Algorytmy i struktury danych
Lista 2. Zadanie 6

26 czerwca 2020

Lemat 1. *Krawędź e należy do jakiegoś MST wtedy i tylko wtedy, gdy nie jest krawędzią o największej wadze w żadnym cyklu.*

Dowód. $\Rightarrow$ Nie wprost. Załóżmy, że e należy do jakiegoś MST i istnieje cykl, w którym ma największą wagę. Weźmy MST zawierające e. Usuńmy z niego krawędź e, otrzymamy dwie składowe spójne. Zauważmy, że któraś krawędź z cyklu, w którym e ma największą wagę, uspójnia z powrotem graf (końce krawędzi e są w oddzielnych różnych spójnych składowych, a istnieją między nimi dwie drogi na tym cyklu). Otrzymaliśmy MST o mniejszej wadze. Sprzeczność.

$\Leftarrow$ Nie wprost. Załóżmy, że e nie jest krawędzią o największej wadze w żadnym cyklu, ale nie należy do żadnego MST. Rozpatrzmy dwa przypadki.

1° Krawędż e nie należy do żadnego cyklu.
To oznacza, że krawędź e jest mostem w grafie G, czyli musi należeć do każdego MST. Sprzeczność.

2° Krawędź e należy do jakiegoś cyklu.
Rozważmy MST niezawierające e. Dołóżmy e do MST. Dostaliśmy cykl. Wiemy, że e nie ma największej wagi w tym cyklu. Niech e' będzie krawędzią o największej wadze w tym cyklu. Zauważmy, że usunięcie e' spowoduje, że otrzymamy inne drzewo o mniejszej wadze niż nasze MST (bo $C(e) < C(e')$. Sprzeczność.

□

Z lematu wynika, że jeżeli w G nie znajdziemy cyklu zawierającego krawędź e o pozostałych krawędziach o wadze mniejszej niż $C(e)$, to e należy do jakiegoś MST. Niech e łączy wierzchołki u i v. Chcemy znaleźć ścieżkę łączącą wierzchołki u i v składającą się z krawędzi o mniejszej wadze. Możemy to zrealizować tworząc graf G' składający się tylko z krawędzi o mniejszej wadze niż $C(e)$. W grafie G' algorytmem DFS sprawdzimy, czy wierzchołki u i v są w jednej składowej spójnej.

Algorithm 1 DFS w G'

for $e' \in E(G)$ **do**
    if $C(e') < C(e)$ **then**
        $G' = G' \cup e'$
$S \leftarrow pustystos$
S.push(v)
while ! S.empty() **do**
    $u \leftarrow$ S.pop()
    for $x \in$ N(u) **do**
        if $x == w$ **then**
            return false
        if $x == nieodwiedzony$ **then**
            $u \leftarrow odwiedzony$
            S.push(x)

Zauważmy, że stworzenie grafu G' wymaga m porównań. Natomiast DFS wykonuje się w $O(n+m)$ (mamy z reguły mniej krawędzi w grafie G' niż w G). Zatem złożoność całego algorytmu to $O(n+m)$.