AiSD

Lista 2.

Zadanie 6

Maksymilian Perduta 308080

7 kwietnia 2020

1 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.

2 Cycle property i dowód poprawności.

Korzystamy z faktu znanego jako **Cycle property** tj.

Lemat 1. *Dla dowolnego cyklu C w grafie, jeżeli waga krawędzi e należącej do C jest większa niż waga każdej innej krawędzi w tym cyklu, wtedy krawędź e nie może należeć do **MST***

Dowód. Dowód nie wprost. Załóżmy, że krawędź e należy do pewnego MST T_1. Wtedy usunięcie krawędzi e spowoduje podział T_1 na dwa poddrzewa, każde z jednym końcem krawędzi e. Skoro jednak e należało do cyklu C to istnieje także krawędź f w C, o mniejszej wadze, która ponownie połączy te dwa poddrzewa. W taki sposób powstanie MST T_2 o sumie wag mniejszej niż T_1, więc T_1 nie było MST. Sprzeczność. □

Korzystając z tego faktu musimy jeszcze pokazać, że:

Lemat 2. *Jeżeli krawędź e nie jest maksymalna na żadnym cyklu w G, to należy do pewnego MST.*

Dowód. Ponownie zakładamy nie wprost, że e nie jest maksymalna i nie należy do MST. Wtedy:

• e nie leży na żadnym cyklu.
Wtedy e musi należeć do MST, więc sprzeczność.

• e jest częścią pewnego cyklu G (ale nie jest maksymalna).
Weźmy pewne MST T_1 i dodajmy do niego krawędź e. Dodanie krawędzi do drzewa zawsze tworzy nam jakiś cykl. Wiemy z założenia oraz Lematu 1, że na tym cyklu istnieje krawędź f o wadze większej niż e (które oczywiście też należy do tego cyklu). Usunięcie krawędzi f spowoduje powstanie nowego MST T_2 o sumie wag mniejszej niż T_1. To znaczy, że T_1 nie było MST, więc mamy sprzeczność. □

3 Idea rozwiązania.

Koszystamy z Lematu 1 i Lematu 2. Wczytujemy krawędź e i zapamiętujemy jej wagę oraz końce. Teraz przechodzimy graf, zaczynając od jednego końca krawędzi e (np. algorytmem BFS lub DFS), ale w taki sposób, że nie korzystamy z krawędzi o większej wadze niż waga e. W ten sposób nasze e będzie zawsze maksymalne na każdym cyklu w takim grafie. Teraz mamy dwie możliwe sytuacje:

• Jeżeli uda nam się dojść w ten sposób do drugiego końca krawędzi e, to znaczy, że krawędź e była częścią cyklu, w którym była maksymalną krawędzią, czyli (z Lematu 1) nie należy do MST.

• Jeżeli jednak nie udało się dojść do drugiego końca krawędzi e, to znaczy, że albo graf G się rozspójnił, albo e nie było maksymalne na żadnym cyklu w G (żeby nie dało się dojść do drugiego końca e na każdej ścieżce do niego prowadzącej musiała istnieć krawędź o większej wadze niż e, żeby algorytm przechodzący po grafie jej nie wykorzystał). Z Lematu 2 wiemy, że w obu tych sytuacjach e musi należeć do MST.

4 Pseudokod i złożoność.

Dane:
G – graf pamiętający dla każdego wierzchołka jego sąsiadów(.first) i wagę krawędzi między nimi(.second).

vis – tablica odwiedzonych wierzchołków
e – krawędź o którą pytamy
p – wierzchołek początkowy e
k – wierzchołek końcowy e
w – waga krawędzi e
W zależności od sposobu zapamiętania e; może to być na przykład para par gdzie p zapisalibyśmy jako $e.first$, a k i w analogicznie.
Pseudokod:

$isMST(e)$:
$G[p].delete(k)$;
$G[k].delete(p)$;
$lessDFS(p, w)$;
if $vis[k]$ **then**
  return false;
end if
return true;

(1)

$lessDFS(p, w)$
$vis[p] \leftarrow true$;
for $0 \leq i < G[p].size()$ **do**
  if $!(vis[G[p][i].first])$ **and** $G[p][i].second < w$ **then**
    $lessDFS(G[p][i].first, w)$;
  end if
end for

W algorytmie korzystamy z lekko zmienionego algorytmu **DFS**. Przechodzimy po wszytkich wierzchołkach i wszystkich krawędziach, więc złożoność tego algorytmu będzie równa $O(n + m)$, gdzie n to liczba wierzchołków, a m to liczba krawędzi. Jako że odwiedzamy wierzchołki tylko raz, algorytm nigdy się nie zapętli.