Lista 2 zadanie 6

Krystian Grabowski

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

Aby stworzyć algorytm, posłużę się własnością cyklu znaną z Matematyki dyskretnej. Jeśli w jakimkolwiek cyklu istniejącym w grafie G jakaś krawędz k jest krawędzią o maksymalnej wadze na tym cyklu, to k nie należy do żadnego MST. Analogicznie formułujemy zależność na potrzeby zadania.

Cycle property
Jeśli e nie jest maksymalną wagowo krawędzią na żadnym cyklu występującym w G, to e należy do jakiegoś minimalnego drzewa spinającego.

D-d Cycle property
Załóżmy, że e nie jest maksymalną wagowo krawędzią na żadnym cyklu występującym w G, a mimo to nie należy do żadnego minimalnego drzewa spinającego. Wtedy mamy dwa przypadki:

1. e nie należy do żadnego cyklu z G.

   Skoro więc e nie leży na żadnym cyklu, to wiemy, że od pierwszego wierzchołka który łączy do drugiego nie istnieje inna droga niż przez e. Stąd e musi należeć do MST, ponieważ bez niej graf nie byłby spójny.

2. e należy do jakiegoś cyklu z G

   Weźmy więc powstałe MST i dołóżmy do niego naszą krawędź e. Powstał w ten sposób cykl. Należy teraz zauważyć, że na żadnym cyklu z G e nie było maksymalną wagowo krawędzią. Istnieje więc jakieś e', które ma większą wagę niż e w tym cyklu. Jeśli usuniemy e' to otrzymamy MST o mniejszej wadze, co przeczy temu, że powstałe drzewo było MST.

Więc e musi należeć do MST w każdym przypadku. Znając tą własność możemy w prosty sposób stworzyć algorytm sprawdzający czy e należy do jakiegoś minimalnego drzewa spinającego. Wystarczy sprawdzić czy e jest maksymalną krawędzią na jakimś cyklu w G. Jeśli nie jest, to znaczy, że będzie należeć do jakiegoś MST.

Algorytm

Na wejściu dostaniemy jakąś reprezentację grafu oraz naszą krawędź e. Zapamiętujemy więc wagę naszej krawędzi, nazwijmy tę wagę w, oraz dwa wierzchołki, które łączyła $v1$ i $v2$. Usuwamy krawędź z grafu, a następnie ustalając jako wierzchołek startowy $v1$, rozpoczynamy przeszukiwanie grafu za pomocą zmodyfikowanego algorytmu BFS. Przechodzimy po krawędzi jedynie wtedy, gdy jej waga jest mniejsza od w. Jeśli waga jest większa, to ignorujemy krawędź. Jeśli algorytm zakończy działanie nie odwiedzając $v2$, zwracamy "TAK". Jeśli natomiast natknie się na $v2$, zwracamy "NIE".

Uzasadnienie działania algorytmu

W przypadku gdy nie dotarliśmy do v2, mamy dwa przypadki.

1. Usunięcie e rozspójniło graf. Wtedy nasze e musiało należeć do MST, ponieważ pomiędzy wierzchołkami które łączy nie istnieje w grafie żadna inna ścieżka.
2. e nie była maksymalna na żadnym cyklu z G. Na każdym takim cyklu zostaliśmy zatrzymani przez krawędź o większej wadze niż e.

W przypadku gdy dotarliśmy do $v2$

1. Istniał co najmniej 1 cykl, w którym e był krawędzią z maksymalną wagą spośród krawędzi tego cyklu. Z cycle property wiemy, że w takim przypadku e nie może należeć do żadnego MST.

Pseudokod

E-zbiór krawędzi z G

V-zbiór wierzchołków z G

Dla każdej krawędzi k k.start i k.end to wierzchołki, które łączy k, a k.weight to jej waga.

```
E.remove(e)
queue = []
visited = [] #musimy ustawić domyślnie False
queue.push(e.start)
while(queue not empty):
    v = queue.pop()
    for all edges u - (v, g):
        if (u.weight < e.weight):
            if not visited[g]:
                queue.push(k)
                visited[g] = True
            if (g == v2):
                return False
return True
```

Złożoność
Czasowa: $O(n+m)$
Pamięciowa: $O(n)$

Złożoność czasowa to złożoność zmodyfikowanego algorytmu BFS, który w najgorszym przypadku zachowa się jak zwykły BFS, który działa w czasie liczba krawędzi + liczba wierzchołków. Każdy wierzchołek oraz krawędź zostaną odwiedzone co najwyżej tylko jeden raz. Możemy w różny sposób trzymać informacje o odwiedzeniu. Jeśli ponumerujemy wierzchołki jednym z rozwiązań możne być użycie zwykłej tablicy i sprawdzanie $visited[g]$.

Złożoność pamięciowa zależy jedynie od kolejki oraz zbioru informacji czy dany wierzchołek został odwiedzony. W obu przypadkach pamiętamy maksymalnie tyle informacji, ile jest wierzchołków w grafie.