Zadanie 2 lista 4

Jakub Kuciński, prowadzący Szymon Dudycz

26 czerwca 2020

Spis treści

1 Treść zadania

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 Idea rozwiązania

Ukorzeńmy graf w dowolnym wierzchołku. Dla dowolnego wierzchołka v zastanówmy się jaki jest maksymalny zbiór niezależny I_v w tworzonym przez niego poddrzewie. Wierzchołek v może należeć lub nie należeć do maksymalnego zbioru niezależnego, stąd mamy przypadki:

1. $v \in I_v$
   Wtedy żadne u dziecko v nie może należeć do I_v. Wystarczy zatem znaleźć najmniejsze zbiory niezależne poddrzew o korzeniach w u, do których nie należą u. Zatem I_v będzie ich sumą mnogościową powiększoną o v.

2. $v \notin I_v$
   Wtedy zbiory niezależne poddrzew utworzonych w u dzieciach v mogą dowolnie zawierać lub nie zawierać u. Czyli I_v będzie sumą mnogościową większych ze zbiorów niezależnych zawierających i niezawierających u.

Z powyższych rozważań widzimy, że wystarczy dla każdego wierzchołka policzyć maksymalne zbiory niezależne zawierające i niezawierające ich dzieci dla poddrzew o korzeniach w dzieciach. Oczywiście dla liścia w mamy odpowiednio $\{w\}$ i $\emptyset$. Aby szybko rozpoznać który zbiór niezależny jest większy wystarczy oprócz jego elementów przechowywać również jego rozmiar.

3 Algorytm

Algorithm 1 *Zbior_niezalezny*(v)

$S_{with} \leftarrow \{v\}$
$S_{without} \leftarrow \emptyset$
for $u \in N(v)$ **do**
    if u nieodwiedzony **then**
        oznacz u jako odwiedzony
        $A_{with}, A_{without} \leftarrow Zbior\_niezalezny(u)$
        $S_{with}. \leftarrow S_{with} \cup A_{without}$
        if $A_{with}.size > A_{without}.size$ **then**
            $S_{without} \leftarrow S_{without} \cup A_{with}$
        else
            $S_{without} \leftarrow S_{without} \cup A_{without}$
return $S_{with}, S_{without}$

W celu obliczenia największego zbioru niezależnego drzewa wystarczy wywołać powyższą procedurę na dowolnym początkowym wierzchołku i zwrócić większy z otrzymanych zbiorów.

4 Złożoność obliczeniowa i pamięciowa

4.1 Złożoność pamięciowa

Utworzenie listy sąsiedztwa wymaga $\mathcal{O}(n)$ pamięci. Zbiory konstruowane przez nasz algorytm mogą być zwykłymi listami wiązanymi, dla których trzymamy wskaźniki na pierwszy i ostatni element oraz rozmiar zbioru. Wówczas łączenie zbiorów polega na podpięciu pewnej listy wiązanej na koniec innej listy wiązanej oraz zmiany rozmiaru zbioru na sumę rozmiarów łączonych zbiorów. W końcowym zbiorze może znaleźć się co najwyżej $n-1$ wierzchołków, stąd zbiory wymagają $\mathcal{O}(n)$ pamięci. Informacje, czy dany wierzchołek został odwiedzony można trzymać w dodatkowej tablicy. Wymaga to $\mathcal{O}(n)$ pamięci. Ostatecznie algorytm zużywa $\mathcal{O}(n)$ pamięci.

4.2 Złożoność obliczeniowa

Każdy wierzchołek zostanie odwiedzony dokładnie raz, tak jak przy zwykłym DFS'ie. Operacje inicjalizacji, sumowania zbiorów, aktualizacji i sprawdzania ich rozmiarów wykonują się w czasie stałym. Stąd złożoność czasowa wynosi $\mathcal{O}(n)$.