Lista 4 zadanie 3

Wiktor Hamberger
308982

5 maja 2020

Ideą rozwiązania rodzi się z pewnej oczywistej obserwacji, dla każdego ukorzenionego drzewa T najliczniejszy zbiór niezależny jego wierzchołków może albo zawierać korzeń, albo go nie zawierać. Jeżeli rozwiązanie ma go zawierać, to naturalnie nie możemy wybrać jego synów do tego zbioru. Natomiast jeżeli nie weźmiemy korzenia, to możemy, ale nie musimy, wybrać do tego zbioru każde z jego dzieci. Dla każdego odebranego dziecka musimy teraz wykonać ten sam proces decyzyjny co dla korzenia – dziecko, po odcięciu od korzenia, jest korzeniem jakiegoś mniejszego poddrzewa. I tak dalej aż do samych liści. Określmy sobie dwa zbiory A_i – najliczniejszy zbiór niezależnych wierzchołków będących potomkami i i zawierający i oraz B_i – najliczniejszy zbiór niezależnych wierzchołków będących potomkami i nie zawierający i. Teraz wzory:

$$A_i = \begin{cases} \{i\} & \text{dla i będących liścmi,} \\ \{i\} \cup \bigcup_{j-dzieci\ i} B_j & \text{w przeciwnym przypadku} \end{cases}$$

$$B_i = \begin{cases} \varnothing & \text{dla i będących liścmi,} \\ \bigcup_{j-dzieci\ i} maksZB(A_j, B_j) & \text{w przeciwnym przypadku} \end{cases}$$

Gdzie funkcja $maksZB()$ zwraca zbiór o większej mocy.

Data:
T – drzewo zawierające co najmniej jeden wierzchołek o indeksie 0,
maksZB() – zdefiniowane jak wyżej
Result: Najliczniejszy zbiór niezależnych wierzchołków tego drzewa
Function *fun(T)* **is**
    alg(0);
    return maksZB(A_0, B_0);
end
Function *alg(u)* **is**
    $A_u := \{i\}$;
    $B_u := \varnothing$;
    if *u nie jest liściem* **then**
        foreach *v* ***in*** *dzieci(u)* **do**
            alg(v);
            $A_u := A_u \cup B_v$;
            $B_u := B_u \cup maksZB(A_v, B_v)$;
        end
    end
end

Dowód poprawności
Indukcja względem liczby wierzchołków

- Dla drzewa o jednym wierzchołku algorytm poprawnie liczy parę (A, B), gdzie $|A| = 1, |B| = 0$ i zwraca jako wynik zbiór A.
- Załóżmy, że algorytm poprawnie liczy zbiory (A_i, B_i) dla każdego drzewa o n wierzchołkach. Weźmy więc dowolne drzewo o n+1 wierzchołkach. Ukorzeniamy je w wierzchołku o etykiecie 0, a następnie usuńmy korzeń z tego drzewa. Powstanie wtedy pewien las złożony z podrzew tego drzewa, przy czym każde drzewo będzie miało co najwyżej n wierzchołków. Każde z tych drzew ukorzeniamy w dziecku wierzchołka 0. Z założenia wiemy, że te drzewa będą miały poprawnie policzone zbiory (A_i, B_i), gdzie i będą etykietami dzieci wierzchołka 0. Z tego wynika, że suma $maksZB(A_i, B_i)$, zgodnie z założeniem, będzie oznaczała najliczniejszy zbiór niezależnych wierzchołków tego lasu, więc nasze B_0 wyliczone przez algorytm jest maksymalne (wierzchołka 0 nie ma w lesie, więc połączenie tych podrzew do tego wierzchołka nic nie psuje). Jeżeli jednak zsumujemy B_i otrzymamy najliczniejszy zbiór niezależnych wierzchołków tego lasu, bez uwzględniania w tym zbiorze żadnego z korzeni – w takim razie nic nie stoi na przeszkodzie, żeby dodać do niego wierzchołek 0. W ten sposób otrzymujemy maksymalny możliwy zbiór A_0. W takim razie para (A_0, B_0) jest obliczona poprawnie, więc algorytm działa poprawnie dla każdego drzewa.

Pozostaje kwestia złożoności. Z racji, że operujemy na drzewie, do każdego wierzchołka wchodzimy dokładnie raz. Przy sprytnej implementacji maksZB()

(na przykład każdy zbiór pamięta też swój rozmiar) możemy uzyskać czas stały takiego sprawdzania. Co za tym idzie dla każdego wierzchołka wykonujemy stałą liczbę operacji, a każdy wierzchołek przetwarzamy dokładnie raz, więc złożoność to O(n), gdzie n to liczba wierzchołków drzewa.