Lista 2. Zadanie 5

Marko Golovko

9 kwietnia 2020

Treść

Udowodnij poprawność algorytmu Boruvki (Sollina).

Idea rozwiązania

Algorytm Boruvki wyznacza minimalne drzewo rozpinające dla grafu nieskierowanego ważonego, o ile jest on spójny. Innymi słowy, znajduje drzewo zawierające wszystkie wierzchołki grafu, którego waga jest najmniejsza możliwa. Jest to przykład algorytmu zachłannego.

Algorytm

Algorytm działa poprawnie przy założeniu, że wszystkie krawędzie w grafie mają różne wagi. Jeśli tak nie jest, można np. przypisać każdej krawędzi unikatowy indeks i jeśli dojdzie do porównania dwóch krawędzi o takich samych wagach, wybrać tę, która otrzymała niższy numer.

1. Dla każdego wierzchołka v w grafie G przejrzyj zbiór incydentnych z nim krawędzi. Dołóż najlżejszą z nich do rozwiązania (zbioru E').

2. Po tym etapie graf tymczasowego rozwiązania powinien zawierać nie więcej niż $\frac{|V|}{2}$ spójnych składowych. Utwórz graf G' w którym wierzchołki stanowiące spójne składowe zostaną ze sobą „sklejone” (nowe wierzchołki będziemy nazywać roboczo *superwierzchokami*).

3. Dopóki nie otrzymamy jednej spójnej składowej, wywołujemy kroki 1–2, za graf G podstawiając graf G'.

Po zakończeniu algorytmu G' jest minimalnym drzewem rozpinającym.

Dowód poprawności

Lemat 1

W każdym momencie działania algorytmu, oraz po jego zakończeniu w ***E'*** *nie będzie cyklu.*

Dowód. Załóżmy nie wprost, że podczas działania algorytmu w którymś etapie pojawiła się spójna składowa, w której jest cykl. Oznaczmy ją jako ***S*** Rozważmy następujące sytuacje:

- ***S*** powstała przez połączenie dwóch superwierzchołków v_1 i v_2. Oznacza to, że do zbioru ***E'*** zostały dołączone krawędzie e_i i e_j Ponieważ e_i została dołączona jako najlżejsza krawędź incydentna do v_1 więc $C(e_i) < C(e_j)$. Ale skoro e_j została dołączona jako najlżejsza krawędź incydentna do v_2 to musi zachodzić $C(e_j) < C(e_i)$ (Pamiętajmy, że w grafie nie ma krawędzi o takiej samej wadze) – sprzeczność.

- ***S*** powstała przez połączenie się trzech lub więcej superwierzchołków. Podzielmy powstały cykl C na następujące części: niech v_1,v_2,v_3,...,v_l będą kolejnymi superwierzchołkami należącymi do C a e_1,e_2,e_3,...,e_l ędą kolejnymi krawędziami należącymi do C, które zostały dodane w zakończonym właśnie etapie algorytmu. W C krawędzie e_i oraz superwierzchołki v_i występują na przemian. zasady działania algorytmu możemy stwierdzić, że aby powstał taki cykl, musi zachodzić $C(e_1) < C(e_2) < \ldots < C(e_{l-1}) < C(e_l) < C(e_1)$ - sprzeczność.

Lemat 2

W każdym etapie działania algorytmu otrzymujemy dla każdego superwierzchołka minimalne drzewo rozpinające

Dowód. Gdy zostanie zakończony etap 1:
Załóżmy, że istnieje taki superwierzchołek V_i, który nie jest minimalnym drzewem rozpinającym poddrzewa złożonego z wierzchołków należących do V_i. Weźmy więc takie minimalne drzewo rozpinające $\mathbf{T}$. Istnieje krawędź e_i taka, że $e_i \in E(V_i) \wedge e_i \notin E(T)$. Dodajmy e_i. W $\mathbf{T}$ powstał cykl. Ponieważ e_i jest incydenta do pewnego wierzchołka z tego cyklu, istnieje więc inna krawędź e_i' incydentna do tego samego wierzchołka. Jednak z tego, że $e_i \in E(V_i)$ wynika, że $C(e_i) < C(e_i')$. Jeśli usuniemy krawędź e_i' z $\mathbf{T}$ otrzymamy mniejsze drzewo rozpinające, co jest sprzeczne z założeniem o minimalności $\mathbf{T}$.

Gdy zostanie zakończony etap 2:
Z poprawności etapu 1 wiemy, że istnieje takie wywołanie etapu 2, w którym każdy z superwierzchołków jest minimalnym drzewem rozpinającym. Jest to choćby pierwsze wywołanie. Załóżmy zatem, że dla pewnego wywołania tego etapu otrzymano superwierzchołki będące minimalnymi drzewami rozpinającymi, jednak scaliło przynajmniej dwa z nich w taki sposób, że dało się otrzymać mniejsze drzewo rozpinające.

Niech etap k-ty będzie pierwszym takim etapem, w którym coś się popsuło. Niech E_1' będzie zbiorem krawędzi przed wywołaniem etapu k, a E_2' będzie zbiorem krawędzi po jego wywołaniu. Niech $\mathbf{T}$ będzie minimalnym drzewem rozpinającym takim, że $V(T) = V(V_i)$, ale że że $E(T) \neq E(V_i)$ Istnieje więc krawędź $e_i \in E(V_i) \wedge e_i \notin E(T)$

Fakt: Krawędź e_i została dodana podczas k-tego wywołania. (Nie może należeć do E_1', gdyż inaczej superwierzchołek do którego by należała nie byłby minimalnym drzewem rozpinającym, co jest sprzeczne z dowodem dla pierwszego etapu i założeniem, że

wywołanie k-te jest najmniejszym wywołaniem, które zwróciło nieoptymalne drzewa)

Dodajmy krawędź e_i do $E(T)$. W $\mathbf{T}$ powstał cykl. Ponieważ e_i jest najmniejszą krawędzią incydentną do pewnego superwierzchołka z tego cyklu, istnieje więc inna krawędź incydenta do tego samego superwierzchołka. Jednak jej waga jest większa niż waga krawędzi e_i, zatem zastąpienie jej krawędzią e_i da nam mniejsze drzewo rozpinające co jest sprzeczne z założeniem o optymalności $\mathbf{T}$.

Złożoność algorytmu

Łatwo udowodnić, że każdym kolejnym wywołaniu liczba wierzchołków zmaleje co najmniej dwukrotnie – zatem wywołań tych będzie co najwyżej $\log V$. Złożoność obliczeniowa całości zależy więc od sposobu implementacji kroków 1, 2 algorytmu. Zastosowanie w implementacji struktury zbiorów rozłącznych pozwala osiągnąć złożoność na poziomie $O(E \log V)$. Można zmodyfikować go tak, aby osiągnąć złożoność $O(E \log V - V)$ – algorytm działa wtedy znacznie szybciej dla grafów rzadkich; dla grafów gęstych modyfikacja nie daje dużych zysków czasowych.