AiSD ćwiczenia lista 2

Oskar Bujacz

10 kwietnia 2020

Zadanie 6

Ułóż algorytm, który dla danego spójnego grafu G oraz krawędzi e sprawdza w czasie $O(n+m)$, czy krawedź e należy do jakiegoś minimalnego drzewa spinającego grafy G. Możesz założyć, że wszytskie wagi krawędzi są różne.

Rozwiązanie

Wprowadźmy oznaczenia u, v na dowolne dwa wierzchołki oraz e na dowolną krawedź. Oznaczmy minimalne drzewo spinającego jako MST. Warto zauwżyć kilka istotnych faktów:

- jeśli krawedź e należy do każdej ścieżki z u do v to e jest mostem, zatem należy do MST z jego definicji.
- jeśli krawedź e ma najwiekszą wagę w jednym cyklu to nie należy do MST - dowodzi tego poprawność algorytmu Kruskali. Dzieje się tak dlatego, że algorytm uruchamiany na takim grafie w ostatniej kolejności w tym cyklu bedzię rozważał krawędż e i jej nie doda do MST.
- w przypadku jaki mamy w zadaniu tj wszystkie wagi krawędzie w grafie spójnym są różne, to istnieje tylko jedno MST dla tego grafu. Jest to znany fakt z wykładu.

Lemat 1. *Cycle property - Krawędź e należy do przynajmniej jednego MST, wtedy i tylko wtedy gdy, to e nie ma największej wagi na żadnym cyklu.*

Dowód przeprowadzimy go przez implikację w dwie strony.

- $\Leftarrow$ Dowód nie wprost. Załóżmy, że e nie ma największej wagi na żadnym z cykli w grafie oraz, że $e \notin MST$. Rozważmy dwie możliwości:
  - e nie należy do żadnego cyklu. Ale wtedy $e \in MST$, zatem $\lightning$.
  - e należy do co najmniej jednego cyklu. Rozważmy dowolne MST i dołóżmy do niego krawędź e. Powstanie wtedy cykl, bo $e \notin MST$. Skoro e nie ma największej wagi, to istnieje krawedź e', która ma większą wagę. Zatem możemy ściągnąć e' i otrzymamy MST o mniejszej wadze $\lightning$.
- $\Rightarrow$ przeprowadzimy dowód przez kontrapozycję. Załóżmy, że e ma maksymalną wagę na każdym cyklu oraz nie należy do żadnego MST. Nie wprost załóżmy, że należy do pewnego MST. Ale wtedy możemy zdjąć ją z MST i wymienić na krawedź lżejszą z pewnego cyklu $\lightning$.

Algorytm
Znając powyższe obserwację konstruujemy algorytm będący modyfikacja DFS-a. Na wejściu podajemy krawędź $e = (u, v)$ do sprawdzenia przynalażności do MST oraz tablicę wag krawędzi.

```
check_if_in_MST(e, u, weight[]):
        e = weight[e]
        visited[] = zeros(#V)
        remove e from G
        zgodnie z DFS przechodzimy po G od u:
                x = aktualny wierzcholek
                visited[x] = 1
                e' = nowa krawedź
                if w[e'] < w[e]:
                        przejdż do kolejnego nieodwiedzonego wierzchołka
                else: pomiń ta krawędź
        if visited[v] == 1:
                return false #gdy nie należy do MST
        if visited[v] == 0:
                retrun true # gdy należy do MST
```

Dowód poprawności algorytmu
Do rozważenia mamy dwa przypadki przynależności do MST:

- $e \in MST$
  - e jest mostem - wtedy usunięcie e z grafu podzieli go na dwie spójne składowe, zatem nie będzie możliwe odwiedzenie v, algorytm zwróci wartość *true*.
  - e w żadnym z cykli, do których należy, nie jest maksymalna - wtedy po usunięciu e algorytm nie będzie mógł dotrzeć do v, bo w każdym cykli musi natrafić na cięższą krawedź.
- $e \notin MST$ - oznacza to, że e jest najcięższa na jakimś cyklu, zatem algorytm przechodząc od u po tym cyklu dojdzie to wierzchołka v, zostanie zwrócona wartość $false$

Złożoność
Wiemy, że złożoność DFS to $O(n + m)$, zatem taka też jest złożoność algorytmu.