Algorytmy i struktury danych

Lista 1 zadanie 4

Kamil Fedzin 310408

2 kwietnia 2020

1 Treść zadania

4. (1pkt) Niech u i v będą dwoma wierzchołkami w grafie nieskierowanym $G = (V, E; c)$, gdzie c : E $\rightarrow R_+$ jest funkcją wagową. Mówimy, że droga $u = u_1, u_2, ..., u_{k-1}, u_k = v$ z u do v jest sensowna, jeśli dla każdego $i = 2, ..., k$ istnieje droga z u_i do v krótsza od każdej drogi z u_{i-1} do v (przez długość drogi rozumiemy sumę wag jej krawędzi).
Ułóż algorytm, który dla danego G oraz wierzchołków u i v wyznaczy liczbę sensownych dróg z u do v.

2 Idea rozwiązania

Naszym zadaniem jest wyznaczenie liczby sensownych dróg z u do v, a sensowną drogą określamy taką, w której przejście do każdego kolejnego wierzchołka przybliża nas do wierzchołka v. Chcielibyśmy mieć zatem możliwość sprawdzenia, czy po dojściu do następnego wierzchołka jesteśmy bliżej, czy dalej od wierzchołka v. Do obliczenia długości drogi do wierzchołka v z każdego innego wierzchołka posłuży nam **algorytm Dijkstry**, dzięki któremu będziemy znali najkrótszą możliwą odległość każdego wierzchołka od wierzchołka v. Następnie będziemy chcieli wyruszyć z wierzchołka początkowego u, który również ma już ustaloną odległość do v i sprawdzać wierzchołki z nim sąsiadujące. Jeżeli z któregoś z tych wierzchołków odległość do v jest krótsza niż odległość w aktualnym wierzchołku, to znaczy że możemy do tego wierzchołka dojść i będziemy bliżej wierzchołka końcowego v, czyli droga którą przeszliśmy była częścią **drogi sensownej**. Wystarczy się teraz zastanowić, jak chcemy policzyć ilość dróg sensownych od wierzchołka u. Mój pomysł jest taki, aby posortować wierzchołki od najmniejszej odległości od v, a następnie dla każdego wierzchołka ustalonego według sortowania wywołać funkcję liczącą drogi sensowne do danego wierzchołka od wierzchołka v(liczbę dróg z wierzchołka v ustawiamy na 1 i do ilości dróg dla danego wierzchołka dodajemy ilości dróg wierzchołków, z którymi sąsiaduje i dla których odległość od v jest mniejsza). Takim oto sposobem będziemy otrzymywać ilość dróg sensownych dla kolejnych wierzchołków z najmniejszą odległością od v, a gdy dojdziemy do wierzchołka u i obliczymy dla niego ilość dróg sensownych, otrzymamy nasz wynik.

3 Pseudokod

$drogi[v] \leftarrow 1;$
$Dijkstra(v);$
sort(wierzchołki z G po odległości od wierzchołka v policzonej za pomocą Dijkstry w tablicy dist);
dla każdego kolejnego wierzchołka(poza V z odległością 0) z posortowanych rosnąco zastosuj: $policz\_drogi(z);$

Funkcja policz_drogi:

(1)

$policz\_drogi(z):$
$drogi(z) \leftarrow 0;$
for każdego w-sąsiada z **do**
  if dist[w] < dist[z] **then**
    $drogi[z] = +drogi[w];$
  end if
end for
if z to nasz szukany wierzchołek u **then**
  return $drogi[u];$
end if

4 Złożoność

Zastosowanie **algorytmu Dijkstry** kosztuje nas $O(E * log(V))$ złożoności czasowej. Natępnie posortowanie wszystkich elementów kosztuje nas $O(E * log(E))$ złożoności czasowej w najgorszym wypadku. Potem pozostaje nam wykonanie funkcji policz_drogi dla kolejnych wierzchołków, w której w najgorszym przypadku dla każdego wierzchołka, odwiedzamy wszystkich jego sąsiadów, co z lematu o uściskach dłoni daje nam $O(2 * V)$ złożoności czasowej(suma stopni wierzchołków). Końcowa złożoność wynosi zatem $O(E * (log(V) + log(E)) + 2 * V)$

5 Dowód poprawności

Musimy pokazać, że nasz algorytm znalazł wszystkie sensowne drogi z u do v. Zakładając, że **algorytm Dijkstry** i **algorytm sortowania** działa poprawnie, to otrzymaliśmy dokładną odległość każdego wierzchołka od wierzchołka v i dobrze posortowaliśmy rosnąco odległość każdego wierzchołka od wierzchołka v. Załóżmy teraz, że istnieje droga sensowna $u = u_1, u_2, ..., u_{k-1}, u_k = v$. Jeżeli ta droga jest poprawna do dla każdego $i = 2, ..., k$ $dist[u_i] < dist[u_{i-1}]$. Jeżeli ta nierówność zachodzi to funkcja $policz\_drogi$ po kolei odwiedzi $u = u_1, u_2, ..., u_{k-1}, u_k = v$ i dla każdego z tych wierzchołków policzy ilość sensownych dróg (bo odwiedza po kolei wierzchołki, które są najbliżej od v i jeśli sąsiadują z innymi, które są bliżej od v to dodaje do siebie ilość sensownych dróg), więc dla u_k funkcja $policz\_drogi$ zwróci sumę ilości sensownych dróg z wszystkich sąsiadów, którzy są bliżej od wierzchołka v, więc zwróci poprawny wynik i przy okazji policzy drogę $u = u_1, u_2, ..., u_{k-1}, u_k = v$.