lista 3 zadanie 4a)

Wiktor Hamberger
308982

21 kwietnia 2020

Rozwiązanie tego zadania będzie opierało się na dwóch faktach. Po pierwsze, skoro mowa o drzewie, liczba krawędzi grafu jest równa liczbie jego wierzchołków - 1, więc złożoność przeszukiwania tego drzewa wszerz redukuje się: $O(|V| + |E|) = O(|V| + |V| - 1) = O(|V|)$. Po drugie w drzewie istnieje dokładnie jedna droga pomiędzy dwoma wierzchołkami, więc znalezienie pomiędzy wierzchołkami drogi długości innej niż C, sprawia, że możemy już nie rozpatrywać danej pary wierzchołków jako kandydatów do zwiększenia rozwiązania.

Algorytm to puszczony w każdym wierzchołku drzewa lekko zmodyfikowany BFS, który zwraca liczbę dróg długości C zaczynających się w wierzchołku startowym. Modyfikacja polega na zauważeniu, że gdy dojdziemy do wierzchołka oddalonego od wierzchołka startowego o więcej niż C, to nie opłaca się przeszukiwać drzewa w tę stronę dalej. Zauważmy, że po wykonaniu takiego BFS-a w każdym wierzchołku i zsumowaniu wyników otrzymamy dwukrotność odpowiedzi, ponieważ każdą drogę policzyliśmy dwa razy – z v_1 do v_2 i z v_2 do v_1, dlatego wynik należy podzielić przez 2.

```
Data: G = {V, E} - graf, wag - funkcja wagowa, odw[] - tablica
      odwiedzonych
Result: Liczba par wierzchołków odległych od siebie o C
Function fun(C) is
    foreach v in V(G) do
        wyzeruj odw[]
        wynik+=myBFS(v, C)
    end
    return wynik/2
end
Function myBFS(v, C) is
    niech Q będzie kolejką
    Q.enqueue(pair(v, 0))
    wynik=0
    while Q is not empty do
        (u, dist)=Q.dequeue()
        odw[u] = true
        if dist>C then
            continue
        end
        if dist==C then
            wynik++
            continue
        end
        foreach w in G.sąsiedzi(u) do
            if odw[w]==false then
                Q.enqueue((w, dist+wag(u, w)))
            end
        end
    end
    return wynik
end
```

Pozostaje kwestia poprawności i złożoności. BFS dla danego wierzchołka sprawdzi każdy inny wierzchołek oddalony o nie więcej niż C, a jako że zostanie puszczony w każdym wierzchołku, rozpatrzy wszystkie, potencjalne odległe od siebie o C, pary wierzchołków. Nie pominie więc żadnej możliwości. Każdą parę wierzchołków, oddalonych od siebie o co najmniej C, algorytm rozpatrzy dokładnie 2 razy, po jednym razie z każdego wierzchołka i jako że istnieje pomiędzy nimi tylko jedna droga, za każdym razem albo dwa razy ją policzy (a później przy zwracaniu wyniku podzieli na 2), albo nic nie zmieni. Stąd mamy pewność, że algorytm nie weźmie żadnej nadliczbowej drogi. Przy obliczaniu złożoności skorzystam z faktu przytoczonego w pierwszym akapicie. W naszym przypadku Przeszukiwanie drzewa wszerz działa w czasie $O(|V|)$ i to przeszukiwanie zostanie wykonane dokładnie raz dla każdego wierzchołka, to sumaryczna złożoność czasowa tego algorytmu to $O(|V|^2)$