Lista 02 zadanie 06

Wiktor Hamberger
308982

6 kwietnia 2020

W rozwiązaniu tego zadania przyda się pewien lemat.

Lemat: jeżeli krawędź e nie jest maksymalna na żadnym cyklu z G to e należy do MST, jeżeli natomiast e jest maksymalną krawędzią na jakimkolwiek cyklu, to e nie należy do żadnego MST.

Dowód Weźmy krawędź e. Mamy trzy możliwości.

1. e nie należy do żadnego cyklu. Wtedy musi należeć do MST.

2. e należy do cyklu (możliwe, że więcej niż do jednego), ale nie jest maksymalną krawędzią na żadnym. Załóżmy nie wprost, że e nie należy do żadnego MST, weźmy więc dowolne MST, nazwijmy je T_1, i dodajmy do niego e. Powstał nam wtedy cykl zawierający jakąś krawędź maksymalną, nazwijmy ją f, oraz e. Z założenia wiemy że $e < f$, więc możemy usunąć f z tego cyklu i otrzymamy nowe drzewo, o mniejszej wadze niż T_1 oraz pokrywające wszystkie wierzchołki, więc T_1 nie było MST. Mamy więc sprzeczność.

3. e należy do cyklu, nazwijmy go C, i jest w nim maksymalną krawędzią. Załóżmy nie wprost, że e należy do MST T_2. Usuńmy e z tego drzewa, co rozdzieli T_2 na dwa poddrzewa. Istnieje wtedy w C taka inna krawędź, która ma swoje końce w obu poddrzewach oraz jest lżejsza od e. W ten sposób otrzymaliśmy nowe drzewo, o mniejszej wadze niż T_2 oraz pokrywające wszystkie wierzchołki, więc T_2 nie było MST. Mamy więc sprzeczność.

Z tą wiedzą możemy przejść do algorytmu.

Data:
G[x] - graf jako lista sąsiedztwa; G[x][i].first to numer wierzchołka i-tego sąsiada x, G[x][i].second to waga krawędzi pomiędzy x i G[x][i].first;
odw[x] - tablica odwiedzonych wierzchołków, używana przez DFS;
u, v - wierzchołki, które łączy krawędź krawędź *e* z zadania;
e - waga krawędzi *e* z zadania;
Result:
true jeżeli krawędź *e* należy do jakiegoś MST, **false** w przeciwnym przypadku

```
Function fun(int u, int v, int e) is
    G[u].delete(v);
    G[v].delete(u);
    myDfs(e, u);
    return odw[v] ? false : true;
end
Function myDFS(int e, int u) is
    odw[u]:=true;
    for i:=0 to G[u].size() do
        if odw[G[u][i].first] == true and G[u][i].second<e then
            myDFS(e, G[u][i].first);
        end
    end
end
```

Algorytm ten usuwa krawędź e z grafu, zapamiętuje jej wagę i próbuje dojść z jednego końca tej krawędzi do drugiego, chodząc tylko po krawędziach o mniejszej wadze niż e. Na koniec działania algorytmu możemy mieć dwie możliwości:

1. **Doszliśmy do drugiego końca krawędzi.** Wtedy krawędź e nie może być częścią żadnego MST, ponieważ istnieje w grafie cykl w którym krawędź e ma największą wagę (lemat).

2. **Nie doszliśmy do drugiego końca krawędzi.** Wtedy krawędź e musi być częścią jakiegoś MST, ponieważ albo nie należy do żadnego cyklu, albo nigdy nie jest maksymalną krawędzią w cyklu (lemat).

Pozostała jeszcze kwestia złożoności. Algorytm to lekko zmodyfikowany DFS. Algorytm ten nie zapętli się (tablica odw[]) i odwiedzi każdy wierzchołek maksymalnie raz. Do tego używamy listy sąsiedztwa jako reprezentacji grafu, więc ten algorytm działa w czasie O(n+m).