

**Най-близък общ предшественик на два върха в дърво.
Минимален елемент в отрез от масив.
(част 2)**

23.10.2020 г.

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

Стъпка I:

Алгоритъма от предишната лекция:

```
time ← 0
visited[0 ... 2|V|]
depth[0 ... 2|V|]
start[0 ... |V| - 1]
end[0 ... |V| - 1]
time ← time + 1                                (Vertex)
procedure DFS(T, v) {
    for each u in p(v) = u do
        DFS(T, u)
        depth[time] ← d(v)
        visited[time] ← v
        time ← time + 1                        (Edge)
    end[v] ← time - 1
}
```

```
procedure LCA(u, v) { // start[u] ≤ start[v]
    k ← RMQ(depth, start[u], start[v])
    return visited[k]
}
```

Коректност:

1. За всеки два върха *u* и *v* с условието $start[u] \leq start[v]$ е в сила точно едно от двете:
 - или $T_v \subseteq T_u$ и тогава $[start[v], end[v]] \subseteq [start[u], end[u]]$
 - или $T_u \cap T_v = \emptyset$ и тогава $[start[u], end[u]] \cap [start[v], end[v]] = \emptyset$

При DFS(*T*, *v*), *time* нараства за всеки връх (Vertex) → $|V_{T_v}|$ пъти, както и за всяко ребро (Edge) → $|E_{T_v}| = |V_{T_v}| - 1$ пъти. Следователно $end[v] - start[v] = 2(|V_{T_v}| - 1)$

В интервала $[start[v], end[v]]$ е записан върха *v* толкова пъти, колкото са синовете *children*(*v*) на *v*.

От тук с индукция по $|V_{T_v}|$ може да видим, че в интервала $[start[v], end[v]]$ се срещат само върхове от T_v .

1) База: $|V_{T_v}| = 1$.

2) Нека $|V_{T_v}| > 1$ и c_1, c_2, \dots, c_s са синовете на *v*, в който са обходетни във *for* цикъла.

Тогава от индуктивното предположение имаме, че в интервала $[start[c_i], end[c_i]]$ има върхове само от T_{c_i} .

От цикъла се вижда, че $end[c_i] < start[c_{i+1}] < end[v]$. Тоест това са непресичащи се интервали в $[start[v], end[v]]$.

Остана да забележим, че:

$$\begin{aligned} end[v] - start[v] &= 2(|V_{T_v}| - 1) = 2 \sum_{i=1}^s (|V_{T_{c_i}}|) = 2 \left(\sum_{i=1}^s (|V_{T_{c_i}}| - 1 + 1) \right) = \\ &= 2 \sum_{i=1}^s (|V_{T_{c_i}}| - 1) + 2s = \sum_{i=1}^s (end[c_i] - start[c_i]) + 2s = \sum_{i=1}^s (end[c_i] - start[c_i] + 1) \underbrace{- s}_{\text{брой записи на върха } v \text{ в} \\ &\quad \text{отворения интервал} \\ &\quad (start[v], end[v])} \end{aligned}$$

Това показва, че други елементи освен v и върховете от T_{c_i} в $[start[v], end[v]]$ няма.

Сега, ако $T_u \cap T_v = \emptyset$, то в интервала $[start[u], end[u]]$ се срещат само елементи на T_u , а в интервала $[start[v], end[v]]$ се срещат само елементи на T_v . Ако двата интервала имаха общ елемент k , то $visited[k] \in T_u \cap T_v$, което е **противоречие** с условието $T_u \cap T_v = \emptyset$.

Обратно: В интервала $[start[u], end[u]]$ са записани всички върхове от T_u и T_v . Тогава без ограничение на общността T_v е поддърво на T_u ($T_v \subseteq T_u$) и елементите на T_v са записани в интервала $[start[v], end[v]]$, а извън този интервал няма елементи на T_v и следователно $[start[v], end[v]] \subseteq [start[u], end[u]]$.

2. Нека u и v са два върха, такива че $start[u] \leq start[v]$ и нека $k = \arg \min depth[l]$, където $start[u] \leq l \leq start[v]$ и $w = visited[k]$. Тогава твърдим, че $w = LCA(u, v)$.

2.1. сл. $T_v \subseteq T_u$. Тогава в интервала $[start[u], end[u]]$ са записани върхове само от дървото T_u , в което всички дълбочини са \geq от тази на върха u и единственият връх с дълбочина $d(u)$ е u . На позиция $t = start[u]$ в масива $depth$ имаме, че $depth[t] = d(u) \Rightarrow \min depth[k] = d(u), start[u] \leq k \leq start[v]$. От първата част $visited[k] = u$. От друга страна от това, че $T_v \subseteq T_u$ е ясно, че $LCA(u, v) = u = visited[k]$.

2.2. сл. $T_u \cap T_v = \emptyset$. Нека \tilde{w} е техен най-близък общ предшественик. Нека \tilde{u} и \tilde{v} са синовете на \tilde{w} : $u \in T_{\tilde{u}}$ и $v \in T_{\tilde{v}}$. Тогава $[start[u], end[u]] \subseteq [start[\tilde{u}], end[\tilde{u}]] \subseteq [start[\tilde{w}], end[\tilde{w}]]$ и $[start[v], end[v]] \subseteq [start[\tilde{v}], end[\tilde{v}]] \subseteq [start[\tilde{w}], end[\tilde{w}]]$

Освен това $T_{\tilde{u}} \cap T_{\tilde{v}} = \emptyset$, от където следва, че $[start[\tilde{u}], end[\tilde{u}]] \cap [start[\tilde{v}], end[\tilde{v}]] = \emptyset$.

Сега: $start[\tilde{u}] \leq start[u] < start[v] \leq end[\tilde{v}]$.

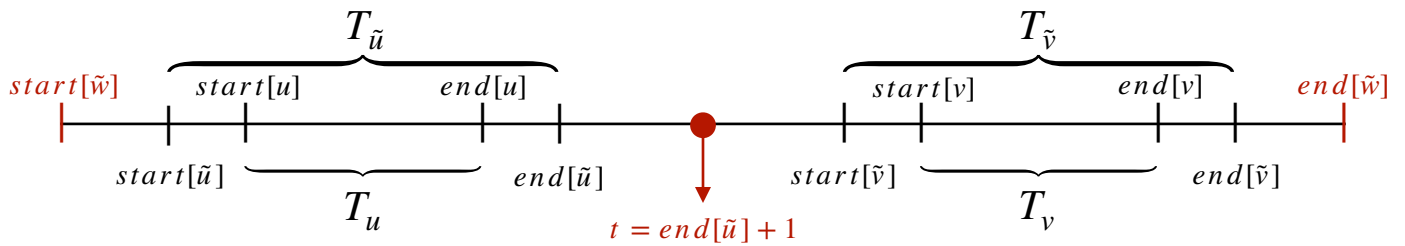
Ако $start[\tilde{v}] \leq start[\tilde{u}]$, то $[start[\tilde{v}], end[\tilde{u}]]$ има общи елементи с

$[start[u], end[u]] \Rightarrow u \in T_v$, но $v \in T_v \Rightarrow \tilde{v}$ е предшественик на u и v , който е по-дълбоко от $\tilde{u} \Rightarrow start[\tilde{u}] < start[\tilde{v}] \Rightarrow end[\tilde{u}] < start[\tilde{v}]$.

$$T_u \cap T_v = \emptyset$$

Тоест разположението на интервалите е следното:

$$start[u] \leq end[\tilde{u}] < end[\tilde{u}] + 1 \leq start[\tilde{v}] \leq start[v]$$



Тъй като \tilde{u} е син на \tilde{w} :

$$\begin{cases} depth[t] = d(\tilde{w}) \\ visited[t] = \tilde{w} \end{cases} \Rightarrow \min_{start[u] < l \leq start[v]} depth[l] \leq d(\tilde{w})$$

Накрая $T_u, T_v \subseteq T_{\tilde{w}} \Rightarrow start[\tilde{w}] \leq start[u] \leq start[v] \leq end[\tilde{w}]$, следователно:

$$\min_{start[u] \leq l \leq start[v]} depth[l] \geq \min_{start[\tilde{w}] \leq l \leq end[\tilde{w}]} depth[l] = d(\tilde{w}) \quad !$$

$\Rightarrow d(\tilde{w}) = depth[t]$ е минимумът в $[start[u], start[v]]$ и тъй като $t \in [start[\tilde{w}], end[\tilde{w}]]$, то $visited[t] = \tilde{w} \Rightarrow \min_{start[u] \leq l \leq start[v]} depth[l] \leq d(\tilde{w})$.

Забележка: $|depth[t] - depth[t + 1]| = 1$ за всяко $t < 2(|V| - 1)$.

Дефиниция: Проблема $\pm 1 RMQ$ е следния казус:

Дадено: Масив $A[0 \dots n - 1]$ от естествени (може и цели) числа, така че $|A[i] - A[i + 1]| = 1$, за $i < n - 1$.

Вход: $0 \leq i \leq j < n$

Изход: $k = \arg \min_{i \leq l \leq j} A[l]$

Следствие: Ако имаме решение $(f(n), g(n))$ за $\pm 1 RMQ$ проблема, то имаме решение $(n + f(2n), g(2n))$ за LCA -проблема.

Стъпка I беше свеждането на LCA -проблема към RMQ -проблема

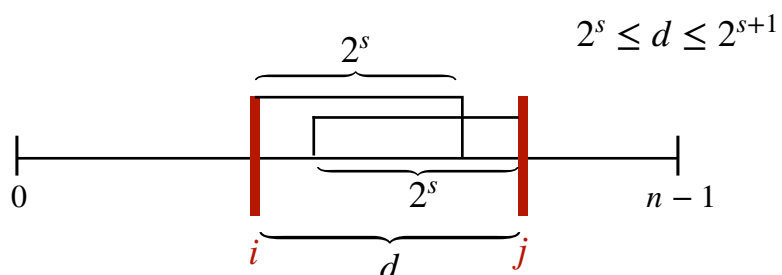
Стъпка II: Решение на RMQ -проблема за $O(n \log n, 1)$ времева сложност.

Дадено: Масив $A[0 \dots n - 1] \subset \mathbb{Z}$

Вход: $0 \leq i \leq j < n$

Изход: $k = \arg \min_{i \leq l \leq j} A[l]$

Идея:



1. Построяване на индекс \mathcal{A}_I .

$table[i][k] = \arg \min A[i \dots i + 2^k - 1]$

($table[i][0] = i$ за всяко i)

```
for  $i = 0$  to  $n - 1$  do
     $table[i][0] = i$ 

 $k \leftarrow 1$ 
 $pow \leftarrow 1$ 
while  $2 * pow \leq n$  do
    for  $i = 0$  to  $(n - 1) - (2 * pow - 1)$  do
         $l \leftarrow table[i][k - 1]$ 
         $r \leftarrow table[i + pow][k - 1]$ 
        if  $A[l] \leq A[r]$  then
             $table[i][k] \leftarrow l$ 
        else
             $table[i][k] \leftarrow r$ 
    done
     $pow \leftarrow 2 * pow$ 
     $k \leftarrow k + 1$ 
}
```

Естествено преди това трябва да сме си преизчислили логаритмите закръглени надолу на всички числа от 2 до n и степените на двойката неरेвишаващи n :

```
 $\log[0 \dots n - 1]$ 
 $pow1[0 \dots \log[n - 1]]$ 
 $pow \leftarrow 1, j \leftarrow 0$  //  $2^j = pow$ 
for  $i = 0$  to  $n - 1$  do
    if  $i = pow - 1$  then //  $i = 2^j - 1$ 
         $pow1[j] \leftarrow i$ 
         $\log[i] \leftarrow j$ 
    if  $i + 1 = 2 * pow$  then
         $pow \leftarrow pow * 2$ 
         $j \leftarrow j + 1$ 
}
```

```
procedure  $RMQ(i, j)$  { //  $i \leq j$ 
     $d \leftarrow j - i + 1$ 
     $k \leftarrow \log[d]$ 
     $pow \leftarrow pow1[k]$  //  $2^k \leq d \leq 2^{k+1}$ 
     $l \leftarrow table[i][k]$ 
     $r \leftarrow table[j - pow][k]$ 
    if  $A[l] \leq A[r]$  then return  $l$ 
    return  $r$ 
}
```

Коректност:

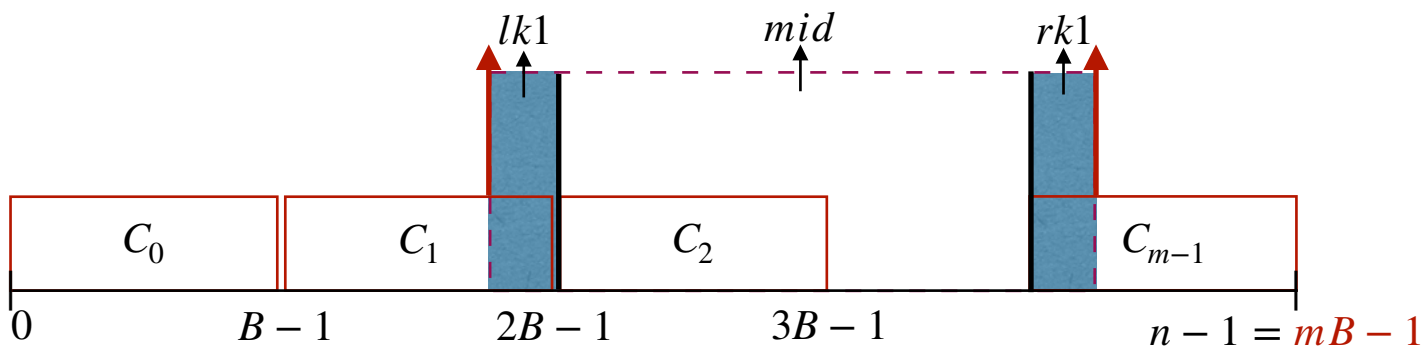
$$d = j - i + 1$$

$$2^k \leq j - i + 1 < 2^{k+1}, \text{ т.е.}$$

$$\Rightarrow j - \text{pow} \geq i$$

От друга страна $i + 2^{k+1} - 1 \geq j + 1 \Rightarrow i + 2^k - 1 \geq j - 2^k + 1 \Rightarrow j - \text{pow}$

III. Решение на ± 1 RMQ-проблема за $O(n, 1)$



За всеки един от тези блокове може да намерим минимума $c_i = \min C_i$

$$I_i = \arg; \min C_i \in [Bi \dots B(i+1) - 1]$$



За всеки блок $c_i \mapsto$ (съпоставяме) масив от стойности

$\tau[j] = C_i[j] - C_i[0] \in [-B \dots B]$, като разликата на всеки два последователни

типа $|\tau[j+1] - \tau[j]| = 1, \tau_i[0] = 0 \Rightarrow$ най-много 2^B различни масива τ . Всеки от тях може да адресираме с наивен алгоритъм $(B^2, 1)$ - решение.

Анализ:

Двете части на решението имат сложност:

$$f_1(n) = \left(\frac{n}{B} \log \frac{n}{B}, 1 \right) \text{ и } f_2(n) = (2^B B^2, 1), \text{ т.е. общото време за индексирание ще е}$$

$$f(n) = f_1(n) + f_2(n) = \frac{n}{B} \log \frac{n}{B} + 2^B B^2 \stackrel{?}{\in} O(n)$$

$$B \sim c \log n$$

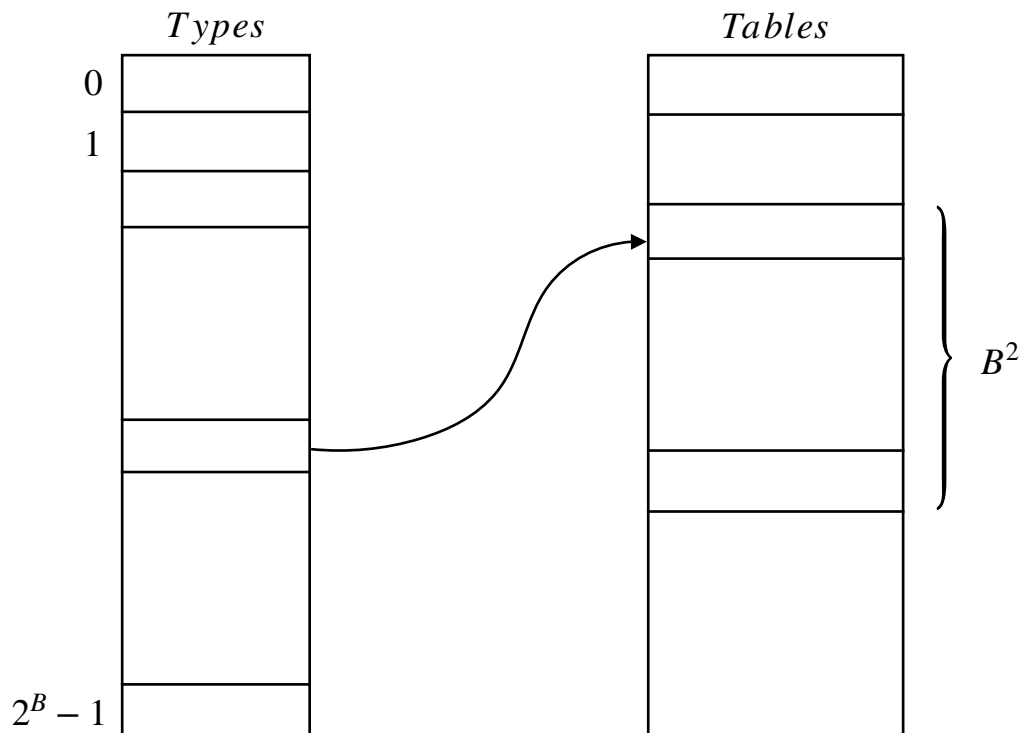
$$2^{c \log_2 n} c^2 \log_2^2 n = n^c \cdot c^2 \log_2^2 n \in O(n) \Leftrightarrow c < 1.$$

$$\text{Сега може да изберем } c = \frac{1}{2} \Rightarrow B = \left\lfloor \frac{\log_2 n}{2} \right\rfloor$$

```

procedure FindBlockType( $A, i, B$ ) {
   $type \leftarrow 0$  //  $-1 \sim 0, 1 \sim 1$ 
   $c \leftarrow A[B_i]$ 
   $I \leftarrow B_i$ 
  for  $j = 1$  to  $B - 1$  do
     $b \leftarrow A[B_i + j] - A[B_i + j - 1]$ 
    if  $b == -1$  then
       $type \leftarrow 2 * type$ 
    else
       $type \leftarrow 2 * type + 1$ 
    if  $A[B_i + j] < c$  then
       $I \leftarrow B_i + j$ 
       $c \leftarrow A[B_i + j]$ 
  done
  return ( $type, c, I$ )
}

```

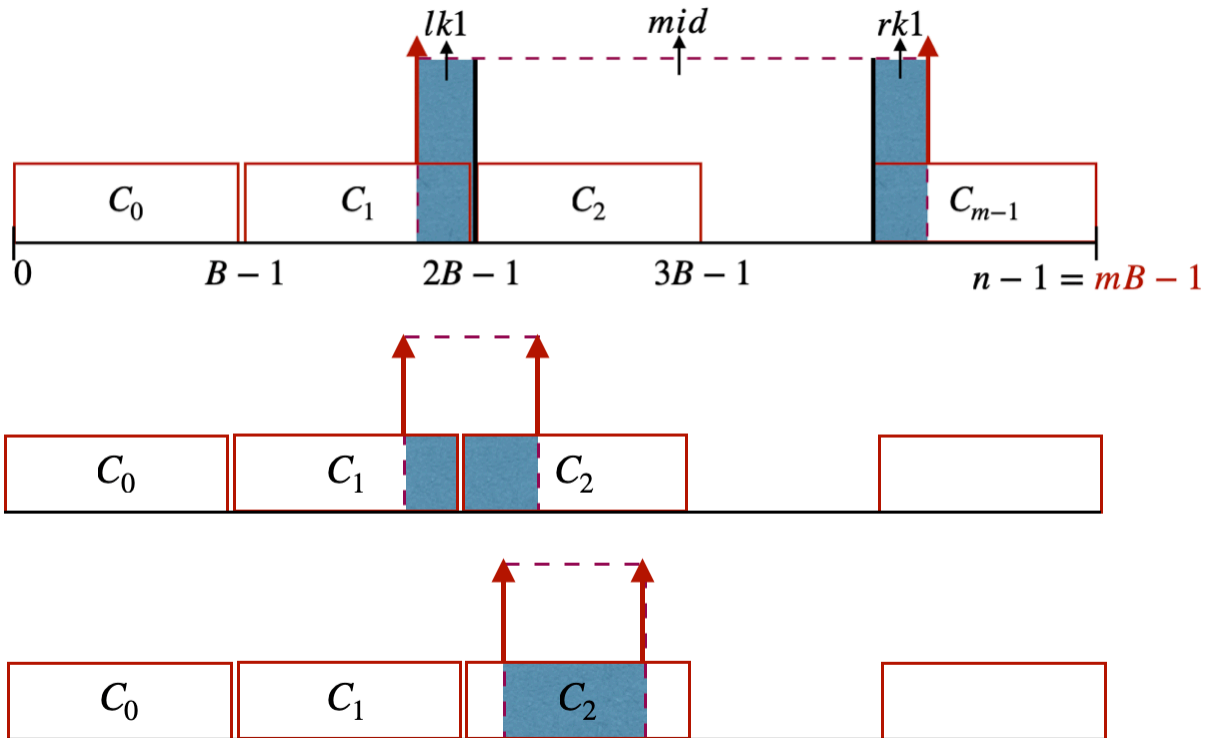


```

procedure initTypes()
    for  $t = 0$  to  $2^B - 1$  do
         $Types[t] \leftarrow -1$ 

procedure SetType( $A, i, B, type$ ) {
    if  $Types[type] \neq -1$  then
        return
     $c \leftarrow currentSize(Tables)$ 
    for  $k = 0$  to  $B - 1$  then
        for  $l = 0$  to  $k - 1$  do
             $Tables[s + l \cdot B + l] \leftarrow -1$ 
             $m \leftarrow A[Bi + k], Tables[s + Bk + k] \leftarrow k$ 
        for  $l = k + 1$  to  $B - 1$  do
            if  $A[Bi + l] < A[Tables[s + kB + l - 1]]$  then
                 $Tables[s + kB + l] \leftarrow l$ 
            else
                 $Tables[s + kB + l] \leftarrow s + lB + l - 1$ 
        done
    done
}

```



```

procedure RMQ(A, l, r) {
     $l_0 \leftarrow \lfloor l/B \rfloor + 1$ 
     $r_0 \leftarrow \lfloor r/B \rfloor - 1$ 
    if  $l_0 \leq r_0$  then
         $mid \leftarrow \text{logRMQ}(C[0 \dots n/B - 1], l_0, r_0)$ 
         $m \leftarrow I[mid]$ 
    else
         $m \leftarrow -1$ 
    3 : { if  $l_0 - 1 = r_0 - 1$  then
         $k \leftarrow \text{RMQTable}(\text{type}[l_0 - 1], l - B[l/B], r - B[l/B])$ 
        return  $k + B[l/B]$ 
    2 : { if  $l_0 = r_0 + 1$  then
         $lk \leftarrow \text{RMQTable}(\text{type}[l_0 - 1], l - B[l/B], B - 1)$ 
         $rk \leftarrow \text{RMQTable}(\text{type}[l_0], 0, r - B[r/B])$ 
         $lk1 \leftarrow lk + B[l/B]$ 
         $rk1 \leftarrow rk + B[r/B]$ 
        if  $A[lk1] \leq A[rk1]$  then
            return  $lk1$ 
        else
            return  $rk1$ 
    1 : { if  $l_0 \leq r_0$  and  $m \neq -1$  then
         $lk \leftarrow \text{RMQTable}(\text{type}[l_0 - 1], l - B[l/B], B - 1)$ 
         $lk1 \leftarrow lk + B[l/B]$ 
         $rk \leftarrow \text{RMQTable}(\text{type}[r_0 + 1], 0, r - B[l/B])$ 
         $rk1 \leftarrow rk + B[r/B]$ 
        if  $A[lk1] \leq A[m]$  then
            return  $m$ 
        else
            return  $rk1$ 

```