z nich składa się z trzech liczb p, k, w, gdzie p i k są miastami końcowymi traktu (trakt łączy te miasta bezpośrednio i nie przebiega przez inne miasta), a w oznacza koszt zbudowania tego traktu. Każdym traktem można podróżować zarówno z miasta p do k, jak i w stronę przeciwną.

Zamierzeniem króla jest budowa sieci traktów w taki sposób, aby można było nimi przejechać między każdymi dwoma miastami, być może odwiedzając po drodze inne miejscowości. Bajtazar jest bardzo oszczędnym królem, więc postanowił zgodzić się tylko na taką sieć, która będzie możliwie najtańsza.

Zadanie

Opracuj program, który:

- wczyta ze standardowego liczbę miast, liczbę proponowanych traktów oraz opisy tych traktów,
- dla każdego traktu określi, czy istnieje taka sieć połączeń między miastami, zgodna z królewskimi rozkazami i w której rozpatrywany trakt, wybudowany za wskazaną kwotę, jest wykorzystywany,
- wypisze na standardowe wyjście zestawienie uzyskanych wyników.

Wejście

Pierwszy wiersz wejścia zawiera dwie liczby całkowite: liczbę miast n i liczbę proponowanych traktów m, rozdzielone pojedynczą spacją i spełniające warunki $2 \leq n \leq 7.000$, $1 \leq m \leq 300.000$. Każdy z kolejnych m wierszy zawiera po trzy liczby całkowite p, k, w rozdzielone pojedynczymi spacjami, opisujące proponowany trakt, przy czym p i k oznaczają miasta będące końcami traktu, zaś w jest ceną budowy tego traktu ($1 \leq p,k \leq n$, $1 \leq w \leq 100.000$).

Wyjście

W każdym z kolejnych m wierszy należy wypisać słowo "TAK" albo "NIE", w zależności od tego, czy można skonstruować plan budowy zgodny z życzeniem króla, dla którego trakt opisany w odpowiednim wierszu jest w nim zawarty. Możesz bezpiecznie założyć, że dla danych wejściowych zawsze istnieje plan budowy spełniający wymogi Bajtazara.

Przykład

Dla danych wejściowych:

```
6 10
1 2 2
```

```
1 6 1
1 5 3
4 1 5
2 6 2
2 3 5
4 3 4
3 5 4
4 5 4
5 6 3
```

poprawnym wynikiem jest:

```
TAK
TAK
TAK
NIE
TAK
NIE
TAK
TAK
TAK
TAK
```

Rozwiązanie

W języku teorii grafów postawiony w treści zadania problem dotyczy znajdowania tzw. *minimalnych drzew rozpinających* w grafie nieskierowanym. *Drzewem rozpinającym* grafu nazwiemy pewien podzbiór jego krawędzi, który umożliwia dotarcie z dowolnie wybranego wierzchołka do każdego innego. Innymi słowy odcinamy niektóre krawędzie w taki sposób, aby te, które zostały, tworzyły spójne drzewo, do którego należą wszystkie wierzchołki grafu. Jak sama nazwa wskazuje uzyskana konstrukcja jest drzewem, więc ma dokładnie $n-1$ krawędzi (gdzie n oznacza liczbę wierzchołków grafu), ponadto dodanie jakiejkolwiek krawędzi spowoduje powstanie cyklu. Te trywialne obserwacje przydadzą się w rozwiązaniu.

Minimalne drzewo rozpinające to takie drzewo rozpinające, w którym krawędzie należące do drzewa zostały wybrane w taki sposób, aby ich łączny koszt był jak najmniejszy. Rozwiązanie zadania polega na wyznaczeniu wszystkich krawędzi, które należą do jakiegokolwiek z minimalnych drzew rozpinających (może ich być wiele). W grafach jednostkowych każde drzewo rozpinające jest równocześnie minimalne. Niestety nasz problem dotyczy grafów o krawędziach ważonych, co komplikuje nieco rozwiązanie.

Ponieważ wyszukiwanie wszystkich minimalnych drzew rozpinających jest zadaniem co najmniej karkołomnym, popatrzmy na postawione zadanie z innej strony. Przypuśćmy, że mamy już jakieś minimalne drzewo rozpinające. Zastanawiamy się, czy pewna krawędź o wadze w może wchodzić w skład tego drzewa zamiast którejś z jego krawędzi.

Obserwacja 1. *Krawędź nienależąca do minimalnego drzewa rozpinającego może zostać wymieniona na jakąś krawędź z drzewa tylko wtedy, gdy obie mają te same wagi.*

Dowód. Obserwacja wynika wprost z minimalności drzewa — jeśli możemy podmienić krawędź z drzewa na jakąś inną, tańszą, otrzymując w rezultacie poprawne drzewo rozpinające o mniejszym koszcie, to poprzednie drzewo nie było minimalne, a więc sprzeczność. Jeśli będziemy próbowali zmienić na krawędź droższą, to możemy otrzymać nieminimalne drzewo rozpinające, czyli też niedobrze. □

Mamy warunek konieczny, lecz jeszcze nie wystarczający. Nie wiemy bowiem, czy zamiana jednego traktu na inny nie doprowadzi do rozspójnienia drzewa. Zamiast zastanawiać się nad znalezieniem pary krawędzi, które można wzajemnie wymienić, dodajmy po prostu krawędź–kandydata (o wadze w) do drzewa, uzyskując w ten sposób graf z jednym cyklem. Aby wrócić do sytuacji, w której ponownie mamy minimalne drzewo rozpinające, należy z tego cyklu usunąć dokładnie jedną krawędź z kosztem w. Jeśli istnieje taka krawędź (inna od tej, którą właśnie dodaliśmy), to wtedy dodana krawędź należy do któregoś z minimalnych drzew rozpinających.

Morał 1. *Znaleźliśmy pewien optymalny plan budowy, do którego krawędź $e = u \overset{w}{\leftrightarrow} v$* ***nie należy****. Krawędź e należy do jakiegoś innego optymalnego planu budowy, jeśli w ścieżce $u \rightsquigarrow v$ znajduje się krawędź o wadze w.*

Ten wniosek prowadzi do algorytmu rozwiązującego problem w czasie $O(mn)$ + koszt budowy dowolnego minimalnego drzewa rozpinającego: dla każdej krawędzi spoza drzewa przechodzimy całe drzewo w poszukiwaniu krawędzi o tej samej wadze leżącej na odpowiedniej ścieżce. Poszukiwanie łatwo przeprowadzić za pomocą wyszukiwania w głąb lub wszerz. Krawędzi mamy $O(m)$, każde przeszukiwanie działa liniowo w drzewie n-wierzchołkowym, z czego wprost wynika złożoność. Dla odpowiednio dużych grafów jest to jednak zbyt wolny algorytm.

Spójrzmy na ten sam problem z innej strony. Zamiast dla każdej krawędzi rozpatrywać jej przynależność do któregoś z drzew możemy dla każdego wierzchołka ze znalezionego już minimalnego drzewa badać wychodzące z niego krawędzie. Zmiana punktu widzenia pomaga w dostrzeżeniu prostej, acz skutecznej optymalizacji.

Obserwacja 2. *Dla każdego wierzchołka należącego do minimalnego drzewa rozpinajacego można w czasie $O(n)$ obliczyć wagi najcięższych krawędzi w ścieżkach prowadzących do dowolnego innego wierzchołka w tym drzewie.*

Dowód. Używamy algorytmu przeszukiwania w głąb do przekazywania informacji o najcięższej krawędzi na pokonywanych ścieżkach. Implementację zostawiamy jako ćwiczenie dla Czytelnika. □

Efektem rozwiązania tego podproblemu jest jednowymiarowa tablica, dla której obliczona wartość w indeksie i-tym oznacza wagę najcięższej krawędzi na ścieżce $u \rightsquigarrow i$, gdzie u oznacza wierzchołek początkowy. Z morału 1. i poprzedzającego go dowodu obserwacji wynika kolejny wniosek.

Morał 2. *Badamy krawędzie wychodzące z wierzchołka u, np. $e = u \overset{w}{\leftrightarrow} v$. Dzięki wcześniejszej prekalkulacji znamy wagę najcięższej krawędzi w ścieżce drzewowej $u \rightsquigarrow v$; oznaczmy ją w_{max}. Jeśli zachodzi $w = w_{max}$, to e należy do jakiegoś minimalnego drzewa rozpinającego.*

Złożoność czasowa algorytmu sprawdzającego krawędzie wychodzące z każdego wierzchołka (po wcześniejszym wyliczeniu najcięższych wag na ścieżkach) wynosi $O(n^2 + m)$+koszt budowy drzewa, co pozwala zmieścić się w limitach czasowych.

Do skonstruowania dowolnego minimalnego drzewa rozpinającego można użyć np. algorytmu Kruskala. Krok algorytmu, w którym dokonujemy scalenia dwóch poddrzew można zmodyfikować w taki sposób, aby od razu wyznaczać krawędzie, które mogą należeć do minimalnych drzew rozpinających. Wtedy otrzymujemy rozwiązanie w chwili zbudowania dowolnego takiego drzewa. Konstrukcję takiego algorytmu zostawiamy jako ciekawe ćwiczenie.