Lista 4 z AiSD

Zadanie 2 (3 na nowej liście)

Kamil Banaś 308262

7 maja 2020

1 Treść

Zbiór $I \subseteq V$ zbioru wierzchołków w grafie $G = (V, E)$ nazywamy *zbiorem niezależnym*, jeśli żadne dwa wierzchołki z I nie są połączone krawędzią. Ułóż algorytm, który dla zadanego drzewa T znajduje najliczniejszy zbiór niezależny jego wierzchołków.

2 Teoria

2.1 Obliczenie ilości elementów w najliczniejszym zbiorze

Na początku dla każdego poddrzewa drzewa T policzymy maksymalną liczbę wierzchołków, które możemy wybrać. Ze względu na to, że operujemy na drzewie, to możemy arbitralnie ukorzenić drzewo T. BSO niech wierzchołek 1 będzie korzeniem. Wobec tego przyjmijmy, że $DP(V)$ oznacza wynik algorytmu dla poddrzewa z korzeniem w wierzchołku V. Rozwiązaniem podproblemu będzie policzenie $DP(1)$, gdyż obliczając go obliczymy także wszystkie wartości DP dla każdego poddrzewa.
Zauważmy, że jeżeli w rozwiązaniu weźmiemy wierzchołek V, to nie możemy już wziąć żadnego dziecka V, natomiast jeżeli w rozwiązaniu nie weźmiemy wierzchołka V, to dla każdego dziecka V możemy zdecydować, czy weźmiemy go do rozwiązania czy nie. Zdefiniujmy więc nowe liczby:
$DP_I(V)$ - wynik dla poddrzewa z korzeniem w wierzchołku V, jeżeli weźmiemy V do wyniku,
$DP_O(V)$ - wynik dla poddrzewa z korzeniem w wierzchołku V, jeżeli V nie będzie należeć do wyniku.
Biorąc pod uwagę wcześniejsze rozważania widzimy, że:
$DP(V) = \max(DP_I(V), DP_O(V))$,
$DP_I(V) = 1 + \sum_{i=1}^{k}(DP_O(v_i))$,
$DP_O(V) = \sum_{i=1}^{k}(\max(DP_I(v_i), DP_O(v_i)))$,
gdzie przyjmujemy, że dzieci wierzchołka V to ciąg $(v_1, ..., v_k)$. Dzięki takiej rekurencji jesteśmy w stanie napisać algorytm, który policzy $DP(1)$.

2.2 Znalezienie najliczniejszego zbioru

Dzięki obliczonym wartościom możemy przystąpić do zasadniczej części zadania. Na początku odpowiadamy na pytanie, czy wierzchołek 1 znajduje się w zbiorze. Odpowiedzią

jest tak, jeżeli $DP(1) = DP_I(1)$, nie w.p.p.. Schodząc głębiej w drzewo postępujemy analogicznie, jednak musimy pamiętać, czy dodaliśmy do zbioru ojca rozpatrywanego wierzchołka. Jeżeli tak, to nie możemy dodać danego wierzchołka i od razu przechodzimy do jego synów.

3 Algorytm

Algorytmami wykorzystywanymi przy rozwiązaniu zadania będą dwa DFS-y. Dzięki pierwszemu DFS-owi możemy policzyć DP dzieci wierzchołka zanim przystąpimy do owego wierzchołka, natomiast dzięki drugiemu DFS-owi możemy trzymać informację dla dzieci, czy ich ojciec znajduje się w wynikowym zbiorze. Algorytm wygląda następująco:

---

procedure DFS1(V)
  k = liczba dzieci V
  for $i = 1; i \leqslant k; i \leftarrow i + 1$ **do**
    DFS1(v_i)
    $Suma_I = Suma_I + DP_O(v_i)$
    $Suma_O = Suma_O + \max(DP_I(v_i), DP_O(v_i))$
  end for
  $DP_I(V) = 1 + Suma_I$
  $DP_O(V) = Suma_O$
  $DP(V) = \max(DP_I(V), DP_O(V))$
end procedure
procedure DFS2(V, b)
  k = liczba dzieci V
  for $i = 1; i \leqslant k; i \leftarrow i + 1$ **do**
    if $b == 0$ and $DP(V) == DP_I(V)$ **then**
      $zbior \leftarrow \{V\}$
      DFS2($v_i, 1$)
    else
      DFS2($v_i, 0$)
    end if
  end for
end procedure
$zbior = \{\}$
DFS1(1)
DFS2(1, 0)
return $zbior$

---

4 Uzasadnienie poprawności

Udowodnimy, że algorytm DFS1 zwraca maksymalny wynik dla danego drzewa. Niezmiennikiem w rozwiązaniu będzie fakt, że dla każdego podrzewa jeżeli DFS1 odwiedzi całe to podrzewo, to wartośc DP korzenia będzie poprawnie policzona, czyli rozwiązaliśmy zadanie dla podrzewa.
Dowód będzie się opierać na indukcji strukturalnej względem drzewa.

Baza indukcji: Dla liścia l zawsze mamy $DP(l) = 1$.
Krok indukcyjny: Zakładamy, że dla danego wierzchołka V mamy policzone DP dla każdego dziecka. Wobec tego musimy policzyć teraz $DP(V)$. W algorytmie widzimy, że obliczamy maksymalne sumy dla przypadków jeżeli weźmiemy V do wyniku lub jeżeli V nie będzie należał do wyniku. Możemy tak zrobić, ponieważ sumy zależą tylko od dzieci V, które są poprawnie wyliczone. Ostatecznie dostajemy $DP_I(V)$ oraz $DP_O(V)$, z których wartości bierzemy maksimum. Widzimy, że wartość $DP(V)$ jest dobrze policzona.
Wniosek: Z powyższej indukcji wynika, że dla każdego poddrzewa V obliczamy poprawną wartość $DP(V)$. Wobec tego również otrzymujemy poprawną wartość $DP(1)$, czyli wynik dla drzewa. Zauważmy, że z tego wynika fakt, że możemy arbitralnie wybrać korzeń drzewa, gdyż dla dowolnego wierzchołka będącego korzeniem dostaniemy poprawny wynik dla całego drzewa.
Musimy jeszcze udowodnić poprawność tworzenia maksymalnego zbioru podczas algorytmu DFS2. Wystarczy rozpatrzyć oba możliwe przypadki:

- bierzemy dany wierzchołek do naszego zbioru. Robimy to w algorytmie tylko wtedy, kiedy wybranie tego wierzchołka jest równoważne z wzięciem największej ilości wierzchołków z danego poddrzewa.

- nie bierzemy danego wierzchołka. Sytuacja ta zachodzi jeżeli wzięliśmy już ojca, ale wtedy zmierzamy do wybrania największej ilości wierzchołków z poddrzewa zaczynającego się od ojca, a wynik poddrzewa ojca liczyliśmy biorąc pod uwagę aktualnie rozpatrywane poddrzewo. W przeciwnym wypadku uzyskamy większą liczbę wierzchołków nie biorąc danego wierzchołka.

Ostatecznie uzyskujemy konkluzję, że DFS1 poprawnie oblicza maksymalny wynik dla każdego podrzewa, co wykorzystuje DFS2 idąc od korzenia i dodając wierzchołki do zbioru jeżeli zapewnia to maksymalną ilość wierzchołków z danego poddrzewa lub z poddrzewa ojca.

5 Złożoność czasowa

W algorytmie używamy tylko dwa razy funkcji DFS, która dla każdego wierzchołka wykonuje stałą ilość operacji dla każdej krawędzi. Złożoność czasowa DFS to $O(V + E)$, która jest ostatecznie równa $O(V + V) = O(V)$, gdyż naszym grafem jest drzewo. Zatem ostateczna złożoność to $O(V + V) = O(V)$

□