

Zad 8.

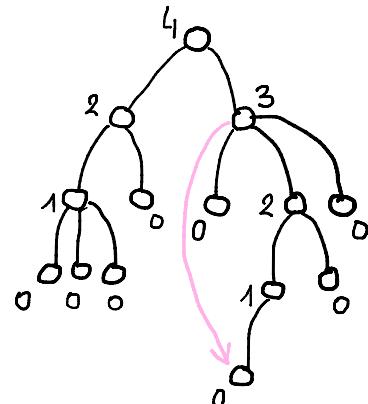
niedziela, 2 kwietnia 2023 14:38

8. (2pkt) Niech $T = (V, E)$ będzie drzewem a $P(u, v)$ niech oznacza ścieżkę w T (rozumianą jako zbiór krawędzi) łączącą wierzchołki u i v .

Ułóż algorytm, który dla drzewa T znajduje trzy wierzchołki a, b, c , dla których zbiór $\{e \in E : e \in P(a, b) \cup P(a, c) \cup P(b, c)\}$ jest maksymalnie duży.

ukazujemy drzewo w dowolnym wierzchołku (były były stopnia ≥ 2)

dodatkowo pamiętamy liścia położonego najdalej (przykład na ●)



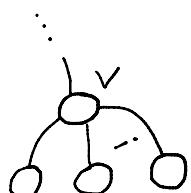
liczymy wysokości poddrzew ukorzenionych w każdym poszczególnym wierzchołku

Fakt 1.

w drzewie ścieżki pomiędzy dowolnymi 3-ma wierzchołkami przecinają się w dokładnie jednym wierzchołku

skorzystamy z tego, żeby rozważać „środki” tych ścieżek

idea algorytmu



chcemy „przypisać” każdemu v „najlepszą wartość z a) oraz b)

jeżeli wierzchołek v ma być miejscem przecięcia tych 3 ścieżek to mamy dwie opcje
 a) a, b, c leżą w tutej różnych poddrzewach v
 b) a, b leżą w dwóch różnych poddrzewach v , a c leży w takim miejscu, że ścieżka z v do c idzie co najmniej raz w góre

puszczamy dfs'a z kozenia z dwoma parametrami:

$v \leftarrow$ aktualnie przetwarzany wierzchołek

$up \leftarrow$ odległość do najbliższego liścia = głębi v (w innym poddrzewie niż v)

procedure $dfs(v, up)$: nieodwiedzonego sąsiada

ilość zyskanych krawędzi idąc w dół poddrzewo → $M[1..3] \leftarrow [0, 0, 0]$

dla każdego dziecka u wierzchołka v rób:

jeżeli $h[u] > M[1]$ to

$M[3] \leftarrow M[2]$

$M[2] \leftarrow M[1]$

$M[1] \leftarrow h[u] + 1$

jeżeli $h[u] \geq M[2]$ to

$M[3] \leftarrow M[2]$

$M[2] \leftarrow h[u] + 1$

jeżeli $h[u] \geq M[3]$ to

$M[3] \leftarrow h[u] + 1$

wyznaczamy trójkę "najlepszych" dzieci (lub mniej, jeśli v ma ich mniej niż 3)

jeżeli $|dzieci(v)| \geq 2$ to ? przypadek (b)

$best \leftarrow \max(best, M[1] + M[2] + up)$] +

jeżeli $|dzieci(v)| \geq 3$ to ? przypadek (a)

$best \leftarrow \max(best, M[1] + M[2] + M[3])$

dla każdego dziecka u wierzchołka v :

jeżeli $h[u] + 1 \neq M[1]$ to ← wracamy do v
 $n_up \leftarrow \max(up + 1, M[1] + 1)$ ← wchodzimy do najbliższego dziecka
 wpp ← obroć ścieżkę co było
 $n_up \leftarrow \max(up + 1, M[2] + 1)$ ← to samo tylko z drugim najlepszym
 $dfs(u, n_up)$

wywołujemy się rekurencyjnie w dzieciach

pierwsze wywołanie: albo $(0, root)$

$dfs(root, 0)$, best = -1

żeby na koniec uzyskać wierzchołki a, b, c zamiast maksymalnego pokrycia krawędzi, to w up 'ie i $M[1..3]$ pamiętamy te liście, o których wspominałem na początku i aktualizujemy je w momencie zmiany best'a i $n_up'a$

złożoność: proporcjonalna względem rozmiaru drewna