Rozwiązanie zadania 1 z listy 6

2020-05-11

1 Opis Zadania

Zadanie 1. INPUT: Graf reprezentowany przy pomocy list sąsiedztwa. Zaprojektuj algorytm rozwiązujący problem wyszukiwania najkrótszej ścieżki od źródła do wszystkich innych węzłów w skierowanym grafie acyklicznym z wagami na krawędziach. Przeanalizuj złozoność obliczeniową algorytmu.

1.1 Struktura Grafu

Procedura Dag-Shortest-Paths jest napisana dla grafu G. Zakładam, że graf ten jest tablicą wierzchołków (Struktury V), z czego każdy wierzchołek ma 4 parametry: pi - poprzednika zapamiętywanego w celu wyznaczenia najkrótszej ścieżki, d - oznacza tutaj wyliczaną przez algorytm długość najkrótszej ścieżki od źródła do danego węzła, f - czas przeszukania DFS(G) i Adj. Adj jest listą która przechowuje struktury E (krawędź) a ta ma dwa parametry: w - waga tej krawędzi i v - wierzchołek do którego biegnie.

1.2 Algorytm

Algorithm 1 Algorytm wyznaczający najkrótsze ścieżki z jednym źródłem w acyklicznych grafach skierowanych

```
1:  procedure DAG-SHORTEST-PATHS(Graph G,Struct V s)
2:      linearized-order ← topological-sort(G)
3:      for each v in G do
4:          v.d ← inf.
5:          v.pi ← NIL.
6:      s.d ← 0
7:      for each u in linearized-order do
8:          for each e in u.Adj do
9:              if e.v.d > u.d + e.v.w then
10:                 e.v.d ← u.d + e.v.w.
11:                 e.v.pi ← u.
```

1.2.1 Sortowanie Topologiczne

Częścią algorytmu Dag-Shortest-Paths jest algorytm topological-sort który ma za zadanie posortować wierzchołki w taki sposób, że jeżeli graf G zawiera krawędź (u,v) to w tym porządku wierzchołek u występuje przed wierzchołkiem v. Sortowanie to polega na wykonywaniu DFS(G) (algorytm podany na wykładzie) i w trakcie działania tego wyszukiwania, gdy kolorujemy jakiś wierzchołek na czarno (czyli zaznaczamy, że został przetworzony) dadajemy go na początek listy. Na koniec tą listę zwracamy.

Jako że sprawdzamy graf acykiliczny to nie będzie tam żadnych cykli. Dzięki temu jeżeli w grafie istnieje krawędź (u,v), i badamy ją aktualnie w DFS to v na pewno nie jest szary, bo wtedy byłby przodkiem u co spowodowałoby pojawienie sie cyklu. Dlatego v musi być biały lub czarny. Jeżeli jest biały to jest potomkiem u, co oznacza, że zostanie dodany do listy wcześniej niż u. Jeżeli jest czarny to znaczy, że już został dodany wcześniej, więc u będzie występować po nim. Dzięki temu uzyskamy liniowe ustawienie wierzchołków

Algorytm linearyzacji (sortowania topologicznego) opisany jest na stronie wykładowcy w notatkach do wykładu z dnia 11.05.2020.; linearyzacja DAG jest także omówiona w rozdziale 3.3 podręcznika "Algorithms" S.Dasgupty et al. Szczegółowy opis tego algorytmu (wraz z przykładami) można również znaleźć w podręczniku T.Cormen et al., "Introduction to Algorithms" (3rd edition, chapter 22.4)

1.3 Uzasadnienie poprawności Dag-Shortest-Paths

Po wykonaniu Dag-Shortest-Paths dla grafu G, w v.d jest waga najkrótszej ścierzki, a graf stworzony z poprzedników jest drzewem najkrótszych ścierzek. Warto na początku zaznaczyć, że jeżeli wierzchołek nie jest osiągalny ze źródła s to jego v.d = inf.

Załóżmy jednak że istnieje najkrótsza ścieżka łącząca $u_0 = s$ i $u_k = v$, składająca się z $< u_0, u_1, .., u_k >$. Jako, że przetwarzamy wierzchołki w kolejności topologicznej to krawędzie będą przetwarzane w kolejności $(u_0, u_1), (u_1, u_2)..,(u_{k-1}, u_k)$.

Poprzez indukcje dla i pokażemy, że po wykonaniu kroków 9 - 11 na danej krawędzi na naszej ścierzce w $u_i.d$ mamy długość najkrótszej ścierzki. Dla i=0 w wyniku inicjowania mamy że s.d=0=$u_0.d$=(najkrótsza ścieżka z s do s). Dla kroku indukcyjnego załóżmy że u_{i-1}.d = (najkrótsza ścieżka z s do u_{i-1}). Po wykonaniu kroków 9 - 11 dla krawędzi (u_{i-1}, u_i) widzimy, że u_i.d = (najkrótsza ścieżka z s do u_i). Wartości $u_i.d$ od tego momentu już się nie zmieni, bo przechodziliśmy po najkrótszej ścieżce (własność relaksacji dla ścieżki). Innymi słowy, na koniec algorytmu wartość v.d w każdym z wierzchołków grafu spełnia zależność $v.d = \min\{u.d + u.Adj[v].w : (u, v) \in E\}$ Dzięki temu z własności grafu poprzedników możemy wywniąskować, że graf G_pi (graf wierzchołków połączony przez poprzedników) jest drzewem najkrótszych ścieżek.

1.4 Złożoność Dag-Shortest-Paths

Sortowanie topologiczne zjmuje nam $\Theta(V + E)$, ponieważ przeszukiwanie w głąb zajuje $\Theta(V + E)$ a wstawienie na początek listy O(1). Pętla w liniach 3 - 5 zajumje $\Theta(V)$, bo przechodzi przez wszystkie wierzchołki, a jej środek zajmuje O(1). Pętle w 7-8 przechodzą dla każdego wierzchołka, po każdej krawędzi dokładnie raz, a każda iteracja wewnętrznej pętli for zajmuje O(1). Czyli łączny czas działania aglorytmu jest liniowy względem rozmiaru listowej reprezentacji grafu. Ostatecznie: $\Theta(V + E)$.