Algorytm znajdujący trzy wierzchołki o maksymalnej sumie odległości - zadanie 8 lista 2

Dominik Szczepaniak

1 Wprowadzenie

Rozważamy problem znalezienia trzech wierzchołków w drzewie, dla których suma odległości między tymi wierzchołkami jest maksymalna.

2 Algorytm

Algorytm rozwiązujący ten problem składa się z następujących kroków:

1. Znalezienie średnicy drzewa
2. Uruchomienie przeszukiwania BFS z jednego z końców średnicy.
3. Wybranie wierzchołka, który ma największą odległość średnicy. Znajdujemy to tak, że zaczynamy liczyć odległości w BFS jeśli nasz wierzchołek nie leży na średnicy (zapisujemy w jakiejś tablicy np. sciezka[] wartość true, jeśli wierzchołek leży na średnicy i false jeśli nie leży, później robiąc krok w BFS jeśli odległość jest równa 0 to sprawdzamy czy sciezka[wierzcholek] == false, jeśli tak to zwiększamy odległość w kroku BFS. Jeśli odległość nie jest równa 0, to znaczy, że zeszliśmy już ze ścieżki między końcami średnicy, więc z własności drzewa wiemy, że się tam nie cofniemy, stąd możemy pominąć sprawdzanie (ale to taki szczegół implementacyjny, złożonościowo sprawdzanie tego warunku nie dodaje nam zbyt dużo)).

3 Dowód

Lemat:
W optymalnym rozwiązaniu dwa wierzchołki muszą być końcami średnicy.

Proof. Jeśli średnica znajduje się w rozwiązaniu, to mamy ilość jej krawędzi (niech będzie to d).

Jeśli usuniemy średnicę z grafu to pozostanie nam las (pozostałe niespójne drzewa). Niech drzewo z największą głębokością zostanie oznaczone jako L1, z głębokością k, a drugii drzewo z największą głębokością niech zostanie oznaczone przez L2 i głębokość l. Wynik naszego algorytmu to d+k+1 (+1 bo łączymy las ze średnicą).

W takim razie zakładając nie wprost, że istnieje lepszy wynik chcemy znaleźć wynik większy niż d + k + 1.

Ponieważ maksymalizujemy wynik, w takim razie inny algorytm weźmie najdłuższy odcinek który nie jest średnicą. Czyli weźmiemy odcinek z najgłębszego drzewa (L1) do drugiego najgłębszego drzewa (L2) (bo ten odcinek jest najdłuższy). Niech odległość między tymi drzewami będzie oznaczona przez p (czyli p jest częścią średnicy łączącą te drzewa oraz wejściem do nich ze średnicy). Wiemy, że l < k (bo nie mogą >= średnicy po dodaniu odległości między nimi). Musimy też założyć, że l + k + p < d, ponieważ my nie chcemy mieć końców średnicy (wtedy mogłoby być l+k+p=d). Ponieważ nie chcemy mieć obu końców średnicy to ostatni wybór wierzchołka znowu musi być w jakimś drzewie pozostałym po usunięciu średnicy lub do jednego z końców średnicy. Te przypadki:

a) Jest w jakimś drzewie (oznaczmy je L3, a jego głębokość y). Mamy y < l (z tego samego argumentu co wcześniej dla l i k). Niech ilość krawędzi którą dodamy będzie równa x. Wiemy znowu, że odległość od tej głębokości do dowolnej innej głębokości wybranych przez nas drzew będzie krótsza od średnicy.

Rozważmy przypadki jak mogą występować drzewa:
- L3 L2 L1
Wtedy wynik to k + p + l + x + y.
Wiemy, że k + p-1 + x + y < d (odległość między L1 i L3 (p-1 ponieważ nie wchodzimy do L2, x to robi)) - z założenia.
Mamy więc k+p-1+l+x+y < d + l - 1 < d + k + 1
Sprzeczność.

- L2 L3 L1
Wtedy x = 0
l + k + p + y + 1 (+1 bo wejście do L3) < d + y + 1 < d+k+1
Sprzeczność.

Pozostałe przypadki rozwiązujemy tymi samymi argumentami co powyżej. Dla wszystkich sprzeczność.

b) Idzie do jednego z końców średnicy, ten koniec oznaczmy jako K.
Czyli mamy najgłębsze drzewo, drugie najgłębsze drzewo oraz dłuższy odcinek do średnicy, oznaczmy go przez z.
W takim razie mamy odcinki k + p + l + z.

Musimy założyć, że k + z < d, ponieważ jeśli byłoby równe to wybralibyśmy dwa końce najdłuższej ścieżki, czego nie możemy zrobić. Mamy więc k + l + z.

Znowu rozważmy przypadki
- K L1 L2
Wtedy liczba krawędzi z L2 do K < d oraz dochodzi jeszcze liczba krawędzi równa k+1 (+1 bo łączymy las do ścieżki z L2 do K). No ale to dalej jest mniej niz d + k + 1.
Sprzeczność.

- K L2 L1
Wtedy liczba krawędzi z L1 do K < d oraz dochodzi liczba krawędzi równa l+1. No ale to dalej mniej niż d + k + 1.
Sprzeczność.

Musimy jeszcze rozpatrzeć co jeśli wszystkie wierzchołki znajdują się w jednym lesie, ale tu możemy zastosować "indukcję" - ponownie znajdujemy średnicę i robimy nasze argumenty - to nam da, że optymalny wynik musi zawierać średnicę tego lasu, no ale jeśli zawiera średnicę tego lasu oraz najgłębsze drzewo tego lasu po usunięciu średnicy, to nasza średnica z całego drzewa (wejściowego) jest większa i głębokość tego drzewa jest większa niż głębokość najgłębszego poddrzewa tego drzewa.

We wszystkich przypadkach mamy sprzeczność, więc średnica musi należeć do optymalnego wyniku.

□

W takim razie jeśli średnica znajduje się w optymalnym wyniku, to nasz algorytm bierze tę średnicę, a później sprawdza wszystkie możliwe odpowiedzi i wybiera tę maksymalną, stąd musi dawać optymalny wynik

4 Podsumowanie

Udowodniliśmy, że nasz algorytm daje optymalny wynik.
To, że algorytm się kończy jest oczywiste.

Złożoność: O(n+m), gdzie n = |V|, m = |E|. Najpierw wykonujemy BFS ze złożonością O(n+m), następnie kolejny z tą samą złożonościa, a następnie kolejny z tą samą złożonością, więc mamy złożoność O(n+m).