Tomasz Idziaszek
Treść zadania, Opracowanie

Kamil Dębowski
Program

Dostępna pamięć: 256 MB. *OI, etap III, dzień drugi, 17.04.2015*

Modernizacja autostrady

W Bajtocji znajduje się n miast połączonych gęstą siecią dróg. Niestety, większość z tych dróg jest kiepskiej jakości. W związku z tym w ostatnich latach zbudowano dodatkowo $n-1$ nowoczesnych autostrad. Autostradami można dojechać z każdego miasta do każdego innego, ale ten luksus nie jest za darmo – za przejechanie każdą autostradą trzeba uiścić osobną opłatę.

Obywatelom Bajtocji nie podobają się wysokie opłaty za przejazdy autostradami. Aby uspokoić opinię publiczną, minister transportu postanowił zmodernizować sieć autostrad. Tegoroczny budżet pozwoli jedynie na rozbiórkę jednej autostrady i zbudowanie jednej nowej (być może w tym samym miejscu, ale nowocześniejszej). Minister chce wybrać położenie starej i nowej autostrady tak, by nadal można było przejechać autostradami pomiędzy dowolną parą miast i żeby największa liczba autostrad na trasie pomiędzy dwoma miastami była tak mała, jak to tylko możliwe. Taki scenariusz nazywamy **optymistycznym**.

Z kolei minister skarbu chciałby, aby modernizacja autostrady przyczyniła się do podreperowania budżetu Bajtocji. Chciałby zatem, żeby po modernizacji nadal można było przejechać autostradami pomiędzy dowolną parą miast, ale żeby największa liczba autostrad na trasie między dwoma miastami była tak duża, jak to tylko możliwe. Taki scenariusz nazywamy **pesymistycznym**.

Pogłoski o planach obu ministrów przedostały się do prasy. Dziennikarz Bajtazar pisze na ten temat artykuł, w którym przedstawi najbardziej optymistyczny i najbardziej pesymistyczny scenariusz modernizacji. Pomóż mu i napisz program, który dostarczy mu konkretnych danych do artykułu.

Wejście

W pierwszym wierszu standardowego wejścia znajduje się jedna liczba całkowita n ($3 \leqslant n \leqslant 500\,000$) oznaczająca liczbę miast w Bajtocji. Miasta numerujemy liczbami od 1 do n. Dalej następuje $n-1$ wierszy opisujących autostrady. W i-tym z tych wierszy znajdują się dwie liczby całkowite a_i i b_i ($1 \leqslant a_i, b_i \leqslant n$, $a_i \neq b_i$) oddzielone pojedynczym odstępem, oznaczające, że istnieje autostrada pomiędzy miastami o numerach a_i i b_i.

W testach wartych łącznie 30% punktów zachodzi dodatkowy warunek $n \leqslant 1000$.

Wyjście

W pierwszym wierszu standardowego wyjścia należy wypisać pięć liczb całkowitych k, x_1, y_1, x_2 i y_2 opisujących optymistyczny scenariusz: liczba autostrad na najdłuższej trasie pomiędzy dwoma miastami to k, dokonana jest rozbiórka autostrady między miastami x_1 i y_1 oraz budowana jest autostrada między miastami x_2 i y_2. W drugim wierszu należy wypisać w takim samym formacie najbardziej pesymistyczny scenariusz. Miasta, między którymi autostradę budujemy lub rozbieramy, można podać w dowolnej kolejności. Jeśli istnieje więcej niż jedno rozwiązanie, Twój program powinien wypisać dowolne z nich.

Ocenianie

Twój program musi wypisać dwa wiersze. Odpowiedź w każdym z nich będzie oceniana niezależnie. Jeśli Twój program poda poprawną odpowiedź tylko dla jednego ze scenariuszy, uzyskasz połowę punktów za test. W takim wypadku zawartość drugiego wiersza nie ma znaczenia.

Przykład

Dla danych wejściowych:	*jednym z poprawnych wyników jest:*
`6`	`3 4 2 2 5`
`1 2`	`5 2 1 1 6`
`2 3`	
`2 4`	
`4 5`	
`6 5`	

Testy „ocen”:

`1ocen`: $n = 5$, *gwiazda;*

`2ocen`: $n = 1000$, *ścieżka;*

`3ocen`: $n = 2^{18}$, *każde miasto o numerze* $i > 1$ *jest połączone autostradą z miastem* $\lfloor \frac{i}{2} \rfloor$.

Rozwiązanie

Sieć autostrad łączących miasta w Bajtocji możemy przedstawić jako graf: jego wierzchołki odpowiadają miastom, a krawędzie – łączącym je autostradom. Graf jest spójny (bo autostradami można przejechać z dowolnego miasta do dowolnego innego) i posiada n wierzchołków i $n - 1$ krawędzi, zatem jest *drzewem*. *Ścieżką* pomiędzy parą wierzchołków w drzewie nazwiemy listę krawędzi, którymi musimy przejść, aby dostać się z jednego z tych wierzchołków do drugiego. Liczbę krawędzi na liście nazywamy *długością* tej ścieżki. Nas będą interesować najdłuższe ścieżki w drzewie; każdą taką ścieżkę nazywamy *średnicą*.

Zadanie polega na usunięciu jednej krawędzi z drzewa i dodaniu jednej nowej tak, by graf pozostał drzewem i żeby po tej operacji miał jak najmniejszą (jak największą) średnicę.

Optymalne łączenie poddrzew

Załóżmy, że wiemy, którą z krawędzi należy usunąć, i zastanawiamy się, które wierzchołki należy połączyć, by osiągnąć jeden z podanych celów. Jeśli usuniemy ustaloną krawędź, to drzewo rozpadnie się na dwa poddrzewa – nazwijmy je poddrzewami T_A i T_B. Aby graf pozostał drzewem, musimy dodać krawędź, która łączy pewien wierzchołek z poddrzewa T_A z pewnym wierzchołkiem z poddrzewa T_B.

Nietrudno się przekonać, że jeśli chcemy połączyć te dwa poddrzewa tak, by średnica nowo powstałego drzewa T była jak *największa*, to najlepiej jest połączyć wierzchołki, które są końcami średnic w poddrzewach T_A i T_B. Dla ustalenia uwagi, niech d_A i d_B będą długościami średnic w poddrzewach T_A i T_B, natomiast v_A i v_B będą pewnymi wierzchołkami leżącymi na końcach tych średnic. Połączenie krawędzią wierzchołków v_A i v_B da nam drzewo zawierające ścieżkę $p_{\max}$ o długości

$$d_{\max} = d_A + 1 + d_B.$$

Pokażemy teraz, że ścieżka ta jest średnicą. Niech p będzie dowolną średnicą drzewa T. Zauważmy, że średnica p nie może w całości zawierać się ani w poddrzewie T_A, ani w poddrzewie T_B, bo miałaby wtedy długość co najwyżej $\max(d_A, d_B) < d_{\max}$. Musi zatem przechodzić nowo dodaną krawędzią; niech p_A i p_B oznaczają długości kawałków tej średnicy, które leżą odpowiednio w poddrzewach T_A i T_B (więc długość średnicy p to $p_A + 1 + p_B$). Z definicji średnic mamy $p_A \leqslant d_A$ i $p_B \leqslant d_B$, zatem $p_A + 1 + p_B \leqslant d_{\max}$. To pokazuje, że ścieżka $p_{\max}$ jest również średnicą drzewa T. W podobny sposób można pokazać, że połączenie poddrzew T_A i T_B dowolną inną krawędzią nie da drzewa o średnicy większej niż $d_{\max}$.

Połączenie poddrzew T_A i T_B tak, by średnica nowo powstałego drzewa T była jak *najmniejsza*, jest trochę trudniejsze. Średnica drzewa T może albo znajdować się w całości w jednym z poddrzew T_A lub T_B (i wtedy ma długość d_A lub d_B), albo przechodzić nowo dodaną krawędzią. Tę krawędź powinniśmy wybrać tak, by zminimalizować odległości jej końców do najdalszych wierzchołków poddrzew T_A i T_B.

Zauważmy, że skoro poddrzewo T_A ma średnicę o długości d_A, to odległość między dowolnym wierzchołkiem u poddrzewa T_A a tym końcem średnicy, który znajduje się dalej od wierzchołka u, wynosi co najmniej $\lceil d_A/2 \rceil$. Z kolei dla wierzchołka u_A, który jest środkiem średnicy (jeśli długość średnicy jest nieparzysta, to jako środek można wybrać dowolny z dwóch wierzchołków leżących na środkowej krawędzi), odległość do dowolnego innego wierzchołka poddrzewa T_A wynosi co najwyżej $\lceil d_A/2 \rceil$.

Wynika z tego, że jeśli połączymy krawędzią wierzchołki u_A i u_B, będące środkami średnic poddrzew T_A i T_B, to w tak powstałym drzewie T długość najdłuższej ścieżki przechodzącej nowo dodaną krawędzią będzie wynosić $\lceil d_A/2 \rceil + 1 + \lceil d_B/2 \rceil$, i co więcej, połączenie tych poddrzew dowolną inną krawędzią nie da krótszej ścieżki. Podsumowując, najmniejsza średnica, jaką da się uzyskać, ma długość

$$d_{\min} = \max(d_A,\ d_B,\ \lceil d_A/2 \rceil + 1 + \lceil d_B/2 \rceil).$$

Rozwiązanie kwadratowe

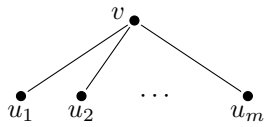
Rozpatrujemy niezależnie każdą z $n-1$ krawędzi drzewa jako kandydata do usunięcia. To wyznacza nam podział drzewa na dwa poddrzewa T_A i T_B. Dla każdego z nich znajdujemy następujące wartości: długość średnicy (d_A, d_B), jeden z jej końców (v_A, v_B) oraz środek (u_A, u_B). Jako kandydata na scenariusz pesymistyczny rozpatrujemy dodanie krawędzi między wierzchołkami v_A i v_B, co daje średnicę o długości $d_A + 1 + d_B$. Jako kandydata na scenariusz optymistyczny rozpatrujemy dodanie krawędzi między wierzchołkami u_A i u_B, co daje średnicę o długości $\max(d_A, d_B, \lceil d_A/2 \rceil + 1 + \lceil d_B/2 \rceil)$.

Znajdowanie średnicy w poddrzewie T_A (a potem niezależnie w poddrzewie T_B) wykonujemy standardowym algorytmem opartym o metodę programowania dynamicznego. Ukorzeniamy drzewo w dowolnym wierzchołku v_0. Następnie, zaczynając od liści i poruszając się w górę drzewa, dla każdego węzła v wyznaczamy dwie wartości: $d[v]$ – średnicę poddrzewa zaczepionego w węźle v oraz $h[v]$ – wysokość poddrzewa zaczepionego w v.

Jeśli v jest liściem, to oczywiście $h[v] = d[v] = 0$. W ogólnym przypadku, gdy węzeł v posiada m synów $u_1, u_2, \ldots, u_m$, dla których obliczyliśmy już wartości $d[u_i]$ i $h[u_i]$, wartości dla węzła v obliczamy z następującej rekurencji:

$$h[v] = 1 + \max_i h[u_i],$$
$$d[v] = \max\big(\max_i d[u_i],\ h[v],\ \max_{i \neq j}(h[u_i] + 2 + h[u_j])\big).$$

Poprawność wzoru na $d[v]$ wynika z faktu, że średnica w poddrzewie v może albo nie przechodzić przez węzeł v (czyli być w całości zawarta w poddrzewie zaczepionym w jednym z synów węzła v), albo kończyć się w węźle v, albo przechodzić przez węzeł v.



Wyznaczenie wartości $h[v]$ i $d[v]$ możemy zaimplementować w czasie liniowym od liczby synów m. Tak więc wyznaczenie długości średnicy całego drzewa (czyli $d[v_0]$) możemy zaimplementować w czasie $O(n)$.

Aby wyznaczyć końce średnic, będziemy dla poddrzewa zaczepionego w każdym wierzchołku v pamiętali nie tylko jego wysokość $h[v]$ i długość średnicy $d[v]$, lecz także dowolny z najgłębszych liści w tym poddrzewie $\ell[v]$ oraz parę węzłów $\alpha[v], \beta[v]$ będącą końcami tej średnicy. Uaktualnianie tych wartości nie jest trudne, przykładowo:

$$\alpha[v], \beta[v] = \begin{cases} \alpha[u_i], \beta[u_i] & \text{jeśli } d[v] = d[u_i] \text{ dla pewnego } i, \\ v, \ell[v] & \text{jeśli } d[v] = h[v], \\ \ell[u_i], \ell[u_j] & \text{jeśli } d[v] = h[u_i] + 2 + h[u_j] \text{ dla pewnych } i \neq j. \end{cases}$$

Mając końce średnicy całego drzewa $\alpha[v_0], \beta[v_0]$, bez kłopotu możemy wyznaczyć jej środek, przeszukując drzewo. Zatem wyznaczenie wartości d_A, v_A i u_A dla ustalonego poddrzewa T_A zajmie czas $O(n)$.

Tak więc złożoność czasowa całego rozwiązania to $O(n^2)$. Implementacja tego typu rozwiązania znajduje się w pliku `mods1.cpp`.

Rozwiązanie liniowe

Zauważmy, że powyższy algorytm wyznaczający średnicę w drzewie wyznacza również średnice niektórych poddrzew tego drzewa. Zmodyfikujemy go teraz tak, by wyznaczał średnice *wszystkich* poddrzew w sumarycznym czasie $O(n)$, dzięki czemu uzyskamy algorytm o złożoności liniowej.

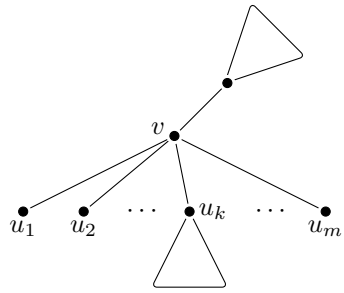
Ukorzeńmy nasze początkowe drzewo w pewnym wierzchołku v_0 i wykonajmy powyższy algorytm oparty na programowaniu dynamicznym. Rozpatrzmy dowolną krawędź drzewa jako kandydata do usunięcia; niech to będzie krawędź łącząca pewien węzeł v z jego ojcem. Usunięcie tej krawędzi podzieli nam drzewo na poddrzewo T_A (poddrzewo zaczepione w węźle v) oraz poddrzewo T_B (reszta drzewa zawierająca ojca węzła v; nazwiemy ją *dopełnieniem* poddrzewa v). Zauważmy, że $d_A = d[v]$, pozostaje zatem obliczyć średnicę poddrzewa T_B. Zrobimy to znowu programowaniem dynamicznym. Tym razem jednak zaczniemy od korzenia i będziemy poruszać się w dół drzewa, licząc wartości: $D[v]$ – średnica dopełnienia poddrzewa v oraz $H[v]$ – długość najdłuższej ścieżki w dopełnieniu poddrzewa v zaczynającej się w ojcu v.

Ponieważ w przypadku korzenia v_0 dopełnienie poddrzewa v_0 nie istnieje (bo nie istnieje krawędź do ojca), możemy przyjąć $H[v_0] = D[v_0] = -\infty$. W ogólnym przypadku węzła v, dla którego obliczyliśmy już wartości $D[v]$ i $H[v]$, następująca rekurencja pozwala wyznaczyć wartości dla jego synów:

$$H[u_k] = \max\big(0,\ 1 + H[v],\ \max_{i\neq k}(1 + h[u_i])\big),$$

$$D[u_k] = \max\big(D[v],\ \max_{i\neq k} d[u_i],\ H[u_k],\ \max_{i\neq k}(H[v] + 2 + h[u_i]),\ \max_{i\neq j\neq k\neq i}(h[u_i] + 2 + h[u_j])\big).$$

Poprawność wzoru na $D[u_k]$ wynika z faktu, że średnica w dopełnieniu poddrzewa u_k może albo nie przechodzić przez węzeł v (i być w całości zawarta w dopełnieniu poddrzewa v lub w całości zawarta w poddrzewie jednego z synów węzła v), albo kończyć się w węźle v, albo przechodzić przez węzeł v (i przechodzić przez ojca v lub nie).



Całość obliczeń dla węzła v o m synach można zaimplementować w czasie $O(m)$. Przykładowo, aby szybko wyznaczać składnik $\max_{i\neq j\neq k\neq i}(h[u_i]+2+h[u_j])$, wystarczy na początku w czasie $O(m)$ wyznaczyć *trzy* indeksy i_1, i_2, i_3, które odpowiadają największym trzem elementom $h[u_{i_1}]$, $h[u_{i_2}]$ oraz $h[u_{i_3}]$, a następnie dla kolejnych wartości k w czasie stałym wybierać największe dwa ze zbioru $\{i_1, i_2, i_3\} \setminus \{k\}$.

Pozostaje wyznaczyć końce i środki średnic. O ile jednak potrzebowaliśmy znać długości średnic dla wszystkich poddrzew, to końców i środków potrzebujemy jedynie dla dwóch poddrzew z optymalnego podziału. Możemy więc je wyznaczyć, korzystając z algorytmu opisanego w poprzednim rozdziale.

W ten sposób otrzymujemy rozwiązanie wzorcowe, działające w czasie liniowym. Przykładową implementację można znaleźć w pliku `mod.cpp`.