Karol Cwalina
Treść zadania, Opracowanie

Szymon Wąsik
Program

OI, Etap II, dzień pierwszy, 15.02.2006

Metro

W pewnym mieście od długiego czasu zmagano się z budową metra. Przy tym źle gospodarowano środkami, nie doszacowano kosztów budowy i zapomniano przewidzieć pieniądze na zakup pociągów. W rezultacie zbudowano wiele stacji, ale wydrążono tylko część zaplanowanych tuneli — ledwie wystarczających do tego, żeby pomiędzy każdymi dwiema stacjami istniała możliwość przejazdu. Liczba tuneli jest o 1 mniejsza od liczby zbudowanych stacji, ponadto wszystkie tunele są dwukierunkowe. Za pozostałe środki udało się kupić zaledwie kilka pociągów.

Chcąc ratować twarz, dyrekcja metra zwróciła się do Ciebie z prośbą o opracowanie tras pociągów w taki sposób, by możliwie najwięcej stacji znalazło się na trasach linii metra. Każdy pociąg musi jeździć po ustalonej trasie. Trasy muszą być proste, tzn. nie mogą się rozgałęziać (żadne trzy tunele zbiegające się na jednej stacji nie mogą jednocześnie leżeć na tej samej trasie). Kilka tras może natomiast przebiegać przez tę samą stację lub ten sam tunel.

Zadanie

Zadanie polega na napisaniu programu, który:

- *wczyta ze standardowego wejścia opis sieci tuneli oraz liczbę tras pociągów metra, które należy zaplanować;*
- *obliczy maksymalną liczbę stacji, jakie mogą się znaleźć na wymaganej liczbie tras metra;*
- *zapisze wynik na standardowe wyjście.*

Wejście

W pierwszym wierszu wejścia zapisane są dwie liczby całkowite n i l ($2 \leqslant n \leqslant 1\,000\,000$, $0 \leqslant l \leqslant n$) oddzielone pojedynczym odstępem. Liczba n to liczba stacji, a l to liczba tras pociągów, które należy zaplanować. Stacje są ponumerowane od 1 do n.

W każdym z kolejnych $n-1$ wierszy znajdują się po dwie różne liczby całkowite oddzielone pojedynczym odstępem. Liczby $1 \leqslant a_i, b_i \leqslant n$ znajdujące się w $i+1$-szym wierszu są numerami stacji połączonych przez i-ty tunel.

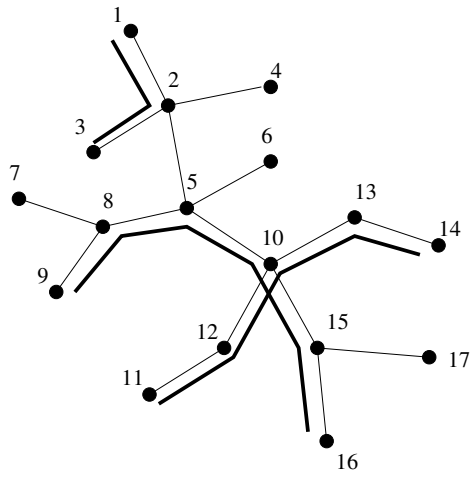
Wyjście

W pierwszym i jedynym wierszu wyjścia należy zapisać jedną liczbę całkowitą równa maksymalnej liczbie stacji, jakie mogą znaleźć się na trasach pociągów.

Przykład

Dla danych wejściowych:

```
17 3
1 2
3 2
2 4
5 2
5 6
5 8
7 8
9 8
5 10
10 13
13 14
10 12
12 11
15 17
15 16
15 10
```



poprawnym wynikiem jest:

```
13
```

Na rysunku przedstawiono sieć tuneli z zaznaczonymi trasami metra w jednym z optymalnych układów.

Rozwiązanie

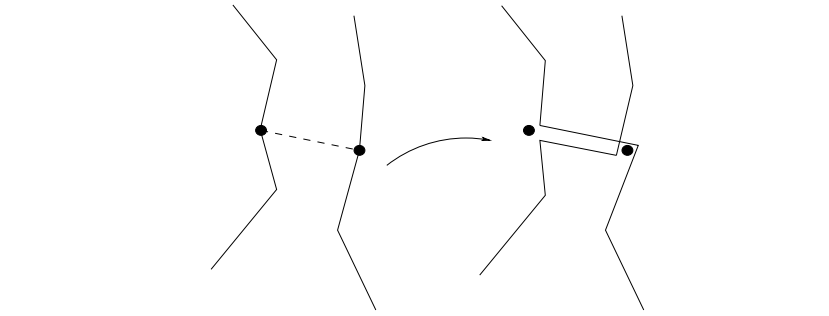
Rozwiązanie

Oczywiście sieć tuneli tworzy *drzewo*[1]. Naszym zadaniem jest znalezienie jego optymalnego (zawierającego maksymalną liczbę wierzchołków) *podgrafu*, który można zbudować z l ścieżek. Zauważmy, że możemy ograniczyć się do poszukiwania *poddrzewa* o tej własności. Wynika to stąd, że jeśli mamy rozwiązanie optymalne, które nie jest spójnym podgrafem drzewa tuneli, to potrafimy je łatwo do takiej postaci przekształcić, nie zmniejszając liczby stacji obsługiwanych przez metro. Wystarczy splatać po dwie trasy tak, jak jest to przedstawione na rysunku 1, aż do uzyskania spójności grafu.

A zatem w n wierzchołkowym drzewie mamy znaleźć maksymalnie liczne poddrzewo, które można pokryć siecią l linii metra. Wydaje się intuicyjnym, że takie pokrycie jest możliwe w przypadku każdego poddrzewa o $2l$ liściach — aby się o tym przekonać, wykażemy następujące

Twierdzenie 1 (Zasadnicze Twierdzenie o Budowie Metra) *Każde drzewo o co najwyżej $2l$ liściach da się pokryć l liniami metra.*

[1] Jest to wniosek z powszechnie znanej charakteryzacji drzewa jako spójnego grafu, w którym liczba krawędzi jest o 1 mniejsza od liczby wierzchołków.



Rys. 1: Splecenie ścieżek

Dowód Na początek przyjmijmy definicję: **skrzyżowaniem** nazywamy każdy węzeł drzewa mający co najmniej trzech sąsiadów.

Aby udowodnić twierdzenie posłużymy się indukcją ze względu na liczbę liści w drzewie.

Jeśli są w nim co najwyżej dwa liście, twierdzenie jest oczywiście prawdziwe. W przeciwnym przypadku w drzewie występuje przynajmniej jedno skrzyżowanie.

Jeśli jest ono tylko jedno, to drzewo ma postać gwiazdy: nie więcej niż $2l$ ścieżek rozchodzi się promieniście z jednego węzła. W takiej sytuacji l linii metra niewątpliwie wystarczy.

Gdy z kolei w drzewie są co najmniej dwa skrzyżowania, jedną z linii metra zaplanujmy tak, by zawierała dwa liście i przynajmniej dwa skrzyżowania. Zmodyfikujmy też drzewo poprzez wyrzucenie z niego kawałków linii metra, łączących liście wybrane do linii z najbliższymi im skrzyżowaniami (w ten sposób usuwamy część wierzchołków i krawędzi, które na pewno zostają pokryte przez wybraną linię metra).

W efekcie liczba liści zmniejszyła się o dwa (w miejscu skrzyżowań nie powstały liście!), co pozwala zastosować założenie indukcyjne. ■

Tym samym zadanie sprowadza się do znalezienia maksymalnie licznego poddrzewa o co najwyżej $2l$ liściach w danym n wierzchołkowym drzewie.

Rozwiązanie autorskie

Ponieważ wybór podzbioru liści drzewa jednoznacznie określa pewne jego poddrzewo, aby znaleźć żądane poddrzewo możemy wybrać jeden jego liść, a następnie dobrać $2l - 1$ kolejnych liści. Równoważnie możemy myśleć o dobraniu $2l - 1$ ścieżek, prowadzących do liści i odchodzących od już wybranego fragmentu poddrzewa (początkowo jest nim wybrany liść, a później także wszystkie węzły leżące na dobranych ścieżkach).

Każdą ścieżkę w drzewie, której jedynym węzłem należącym do wybranego poddrzewa jest jej koniec, będziemy dalej nazywać **odrostem** tego **poddrzewa**.

W przypadku, gdy $l = 1$, wystarczy znaleźć najdłuższą ścieżkę odchodzącą od wybranego liścia, z tym, że wcześniej jako startowy liść trzeba wybrać koniec pewnej spośród najdłuższych ścieżek w drzewie.

Okazuje się, że tę metodę można zastosować także dla $l > 1$ i poprawny jest następujący algorytm:

1: S := {koniec pewnej najdłuższej ścieżki w drzewie};

2: **for** $i := 1$ **to** $\min(|\{\text{liście}\}|-1, 2l-1)$ **do**
3: $\quad S := S \cup \{\text{najdłuższy odrost poddrzewa } S\}$;
4: **return** S;

Aby przekonać się o poprawności powyższej metody rozważmy dwa układy linii metra w naszym drzewie: niech P_{alg} będzie układem stworzonym przez algorytm, zaś P pewnym układem optymalnym, przy czym układy będziemy utożsamiać ze zbiorami ich węzłów.

Niech O będzie pierwszym w kolejności odrostem wybranym przez algorytm takim, że $O \not\subset P$:

- jeśli $O \cap P = \emptyset$, to dowolną ścieżkę O' w układzie P, łączącą liść z najbliższym mu skrzyżowaniem w P, która nie jest zawarta w P_{alg}, można zastąpić odrostem O — taka ścieżka istnieje, bo P jest optymalny i różny od P_{alg}, więc $P \not\subset P_{alg}$. Z definicji O jako najdłuższego odrostu mamy $|O| \geqslant |O'|$;
- jeśli $O \cap P \neq \emptyset$, to istnieje ostatni w stronę liścia O węzeł, wspólny dla O i P. Wówczas dowolną wychodzącą z tego węzła ścieżkę należącą do P można zastąpić przez odpowiedni fragment odrostu O. Z definicji O fragment ten nie jest krótszy niż owa ścieżka.

W obu przypadkach znaleźliśmy inne rozwiązanie optymalne P', które zawiera odrost O oraz wszystkie odrosty wcześniej znalezione przez algorytm.

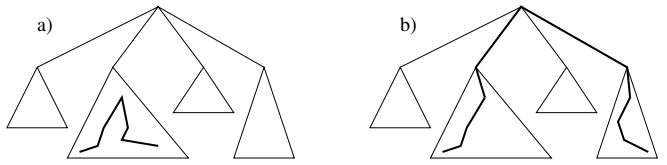
Po co najwyżej $2l-1$ krokach powyższego rozumowania znajdziemy rozwiązanie optymalne P_{opt}, zawierające wszystkie odrosty, czyli takie, że $P_{alg} \subset P_{opt}$. Ponieważ rozwiązanie to zawiera wszystkie liście danego drzewa zawarte w P_{alg}, a P_{alg} zawiera maksymalną możliwą liczbę liści, więc $P_{alg} = P_{opt}$.

Szczegóły implementacyjne

Pozostaje tylko wspomnieć o tym, jak znaleźć koniec którejś z najdłuższych ścieżek oraz jak znajdować najdłuższe odrosty.

Z pierwszym z zadań można uporać się za pomocą przeszukiwania w głąb. Dla ustalenia uwagi wybierzmy dowolny wierzchołek drzewa i nazwijmy go **korzeniem** drzewa. W drzewie ukorzenionym możemy dla każdego węzła zdefiniować jego wysokość — jest to wysokość poddrzewa, którego ten węzeł jest korzeniem. Dla każdego węzła v o wysokości h istnieje ścieżka tej długości łącząca v z jednym z liści jego poddrzewa — ścieżkę tę nazwiemy **pniem** w poddrzewie wierzchołka v. Najdłuższa ścieżka w drzewie:

- albo w całości jest zawarta w którymś z poddrzew korzenia (rys. 2a);
- albo jest zbudowana z korzenia i z pni dwóch poddrzew korzenia o największej wysokości, ewentualnie jednego pnia, jeśli korzeń ma tylko jedno poddrzewo (rys. 2b).



Rys. 2: Położenia najdłuższej ścieżki względem korzenia drzewa

Z kolei znajdowanie najdłuższych odrostów można sprowadzić do wybierania węzłów o największej wysokości — najdłuższy odrost to pień niewybranego dotychczas do metra węzła o największej wysokości wraz z krawędzią łączącą ten węzeł z pokrytym już fragmentem metra.

Poczynione powyżej spostrzeżenia prowadzą do następującego algorytmu:

```
1:  S := {koniec pewnej najdłuższej ścieżki w drzewie};
2:  ukorzeń drzewo w wybranym liściu;
3:  wyznacz wysokości wszystkich węzłów oraz pień dla każdego z nich;
4:  Kol := {sąsiad wybranego liścia};
5:  for i := 1 to min(|{liście}|-1,2l-1) do begin
6:      v := element Kol o największej wysokości;
7:      S := S∪{pień dla węzła v};
8:      Kol := Kol∪{sąsiedzi węzłów z pnia węzła v}-{v};
9:  end;
10: return S;
```

Wykorzystując kopiec do reprezentacji kolejki priorytetowej *Kol*, można otrzymać implementację powyższego algorytmu, działającą w czasie $O(n \log n)$.

Można jednak wyznaczone wysokości posortować zawczasu w czasie liniowym za pomocą sortowania kubełkowego, a następnie przejrzeć węzły w kolejności malejących wysokości, co prowadzi do poniższego rozwiązania liniowego.

```
1:  S := {koniec pewnej najdłuższej ścieżki w drzewie};
2:  ukorzeń drzewo w wybranym liściu;
3:  wyznacz wysokości wszystkich węzłów oraz pień dla każdego z nich;
4:  posortuj węzły względem ich wysokości;
5:  ile_liści := 1;
6:  while ile_liści<min(|{liście}|,2l) do begin
7:      v := kolejny węzeł w kolejności malejących wysokości;
8:      if v∈S then continue;
9:      S := S∪{pień węzła v};
10:     ile_liści := ile_liści +1;
11: end;
12: return S;
```

Zaprezentowany powyżej algorytm był podstawą dla rozwiązania, które przyjęto jako wzorcowe podczas zawodów drugiego etapu: jego kod znajduje się w plikach `met.cpp` oraz `met1.pas`.

Rozwiązanie wzorcowe

Zaprezentujemy teraz rozwiązanie, które nie zostało wybrane jako wzorcowe na czas drugiego etapu Olimpiady tylko dlatego, iż nie było ono wtedy znane twórcy niniejszego opracowania. Ze względu jednak na swoją elegancję i prostotę, zasługuje na miano rozwiązania wzorcowego.

Definicja 1 Na początek przyjmijmy następującą definicję *warstwy*:

- liście drzewa tworzą warstwę pierwszą;
- węzły z nieokreślonej jeszcze warstwy, których wszyscy sąsiedzi, prócz co najwyżej jednego, należą do warstw o numerach $\leqslant j-1$, tworzą warstwę numer j.

Podamy teraz dwa proste fakty o warstwach.

Fakt 1 Ścieżki wychodzące z każdego węzła, w prawie każdym kierunku (prócz być może jednego) są malejące w sensie numerów warstw ich węzłów.

Fakt 2 Z każdego węzła warstwy j wychodzi malejąca (w powyższym sensie) ścieżka długości j.

Fakt 1 wynika z tego, że dla każdego węzła co najwyżej jeden sąsiad nie należy do warstwy o niższym numerze. Fakt 2 zaś z tego, że każdy węzeł wewnętrzny ma sąsiada, który należy do warstwy o numerze o jeden mniejszym.

Zauważmy, że na każdej linii metra są co najwyżej 2 węzły z każdej warstwy: gdyby bowiem znalazły się trzy węzły z jednej warstwy, otrzymalibyśmy sprzeczność z faktem 1 dla tego z węzłów, który znajduje się pomiędzy pozostałymi dwoma. Tym samym liczba stacji obsługiwanych przez l linii metra jest niewiększa niż

$$\sum_{j=1}^{\infty} min(2l, |\{\text{warstwa nr j}\}|).$$

Okazuje się, że szukana maksymalna liczba obsługiwanych stacji jest równa tej sumie.

Podobnie jak w przypadku rozwiązania autorskiego, główną trudność stanowi pokazanie poprawności proponowanego rozwiązania.

Jeśli liczba liści drzewa jest $\leqslant 2l$, to na mocy Zasadniczego Twierdzenia o Budowie Metra można zaproponować układ linii, pokrywający całe drzewo, czyli podany wzór jest poprawny.

Załóżmy teraz, że liczba liści jest większa niż $2l$. Niech i będzie najmniejszym numerem warstwy o mocy $\leqslant 2l$:

- każdemu węzłowi warstwy i-tej przypiszmy jeden węzeł z warstwy $(i-1)$-szej;
- z pozostałych węzłów warstwy $(i-1)$-szej wybierzmy dowolne tak, by razem wybrać z tej warstwy $2l$ węzłów.

Na mocy faktu 2 wybranym węzłom z $(i-1)$-szej warstwy odpowiadają parami rozłączne ścieżki długości $(i-1)$. Razem z elementami warstw o numerach $\geqslant i$ tworzą one poddrzewo o $2l$ liściach, które na mocy Zasadniczego Twierdzenia o Budowie Metra można pokryć l liniami. Z drugiej strony liczba węzłów tego drzewa jest równa ograniczeniu górnemu, określonemu przez wzór, co dowodzi poprawności proponowanego algorytmu.

A oto jego pseudokod:

```
1: wynik := 0;
2: warstwa := {liście};
3: while warstwa ≠ ∅ do begin
```

```
4:      wynik := wynik + min(2l,warstwa);
5:      nowa_warstwa := ∅;
6:      for v ∈ warstwa do begin
7:          w := nieodwiedzony jeszcze sąsiad v;
8:          w.ilu_sąsiadów := w.ilu_sąsiadów−1;
9:          if w.ilu_sąsiadów = 1 then
10:             nowa_warstwa := nowa_warstwa ∪ {w};
11:     end
12:     warstwa := nowa_warstwa;
13: end;
14: return wynik;
```

O wyższości rozwiązania wzorcowego nad autorskim

Przedstawione powyżej rozumowanie powstało z inspiracji rozwiązaniami stworzonymi w czasie zawodów przez niektórych olimpijczyków.

Dzięki swojej prostocie pozwala ono na stworzenie bardzo szybkiego programu: najszybsze rozwiązanie stworzone przez zawodników było implementacją właśnie tego algorytmu i było szybsze od implementacji rozwiązania autorskiego.

Kolejną widoczną jego zaletą jest dużo mniejsze zapotrzebowanie na pamięć, gdyż nie ma konieczności wykonywania pamięciożernego rekurencyjnego przeszukiwania w głąb, potrzebnego dla znalezienia pierwszego węzła.

Dla przykładu, w zapisanej w pliku `met3.cpp` implementacji algorytmu wzorcowego są wykorzystane zaledwie trzy n-elementowe tablice liczb 32-bitowych i kilka zmiennych. Dzięki trickowemu określaniu nieodwiedzonego sąsiada węzła v jest ona istotnie szybsza zarówno od implementacji rozwiązania autorskiego, jak i od rozwiązań zawodników.

Inne rozwiązania

Mając w pamięci opisane powyżej rozwiązania, można dostrzec przynajmniej dwie kategorie „innych rozwiązań”: mniej efektywne implementacje poprawnych algorytmów oraz niepoprawne heurystyki, będące jedynie ich przybliżeniami.

Rozwiązania mniej efektywne

Do tej grupy możemy zaliczyć implementację rozwiązania autorskiego, wykorzystującą kolejkę priorytetową — odpowiedni program jest zapisany w pliku `met2.c`. Rozwiązanie to działa w czasie $O(n \log n)$.

Inne rozwiązanie o nieoptymalnej złożoności zapisane jest w pliku `mets1.cpp`. W programie tym posłużono się wielokrotnym przeszukiwaniem wszerz w celu znalezienia najdłuższej ścieżki w drzewie, co zaowocowało algorytmem działającym w czasie $O(n^2)$.

Rozwiązania niepoprawne

Przykładem rozwiązania niepoprawnego może być prosta heurystyka zakładająca, że wszystkie węzły wewnętrzne zostaną pokryte przez linie metra. Przypuszczenie to prowadzi do formuły $|\{\text{węzły wewnętrzne}\}| + \min(2l, |\{\text{liście}\}|)$, która wykazuje nieprzypadkowe podobieństwo do wzoru opisującego rozwiązanie wzorcowe.

Oprócz tego istnieje zapewne wiele innych rozwiązań niepoprawnych, ale ich poszukiwanie nie wydaje się być wartościową formą spędzania czasu. Dlatego też poprzestaniemy na tym jednym.

Testy

Do oceny rozwiązań zawodników użyto następującego zestawu testów:

Nazwa	n	l	wynik
met1a.in	15	2	11
met1b.in	15	3	13
met1c.in	2	2	2
met1d.in	19	2	12
met2a.in	13	3	13
met2b.in	2	0	0
met2c.in	16	2	13
met3.in	15	2	13
met4a.in	2821	600	2610
met4b.in	5656	704	4230

Nazwa	n	l	wynik
met5a.in	33375	240	31616
met5b.in	70023	9996	55099
met6a.in	52500	1400	52341
met6b.in	119008	14101	86156
met7.in	200000	47000	193969
met8a.in	315001	70000	297228
met8b.in	874641	123456	680918
met9.in	980001	240000	969882
met10.in	991520	1200	991432
met11a.in	1000000	30000	342789
met11b.in	1000000	2	1000000

A oto krótka charakteryzacja poszczególnych grup testów:

- `met[1-3].in` to proste testy poprawnościowe,
- `met[4,5].in` to średniej wielkości testy, na których przestają sobie radzić rozwiązania kwadratowe,
- `met[6-8].in` to duże testy wydajnościowe, zaliczane przez rozwiązania liniowe,
- `met[9-11].in` to testy o maksymalnych i prawie maksymalnych rozmiarach, sprawdzające pewne niuanse, jak na przykład możliwość utrzymywania zbyt dużej liczby zmiennych rekurencyjnych w procedurze wyznaczającej najdłuższą ścieżkę.