Jakub Radoszewski
Treść zadania, Opracowanie

Dawid Dąbrowski, Karol Pokorski
Program

Dostępna pamięć: 64 MB.

OI, etap I, 15.10–12.11.2012

Cennik

Najpopularniejszym środkiem transportu w Bajtocji od zawsze była kolej. Spośród n miast tego kraju, m par miast jest połączonych odcinkami torów należącymi do Bajtockich Kolei Państwowych (BKP). Odcinki nie krzyżują się nigdzie poza miastami i mogą prowadzić tunelami lub mostami. Koszt przejazdu między dwoma miastami połączonymi bezpośrednio odcinkiem torów jest zawsze taki sam i wynosi a bajtalarów.

Obecnie sytuacja na rynku usług komunikacyjnych w Bajtocji uległa zmianie. Pojawiła się konkurencja dla BKP – powstały Bajtockie Linie Lotnicze (BLL). BLL zamierzają uruchomić połączenia lotnicze między niektórymi parami miast. Ponieważ jazda bajtocką koleją jest bardzo wygodna, zarząd BLL postanowił uruchamiać połączenia lotnicze tylko między takimi parami miast, dla których **nie** *istnieje bezpośrednie połączenie kolejowe. Ze względów ekonomicznych, BLL utworzy połączenia lotnicze jedynie między tymi parami miast, dla których najtańsze połączenie kolejowe wymaga dokładnie jednej przesiadki. Koszt biletu lotniczego na jedno połączenie będzie stały i równy b bajtalarów.*

Żeby pomóc mieszkańcom Bajtocji w planowaniu podróży, Ministerstwo Transportu Bajtocji (MTB) postanowiło stworzyć cenniki zawierające koszty najtańszych tras między poszczególnymi miastami kraju. Trasę rozumiemy tu jako sekwencję złożoną z dowolnej liczby pojedynczych połączeń kolejowych lub lotniczych. Zadanie stworzenia cenników przypadło w udziale Bajtazarowi, który pracuje jako urzędnik w MTB. Czy pomógłbyś mu w napisaniu programu, który wyznaczy odpowiednie cenniki?

Dodajmy dla jasności, że wszystkie połączenia kolejowe i lotnicze w Bajtocji są dwukierunkowe.

Wejście

Pierwszy wiersz standardowego wejścia zawiera pięć liczb całkowitych n, m, k, a oraz b ($2 \leqslant n \leqslant 100\,000$, $1 \leqslant m \leqslant 100\,000$, $1 \leqslant k \leqslant n$, $1 \leqslant a, b \leqslant 1000$) pooddzielanych pojedynczymi odstępami. Liczby n oraz m oznaczają odpowiednio liczbę miast oraz liczbę połączeń kolejowych w Bajtocji. Dla uproszczenia miasta w Bajtocji numerujemy od 1 do n. Kolejne liczby w wierszu oznaczają: k – numer miasta początkowego, dla którego należy wygenerować cennik opisujący najtańsze trasy; a – koszt biletu na jedno połączenie kolejowe; b – koszt biletu na jedno połączenie lotnicze.

Każdy z kolejnych m wierszy zawiera dwie liczby całkowite u_i oraz v_i ($1 \leqslant u_i, v_i \leqslant n$, $u_i \neq v_i$ dla $i = 1, 2, \ldots, m$) oddzielone pojedynczym odstępem, oznaczające numery miast połączonych bezpośrednim odcinkiem torów.

Możesz założyć, że z miasta numer k można dojechać koleją do wszystkich pozostałych miast kraju.

W testach wartych łącznie 30% punktów zachodzą dodatkowe warunki $n \leqslant 700$ oraz $m \leqslant 700$.

Wyjście

Twój program powinien wypisać na standardowe wyjście n wierszy. Wiersz o numerze i (dla $i = 1, 2, \ldots, n$) powinien zawierać jedną liczbę całkowitą: koszt najtańszej trasy z miasta numer k do miasta numer i. Spośród tych wierszy, wiersz o numerze k powinien zawierać liczbę 0.

Przykład

Dla danych wejściowych:	*poprawnym wynikiem jest:*
`5 5 1 3 2`	`0`
`1 2`	`3`
`2 3`	`3`
`3 4`	`2`
`4 5`	`5`
`3 1`	

Wyjaśnienie do przykładu: *Najtańsza trasa z miasta numer 1 do miasta numer 5 wiedzie przez miasto numer 3 lub miasto numer 4. W obu przypadkach składa się ona z jednego połączenia kolejowego i jednego lotniczego.*

Rozwiązanie

Problem opisany w tym zadaniu ma naturalną interpretację grafową. Oznaczmy przez $G = (V, E)$ graf nieskierowany, w którym wierzchołki odpowiadają miastom ($|V| = n$), a krawędzie – połączeniom kolejowym ($|E| = m$). Niech dalej $G' = (V, E')$ oznacza graf reprezentujący połączenia lotnicze (oznaczmy $m' = |E'|$). Zbiór E' można całkiem zgrabnie opisać, korzystając z pojęcia *kwadratu* grafu. Kwadratem grafu $G = (V, E)$ nazywamy graf $G^2 = (V, E^2)$, w którym dwa wierzchołki są połączone krawędzią, jeśli w grafie G istnieje między nimi ścieżka o długości 2, tzn.:

$$E^2 = \{uw : uv, vw \in E \text{ dla pewnego } v \in V\}.$$

Krawędzie w grafie G' łączą wierzchołki położone w *odległości* dokładnie 2 w grafie G, więc $E' = E^2 \setminus E$. Celem zadania jest wyznaczenie długości najkrótszych ścieżek z ustalonego wierzchołka $k \in V$ w ważonym grafie $G'' = (V, E \cup E')$, w którym krawędzie ze zbioru E mają wagę a, a krawędzie ze zbioru E' – wagę b. Zakładamy dla uproszczenia, że same grafy G i G' nie są ważone.

Do rozwiązania zadania możemy zastosować algorytm Dijkstry zaimplementowany z pomocą kopca zupełnego, który pozwala wyznaczyć długości najkrótszych ścieżek ze źródła (tj. wierzchołka k) w czasie $O((n + m + m') \log n)$. Aby stwierdzić, na ile dobre jest to rozwiązanie, należy odpowiedzieć na pytanie, jak duże może być m'. Krawędziami ze zbioru E' łączymy tylko bliskie sobie wierzchołki w grafie G, a takich par wierzchołków w grafie intuicyjnie nie powinno być zbyt wiele... Niestety, istnieją grafy rzadkie (tj. spełniające warunek $m = O(n)$), w których wszystkie wierzchołki są położone stosunkowo blisko siebie. Przykładem może tu być *gwiazda*, czyli drzewo,

w którym $n-1$ wierzchołków jest podłączonych bezpośrednio do jednego, centralnego wierzchołka. Jeśli G jest taką gwiazdą, to graf G'' stanowi klikę (graf pełny) i wówczas $m' = \Theta(n^2)$. Nasz algorytm działa zatem pesymistycznie w czasie $\Theta(n^2 \log n)$. W takim przypadku już lepiej byłoby wykorzystać prostszą wersję algorytmu Dijkstry, w której do implementacji kolejki priorytetowej stosuje się tablicę przechowującą aktualne wagi wierzchołków. Wówczas złożoność czasowa spada do $O(n^2)$, jednak wciąż nie jest ona zadowalająca. Zgodnie z warunkiem z treści zadania, opisane tu rozwiązania mogły zdobyć 30% punktów. Program zawierający tego typu rozwiązanie można znaleźć w pliku `cens1.cpp`.

Najkrótsze ścieżki

Aby uzyskać bardziej efektywne rozwiązanie, zamiast od razu uruchamiać algorytm Dijkstry warto skorzystać ze specjalnej struktury grafu G''. Niech k będzie źródłem i niech $v \in V$ będzie dowolnym wierzchołkiem grafu. Zastanówmy się, jak może wyglądać najkrótsza ścieżka z k do v w grafie G''.

Na początek warto rozpatrzyć przypadek, w którym w ogóle nie opłaca się nam używać połączeń lotniczych. Jest tak, gdy koszty biletów spełniają nierówność $b \geqslant 2a$. Wówczas szukana najkrótsza ścieżka odpowiada po prostu najkrótszej ścieżce z k do v w grafie G. Długości takich ścieżek do wszystkich $v \in V$ możemy wyznaczyć w czasie $O(n + m)$ za pomocą algorytmu BFS (przeszukiwanie wszerz), więc ten przypadek możemy uznać za rozpatrzony.

Odtąd założymy, że $b < 2a$. Oznaczmy przez x długość najkrótszej ścieżki z k do v w grafie G, przez y – długość najkrótszej takiej ścieżki w grafie G', a przez z – szukaną długość najkrótszej ścieżki w ważonym grafie G''. Dalsza część rozwiązania zależy od parzystości liczby x. Rozważymy dwa przypadki, opisane w poniższych lematach.

Lemat 1. Jeśli $2 \mid x$, to $y = \frac{x}{2}$ i w konsekwencji $z = y \cdot b$.

Dowód: Uzasadnijmy najpierw, dlaczego w tym przypadku zachodzi $y = \frac{x}{2}$. Niech $P = (u_0, u_1, \ldots, u_x)$ będzie najkrótszą ścieżką łączącą wierzchołki $u_0 = k$ i $u_x = v$. Żadna para wierzchołków u_i, u_{i+2} nie jest połączona krawędzią w grafie G, gdyż wówczas w grafie G istniałaby ścieżka z k do v krótsza niż P. Tak więc ścieżka P wyznacza ścieżkę $P' = (u_0, u_2, \ldots, u_{x-2}, u_x)$ z k do v w grafie G' o długości $\frac{x}{2}$. Z drugiej strony, gdyby w grafie G' istniała ścieżka z k do v o długości mniejszej niż $\frac{x}{2}$, to wyznaczałaby ona ścieżkę w grafie G o długości mniejszej niż x, a założyliśmy, że takiej ścieżki nie ma.

Uzasadniliśmy już pierwszą część tezy lematu i tym samym skonstruowaliśmy ścieżkę z k do v w grafie G'' o długości $y \cdot b$. Wiemy też, że ani najkrótsza ścieżka w grafie G, ani najkrótsza ścieżka w grafie G' przeniesiona do grafu G'' nie ma kosztu mniejszego niż $y \cdot b$. W grafie G'' mogłaby teoretycznie istnieć ścieżka o mniejszej długości, złożona z krawędzi pochodzących zarówno z G, jak i z G'. Taka ścieżka, zawierająca p krawędzi z G i q krawędzi z G', odpowiadałaby ścieżce o długości $p+2q$ w grafie G. Jednak $p + 2q \geqslant x$, więc taka ścieżka miałaby w grafie G'' długość

$$p \cdot a + q \cdot b = \tfrac{1}{2}(p \cdot 2a + 2q \cdot b) > \tfrac{1}{2}b(p + 2q) \geqslant \tfrac{1}{2}b \cdot x = y \cdot b,$$

czyli nie byłaby krótsza niż ścieżka P'. ■

Lemat 2. Jeśli $2 \nmid x$, to $z = \min(\frac{x-1}{2} \cdot b + a,\ y \cdot b)$.

Dowód: Niech P będzie najkrótszą ścieżką z k do v w grafie G, a P' – najkrótszą taką ścieżką w grafie G'. Teza lematu orzeka, że najkrótsza ścieżka z k do v w grafie G'' to albo ścieżka P, na której pary kolejnych krawędzi zamieniono na krawędzie z G' (ostatnią krawędź ścieżki zostawiamy bez zmian), albo ścieżka P'. Wykażemy, że każda inna ścieżka w grafie G'' ma wagę nie mniejszą od tych dwóch wymienionych.

Niech $P'' = (u_0, \ldots, u_d)$ będzie najkrótszą ścieżką z $u_0 = k$ do $u_d = v$ w grafie G''. Jeśli jest więcej niż jedna taka ścieżka, wybieramy tę, która zawiera minimalną liczbę krawędzi pochodzących z G, a jeśli wciąż mamy wybór, wybieramy ścieżkę, w której pierwsza krawędź pochodząca z G występuje możliwie najpóźniej. Uzasadnimy, że P'' albo nie zawiera żadnych krawędzi z G, albo zawiera dokładnie jedną, położoną na samym końcu. To już nam wystarczy do wykazania tezy lematu: jeśli P'' nie zawiera żadnych krawędzi pochodzących z grafu G, to jest nie krótsza niż P', a w przeciwnym razie jej długość, wyrażona w krawędziach grafu G, jest nie mniejsza niż x, więc jej faktyczna długość jest nie mniejsza niż długość przekształconej w opisany powyżej sposób ścieżki P.

Dowód struktury ścieżki P'' przeprowadzimy w dwóch nietrudnych krokach. Przede wszystkim, P'' na pewno nie zawiera dwóch krawędzi z G występujących pod rząd. Faktycznie, gdyby pewne dwie krawędzie u_iu_{i+1}, $u_{i+1}u_{i+2}$ należały do E, wówczas albo $u_iu_{i+2} \in E$, albo $u_iu_{i+2} \notin E$, więc $u_iu_{i+2} \in E'$. W każdym z przypadków opłacałoby się nam zastąpić obie te krawędzie, o łącznej wadze $2a$, krawędzią u_iu_{i+2} o wadze a lub o wadze b.

Jeśli teraz P'' nie zawiera krawędzi pochodzących z G, to nie mamy już czego dowodzić. Załóżmy więc, że P'' zawiera krawędź z G i nie jest to ostatnia krawędź ścieżki. Niech $u_iu_{i+1} \in E$ będzie pierwszą taką krawędzią, mamy $u_{i+1}u_{i+2} \in E'$. Istnieje wówczas wierzchołek $w \in V$, taki że $u_{i+1}w, wu_{i+2} \in E$. Zastąpmy rozważaną parę krawędzi z P'' parą krawędzi u_iw, wu_{i+2}. Podobnie jak poprzednio, mamy $u_iw \in E'$ albo $u_iw \in E$. W pierwszym przypadku długość P'' nie zmieniła się, ale krawędź pochodząca z G znalazła się dalej na ścieżce. W drugim zaś przypadku również parę krawędzi u_iw, wu_{i+2} możemy zastąpić jedną krawędzią u_iu_{i+2} należącą do E albo do E', co powoduje skrócenie ścieżki P''. Widzimy zatem, że żadna z tych sytuacji nie była możliwa, i krawędź z E może wystąpić jedynie na końcu ścieżki P''. ■

Algorytm

Aby dokończyć rozwiązanie, wystarczy dla każdego wierzchołka $v \in V$ obliczyć długość najkrótszej ścieżki z k do v w grafie G oraz w grafie G'. Najkrótsze ścieżki w grafie G obliczamy wspomnianym już algorytmem BFS, jedyna kwestia to najkrótsze ścieżki w grafie G'.

Zauważmy, że gdyby w zadaniu należało znaleźć najkrótsze ścieżki w grafie $G^2 = (V, E^2)$, to rozwiązanie również byłoby proste. Mógłby to być algorytm BFS zastosowany dla grafu G, w którym dla każdego wierzchołka pamiętalibyśmy długości dwóch najkrótszych ścieżek z k: ścieżki o długości parzystej oraz ścieżki o długości nieparzystej (można by też wykonać przeszukiwanie BFS w nieznacznie przekształconym grafie G, co opisano w opracowaniu zadania *Morskie opowieści* w tej książeczce).

Naszym grafem jest jednak $G' = (V, E^2 \setminus E)$, co istotnie utrudnia sprawę. Algorytm w tym przypadku będzie podobny, lecz nieco bardziej subtelny.

W naszym przeszukiwaniu BFS będą występować dwa rodzaje stanów: (v) oraz (v, u). Pierwszy typ stanu reprezentuje wierzchołek v, do którego możemy dotrzeć z k za pomocą sekwencji krawędzi z grafu G'. Drugi typ stanu wskazuje na wierzchołek v, taki że istnieje wierzchołek u, do którego możemy dotrzeć z k za pomocą sekwencji krawędzi z grafu G', oraz w grafie G istnieje krawędź z u do v. Z pierwszego rodzaju stanów przechodzimy do drugiego rodzaju i na odwrót. Do przechodzenia grafu będziemy wykorzystywali tylko krawędzie ze zbioru E.

Stanem początkowym jest stan (k). Będąc w stanie postaci (v), rozważamy wszystkie krawędzie $vw \in E$ i odwiedzamy stany (w, v). Sumarycznie, dla wszystkich stanów tego rodzaju zajmie to czas $O(m)$.

Będąc w stanie postaci (v, u), rozważamy wszystkie krawędzie $vw \in E$ i dla każdej z nich:

(1) Jeśli istnieje krawędź uw, to ignorujemy krawędź vw (tj. nie możemy nią pójść).

(2) Jeśli nie istnieje krawędź uw, to idziemy krawędzią vw do stanu (w) i **usuwamy** skierowaną krawędź vw (ale tylko ze zbioru krawędzi rozpatrywanych przy stanach drugiego rodzaju!). Możemy to zrobić, bo wykonujemy przeszukiwanie wszerz, czyli w momencie przejścia do stanu (w) mamy już obliczoną długość najkrótszej ścieżki z (k) do (w), tak więc ponowne przechodzenie krawędzią vw tego wyniku nie poprawi (a moglibyśmy ją próbować ponownie odwiedzać dla innych stanów (v, u'), $u' \neq u$).

Ponieważ w przypadku (2) zawsze usuwamy jakąś krawędź z grafu w jednym z kierunków, więc znajdziemy się w nim łącznie co najwyżej $2m$ razy. Mogłoby się jednak wydawać, że z powodu przypadku (1) koszt algorytmu jest zbyt duży. Dokładniejsze oszacowanie pokazuje, że tak nie jest. Ustalmy wierzchołek v i niech $deg(v)$ oznacza jego stopień, czyli liczbę krawędzi z nim incydentnych. Wówczas liczba sytuacji, w których znajdziemy się w przypadku (1) dla ustalonego v, z jednej strony nie przekracza liczby wszystkich krawędzi $uw \in E$, czyli m, a z drugiej strony – liczby par krawędzi $uv, vw \in E$, czyli $deg(v)^2$. Sumaryczny koszt przypadku (1) szacuje się zatem przez:

$$\sum_{v \in V} \min(deg(v)^2, m) \leqslant \sum_{v \in V} \sqrt{deg(v)^2 \cdot m} = \sum_{v \in V} deg(v)\sqrt{m} = O(m\sqrt{m}).$$

W tym oszacowaniu skorzystaliśmy z nierówności między minimum z dwóch liczb a ich średnią geometryczną.

Podsumowując, złożoność czasowa całego rozwiązania to $O(m\sqrt{m})$, a złożoność pamięciowa jest liniowa. Stany drugiego rodzaju możemy przechowywać w tablicy z haszowaniem. Implementacje rozwiązania wzorcowego można znaleźć w plikach `cen.cpp`, `cen1.pas` i `cen2.cpp`. Natomiast w plikach `cens2.cpp` i `cens3.pas` znajdują się rozwiązania wolniejsze, w których nie usuwamy wykorzystanych krawędzi w kroku (2). Tego typu rozwiązania uzyskiwały na zawodach ok. 50% punktów (złożoność czasową takich rozwiązań można oszacować jako $O(\sum_{v \in V} deg(v)^2) = O(\sum_{v \in V} n \cdot deg(v)) = O(nm)$).

Testy

Testy są podzielone na pięć grup.

Testy z grupy a zawierają przypadek $b < a$. W testach tych najkrótsza ścieżka złożona z połączeń kolejowych ze źródła do pewnego wierzchołka ma długość nieparzystą, natomiast w optymalnym rozwiązaniu korzystamy jedynie z połączeń lotniczych. Testy te składają się z cyklu nieparzystej długości zawierającego źródło, do którego doczepiona jest duża gwiazda.

Testy z grupy b zawierają przypadek $a < b < 2a$. Wyglądają one podobnie jak testy z grupy a.

W testach z grupy c występuje dużo sytuacji, w których istnieją wszystkie krawędzie uv, vw, uw. Są szczególnie niewygodne dla rozwiązań rozważających graf E^2 zamiast $E^2 \setminus E$.

Testy z grupy d mają na celu odsiać rozwiązania, w których pesymistyczny czas działania algorytmu BFS lub Dijkstry, z powodu błędu w oznaczaniu wierzchołków, jest wykładniczy. Liczba możliwych ścieżek prowadzących ze źródła do poszczególnych wierzchołków jest wykładnicza.

Testy z grupy e są strukturalnie podobne do testów z grup a, b, ale parametry generatora zostały dobrane w celu uwypuklenia różnic czasowych programów (w szczególności, odsiewają programy nieusuwające krawędzi).