Най-близък общ предшественик на два върха в дърво. Минимален елемент в отрез от масив. (част 1)

16.10.2020 г.

(Lowest Common Ancestor (LCA) and Range Minimum Query (RMQ))

RMQ-проблем:

Дадено: Имаме даден масив от (различни) естествени числа - ключове $A[0...n-1]$.

Вход: $0 \le i \le j < n$.

Изход: Търси се $k = \arg \min_{l \in [i,j]} A[l]$, т.е. $A[k] = \min\{A[l] \mid i \le l \le j\}$.

LCA:

$T(V, p, r)$ е кореново дърво. Тогава за връх в кореново дърво $v \in V$ нека $p^*(v) = \{p^{(k)}(v) \mid 0 \le k \le d(v)\}$. За върхове $u, v \in V$, общите предшественици са $p^*(u) \cap p^*(v)$. Ще търсим $\arg\max d(w) : w \in p^*(u) \cap p^*(v)$. Тук формално изказахме дефиницията за LCA, която не толкова формално може да се каже и по следните начини: Най-близкия общ предшественик на върховете $u, v \in V$, от кореновото дърво $T(V, p, r)$ е върхът w,

1. който се съдържа и в двата прости пътища от корена r до връх u и от корена r до върха v и е с възможно най-голяма дълбочина.
2. който има и върховете u и v като свой наследници и е с максимална дълбочина.
3. се намира на най-краткия път между върховете u и v и е най-близо до корена r.

И трите дефиниции са еквивалентни.

LCA-проблем:

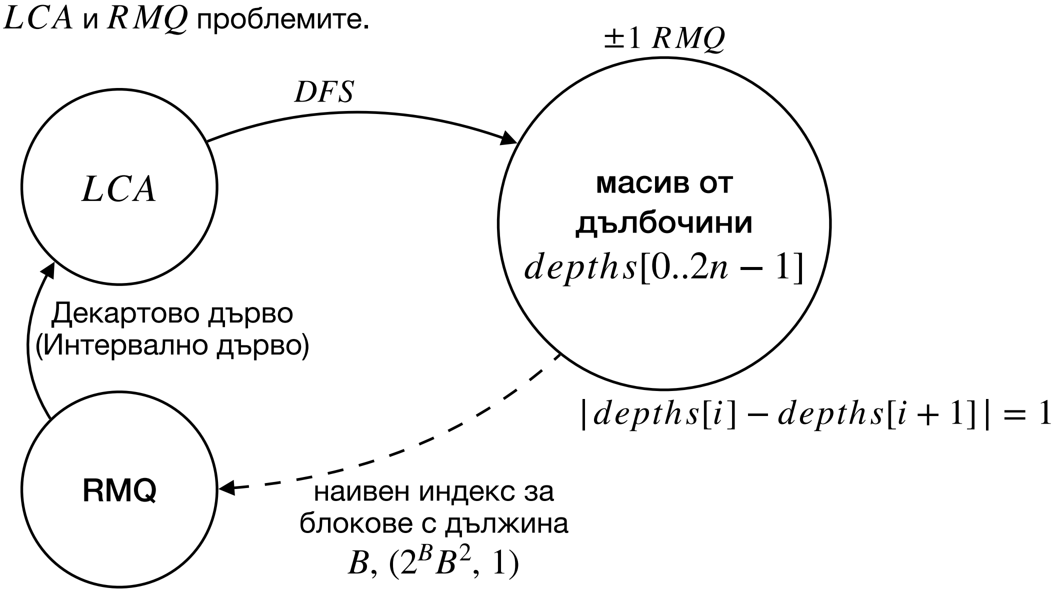
Дадено: $T = (V, p, r)$ кореново дърво.

Вход: Върхове $u, v \in V$.

Изход: Търси се $\arg\max d(w) : w \in p^*(u) \cap p^*(v)$.

Цел: $< O(n, 1) >$ времева сложност съответно за създаване на индекс (структура от данни за индексиране) и за заявка за LCA.

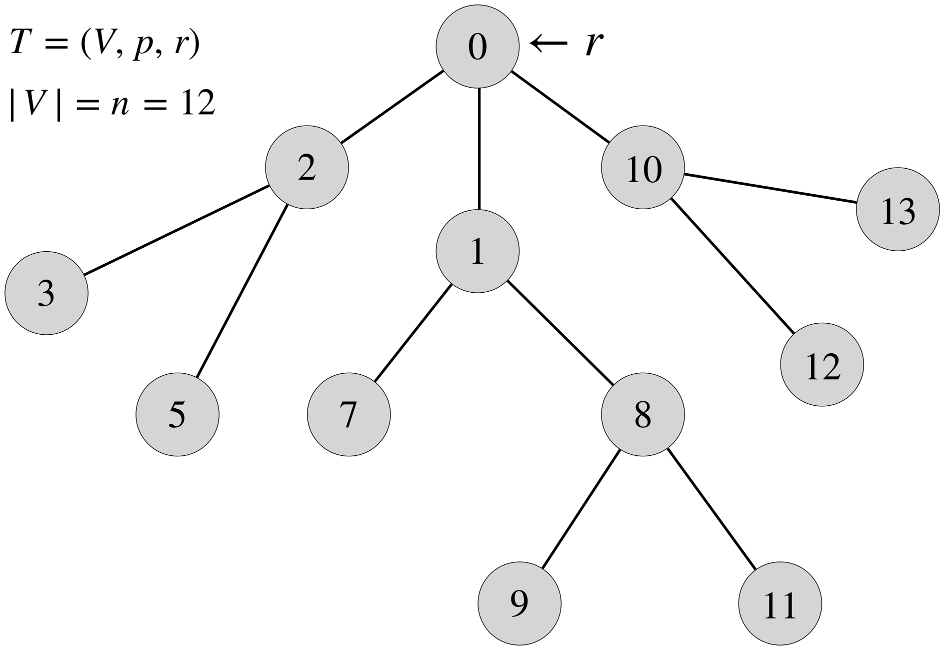
Свързаност на LCA и RMQ проблемите.



Свеждане на LCA-проблема към ± 1 RMQ-проблем.

$T = (V, p, r)$

$|V| = n = 12$



$euler$	0	2	3	2	5	2	0	1	7	1	8	9	8	11	8	1	0	10	12	10	13	10	0
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22

$depths$	0	1	1	2	Ø	2	Ø	2	2	3	1	3	2	2
$start$	0	7	1	2	Ø	4	Ø	8	10	11	17	13	18	20
end	22	15	5	2	Ø	4	Ø	8	14	11	21	13	18	20
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13

Обхождане в дълбочина (псевдо код):

```
time ← 0
start[0...|V| − 1]
end[0...|V| − 1]
visited[0...2|V| − 1]
depths[0...2|V| − 1]
DFS(T, v)
    start[v] = time
    visited[time] ← v
    depths[time] ← d(v)
    time ← time + 1
    for u : p(u) = v
        DFS(T, u)
        visited[time] ← v
        depths[time] ← d(v)
        time ← time + 1
    end[v] ← time − 1
```

Друг подход (C++):

```
        global variables :
        vector < list < int > > adj;
        vector < int > euler, dep, s, f;
        int timer;
void dfs(int v = 0, int d = 0, int p = − 1){
    s[v] = timer;
    euler[timer + + ] = v;
    dep[v] = d;
    for (const int & child : adj[v]){
        if (child = = p) continue;
        dfs(child, d + 1, v);
        euler[timer + + ] = v;
    }
    fin[v] = timer − 1;
}
```

Интересува ни: $DFS(T, r)$

1. Последната стойност на $time$ е $2|V| - 1$, тъй като $time$ се увеличава веднъж за всеки връх и всяко ребро $\Rightarrow |V|$ пъти за всеки връх и $|V| - 1 = |E|$ за всяко ребро $\Rightarrow 2|V| - 1$.
2. За всеки $u, v \in T$:
    1. Ако $start[u] < start[v]$ и $end[v] < end[u]$, то тогава $T_v \subsetneq T_u$;
    2. Ако $start[u] < start[v]$ и $end[u] < start[v]$, то тогава $T_u \cap T_v = \emptyset$.
3. $|depths[t] - depths[t + 1]| = 1$
4. Ако $k = RMQ(depths, start[u], start[v])$, то $visited[k] = LCA(u, v)$.