Намиране на минимум в отрез от масив, чрез намиране на най-близък общ предшественик

30.10.2020 г.

(Solving RMQ by finding LCA)

RMQ :
Дадено: Масив $A[0...n-1] \subseteq \mathbb{N}$.
Вход: $0 \leq i \leq j < n$.
Изход: $k = \arg\min_{i \leq l \leq j} A[l]$.

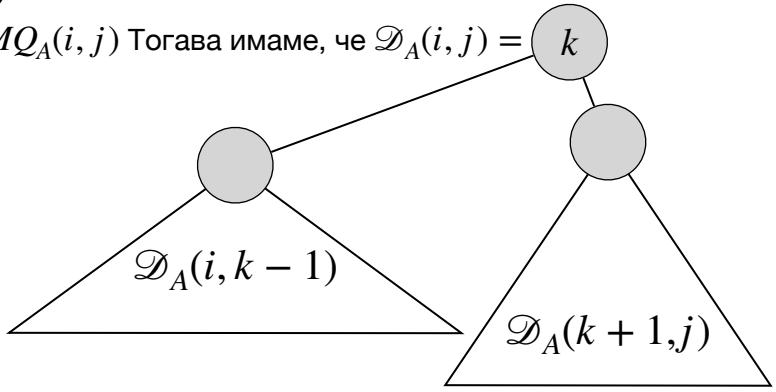
Знаем решение на LCA-проблема:
Дадено: $T = (V, p, r)$
Вход: $u, v \in V$
Изход: $w = \arg\min\{d(w') \mid w' \in p*(u) \cap p*(v)\}$
За LCA вече имаме решение $(n, 1)$

За да получим $(n, 1)$ решение на RMQ проблема - ще сведем RMQ към LCA за линейно време. Това ще постигнем чрез т.нар. декартови (сегментни) дървета.

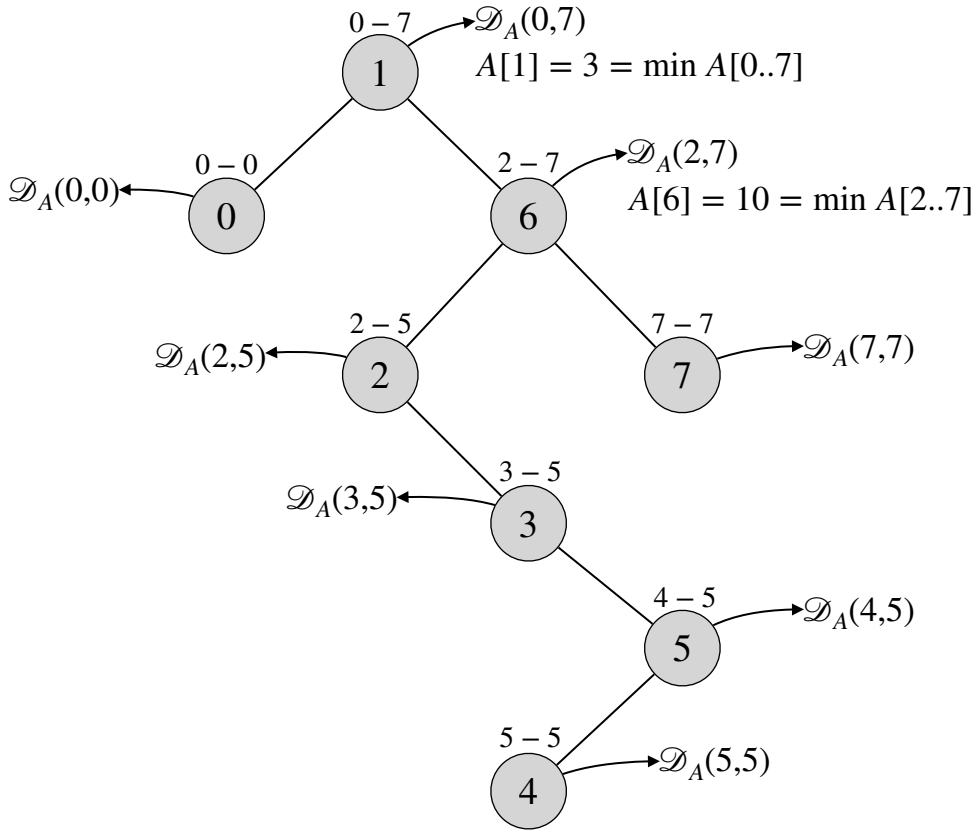
Декартови дървета

Дефиниция: Нека $A[0...n-1]$ е масив от естествени числа (може и цели, които в последствие да транслираме към естествени с някаква известна константа) числа. За $i, j < n$ декартово дърво $\mathscr{D}_A(i, j)$ дефинираме рекурсивно по следния начин:

1 сл. ако $i > j$, $\mathscr{D}_A(i, j) = \emptyset$
2 сл. ако $i \leq j$, нека $k = RMQ_A(i, j)$ Тогава имаме, че $\mathscr{D}_A(i, j) =$



Пример: A : $\underset{0}{7}, \underset{1}{3}, \underset{2}{12}, \underset{3}{128}, \underset{4}{1001}, \underset{5}{500}, \underset{6}{10}, \underset{7}{17}$; $n = 8$.
Започваме да строим дървото според дефиницията: $A \mapsto \mathscr{D}_A(0, n-1) = \mathscr{D}_A(0, 7)$.



Пример 1:
$i = 2$
$j = 5$

$RMQ_A(2,5)$: Търсим върха, който съответства на i, т.е. в конкретния случай на 2. Намираме го като се спускаме от корена на дървото и пресмятаме в коя посока да поемем. Ако стойността в корена е по-малка от i - отиваме в дясното му дете, а ако е по-малка от i - отиваме в лявото му дете и повтаряме рекурсивно описаната процедура като вече новия корен е посетеното дете. Аналогично намираме върха, който съответства на j, след което намираме $RMQ_A(2,5) = LCA_{\mathscr{D}_A(0,7)}(2,5) = 2$.

Пример 2:
$i = 4$
$j = 7$
$RMQ_A(4,7) = LCA_{\mathscr{D}_A(0,7)}(4,7) = 6.$

Във върховете на това декартово дърво стоят индекси. Тези индекси са различни числа от 0 до $n - 1$.

В момента, в който се построи декартовото дърво за целия масив A, проблема за еднакви елементи в масива отпада, тъй като всеки елемент е уникален връх в дървото, който се представлява от индекса на елемента в масива.

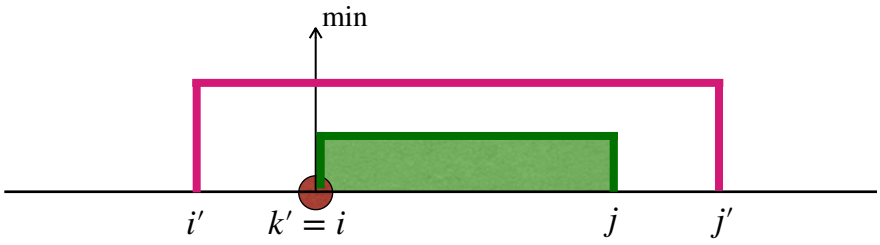
Твърдение: Нека $A[0...n-1]$ е масив от естествени числа и $T = \mathscr{D}_{A[0...n-1]}$. Тогава за всяко $i, j : 0 \leq i \leq j < n$ е в сила, че $RMQ_A(i,j) = LCA_T(i,j)$ (по-коректно: ако $k = RMQ_A(i,j)$ и $k' = LCA_T(i,j)$, то $A[k] = A[k']$. Но тези детаили не възникват, ако предполагаме някакъв детерминизъм на RMQ_A, например ако връща най-левия или най-десния елемент в съответния сегмент от i до j, изпълняващ условието)

Доказателство: Нека $k' = LCA_T(i, j)$. Тогава от построението на дървото T знаем, че на k' съответства някакъв интервал $[i', j']$. По-точно, това което знаем е, че поддървото на T с корен k' е някакво декартово дърво $\mathscr{D}_A(i', j')$, т.е. $T_{k'} = \mathscr{D}_A(i', j')$.

Сега разглеждаме лявото и дясното поддърво на k'.

От построяването на $\mathscr{D}_A(i', j')$ следва, че $i, j \in [i', j']$. Освен това знаем, че $k' = RMQ_A(i', j')$ (от дефиницията на декартово дърво)

1 сл. $i = k'$, тогава $A[i] = A[k'] = \min\ A[i' \ldots j'] \leq \min\ A[k' \ldots j]$, т.к. $[k', j] \subseteq [i', j']$. Но $k' \in [i', j]$, така че $A[k'] = \min\ A[i \ldots j]$.



Минимума в зеления интервал не е по-малък от минимума в розовия интервал и тъй като $k' \in$ зеления интервал, то $A[k'] = \min$ в зеления интервал $= \min$ в розовия интервал.

2 сл. $j = k'$. Аналогично на 1 сл.

3 сл. $i \neq k' \neq j$. Тъй като $k' = LCA_T(i, j)$, то i и j принадлежат на различни поддървета на k'. Но $T_{k'}$ има най-много 2 $\Rightarrow$ точно 2 поддървета $T^{(l)} = \mathscr{D}_A(i', k' - 1)$ и $T^{(r)} = \mathscr{D}_A(k' + 1, j')$.

Ако $i \in T^{(r)}$ и $j \in T^{(l)}$, то: $j \leq k' - 1 < k' + 1 \leq i$.

Но $i \leq j$ (от заявката) $\Rightarrow$↯ (противоречие).
Следователно, т.к. i и $j \in$ разл. поддървета на k', то $i \in T^{(l)}$, $j \in T^{(r)}$, т.е. $i \leq k' - 1 < k' < j$.

Остава да забележим, че $i' \leq i < k' < j \leq j'$

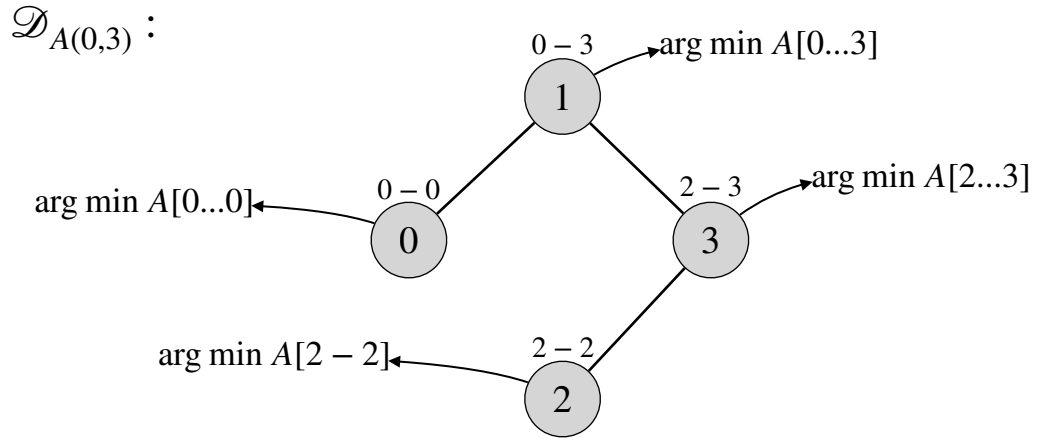
$$\Rightarrow A[k'] \overset{def.}{=} \min_{k' = RMQ_A(i', j')} A[i', j'] \underset{[i,j] \subseteq [i', j']}{\leq} A[i, j] \leq \min_{k' \in [i, j]} A[k']$$

$$\Rightarrow \min[i, j] = A[k']$$

$\Rightarrow$ ако $k = RMQ_A(i, j)$, то $A[k] = A[k']$.
(и като допълнение: ако знаем, че няма равни елементи в масива A, то $k = k'$)

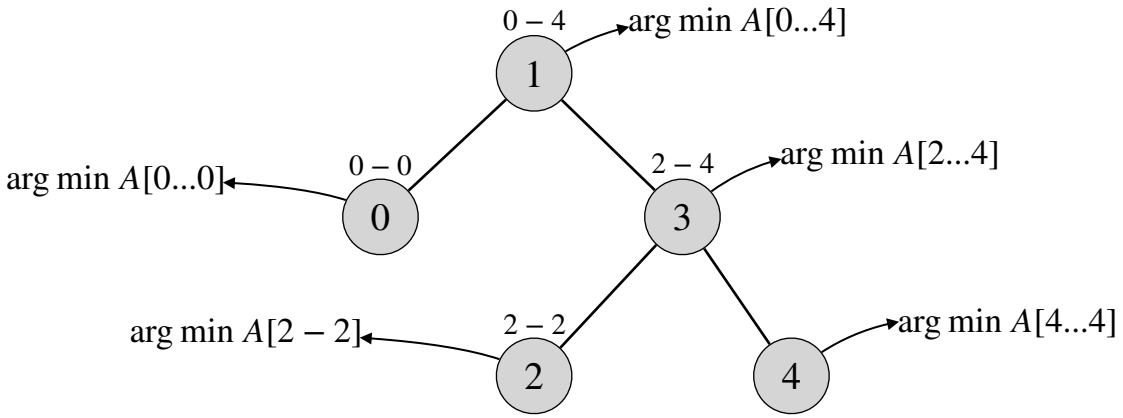
$$A: \underset{0}{7}, \underset{1}{3}, \underset{2}{500}, \underset{3}{12}, \underset{3}{1001}, \underset{5}{10}, \underset{6}{17}, \underset{7}{128}; \quad n = 8.$$

Как може ефективно да построим декартово дърво за масива A. Нека си представим, че имаме готово построено декартово дърво за първите четири елемента от масива A и искаме да добавим петия елемент към него.



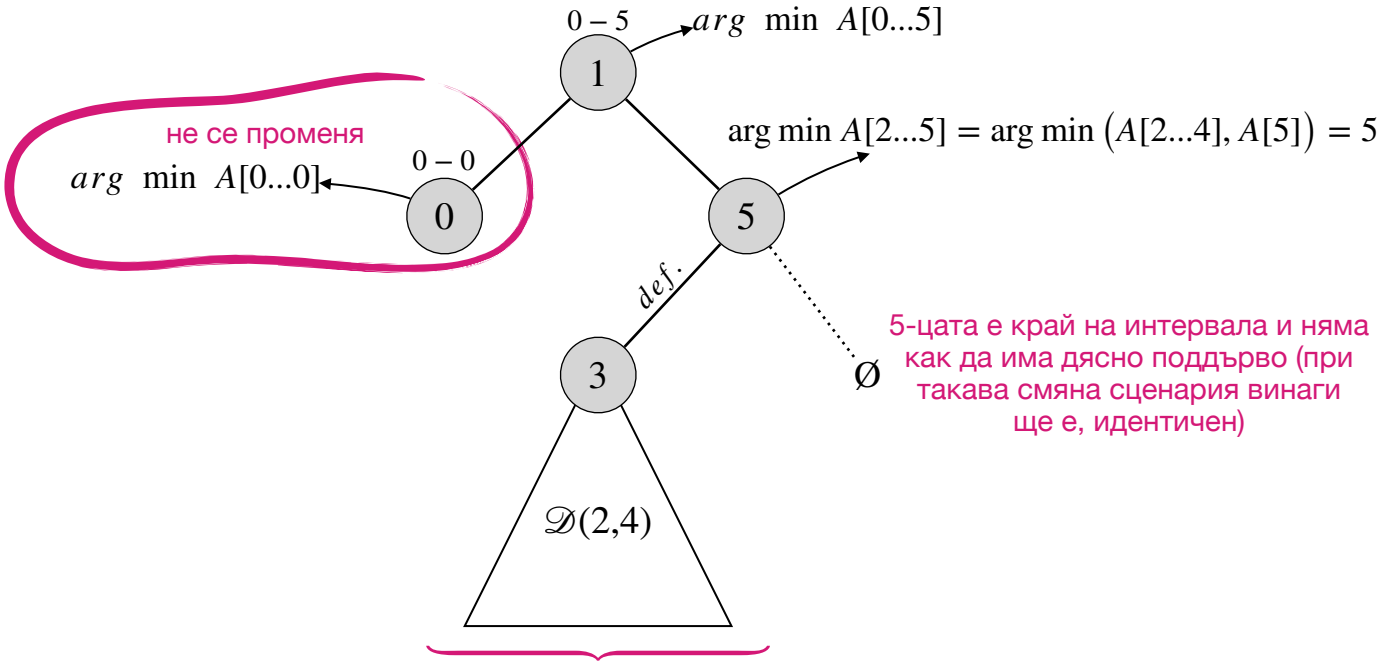
Въпросът е как да добавим индекса 4 на следващия елемент $A[4]$. Ако съумеем да го направим това за константно време, значи ще успеем да изградим цялото дърво за линейно време.

По дефиниция в корена трябва да е индекса на минималния елемент в масива $A[0...3]$. Следователно след добавянето на $A[4]$, този индекс на минимум или ще остане същия или ще се актуализира с новия индекс 4. В случая, т.к. $A[4] = 1001 > 3 = A[1]$, то корена се запазва. Лявата част също няма да се промени, т.к. тя отговаря за тривиалния отрез $A[0...0]$, а $4 \notin [0, 0]$. Продължаваме с дясното поддърво. То отговаря за интервала $A[2...3]$, но за него добавяме индекса 4. Минимума остава в индекс 3, а лявото поддърво отново не се променя.



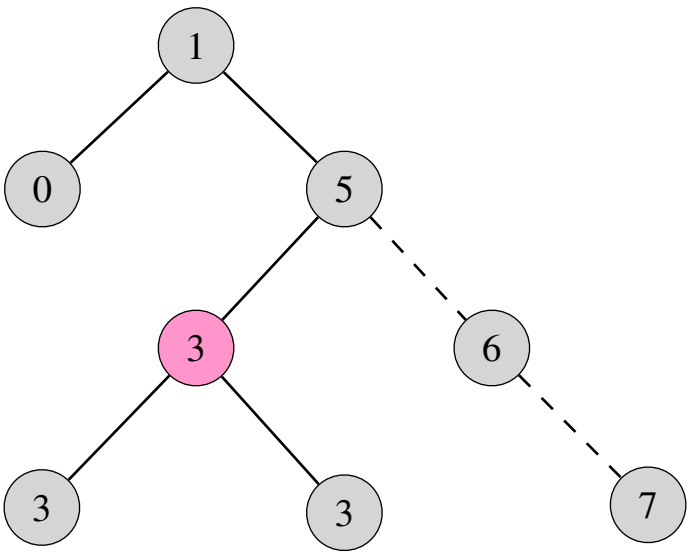
Забележете, че при това добавяне левите поддървета си останаха непроменени и съответно необходени. Нека продължим с добавянето на следващия елемент, а именно индекса 5 в дървото.

Отново тръгваме от корена, като вече ни интересува индекса на минималния от елементите на $A[0...5]$.



5-цата е край на интервала и няма как да има дясно поддърво (при такава смяна сценария винаги ще е, идентичен)

вече го имаме и трябва само да го закачим на лявото дете на върха с индекс 5

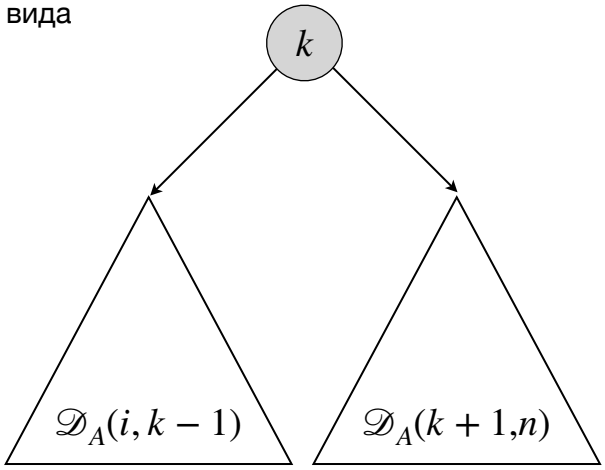


Може да направим следните заключения:
Промените, които се случват се случват само по най-десния път на декартовото дърво. В момента, в които се наложи да се направят промени в ляво - процесът на добавяне спира, тъй като вече сме добавили новия връх и на него трябва само да прикачим вече изграденото дърво.
Но ако следваме тази логика, нещата няма да са много добри, тъй като например на входа може да получим сортиран масив. Тогава тази процедура ще се спуска всеки път по пътища от порядъка на $O(n)$ за да добави елемент. така сложността ще стане $O(n^2)$, тъй като пътя нараства всеки път с едно.

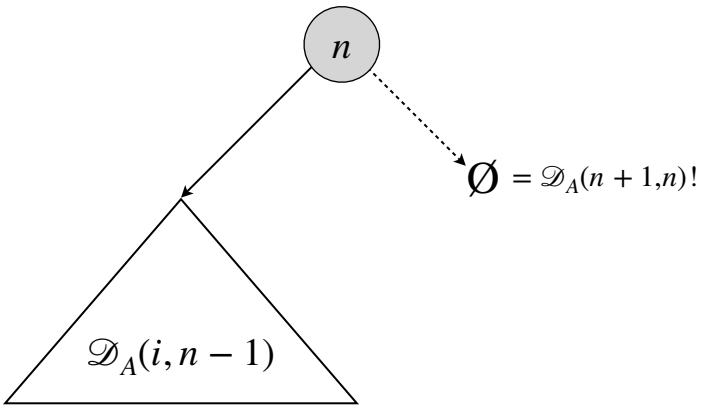
По формално може да опишем следните **НАБЛЮДЕНИЯ**:

(1) Ако имаме дърво $T_{n-1} = \mathscr{D}(0,n-1)$, то T_n Се получава от T_{n-1} като n се добавя по най-десния път в T_{n-1}, защото

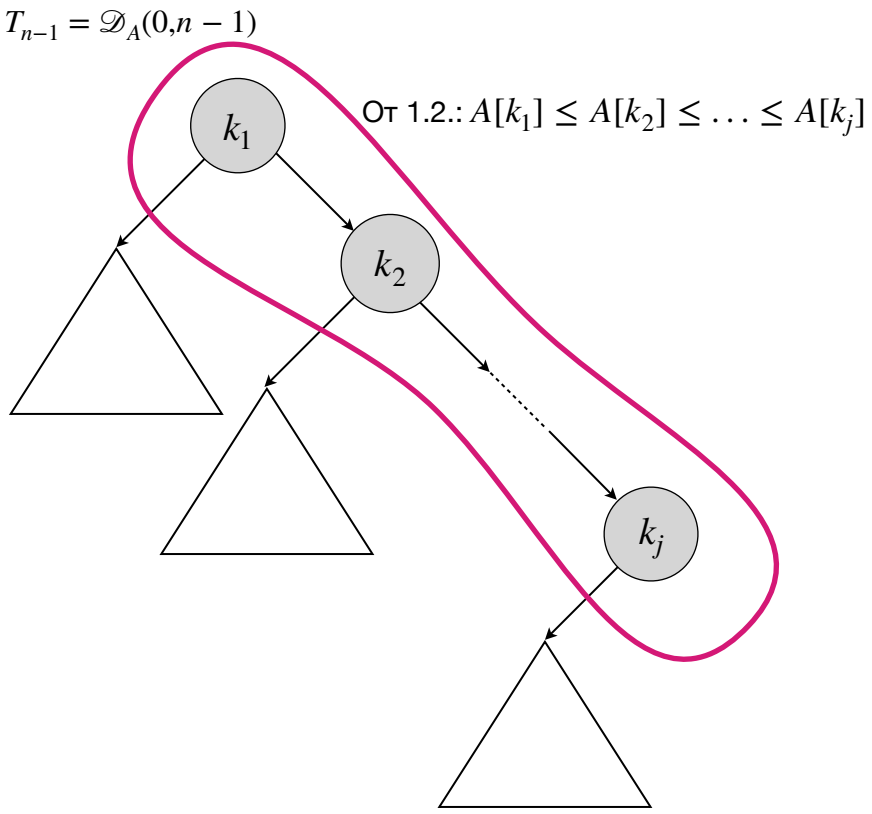
1.1. сл. Ако имаме $\mathscr{D}_A(i,n-1)$ с корен k и $A[k] < A[n]$, то $A[k] = \min A[i \dots \underline{n}]$ и съответно $\mathscr{D}_A(i,n)$ ще има вида



1.2. сл. Ако имаме $\mathscr{D}_A(i,n-1)$ с корен k и $A[k] > A[n]$, то $A[n] < A[k] = \min A[i \dots n-1] \Rightarrow A[n] = \min A[i \dots n]$. Тогава по дефиниция $\Rightarrow \mathscr{D}_A(i,n)$.



(2) От дефиницията за декартово дърво, ако $i \in T_k$ $\left(T = \mathscr{D}_A(0,n-1)\right)$, то $A[i] \geq A[k]$ $(A[i] > A[k]$ при липса на повторения в масива $A)$



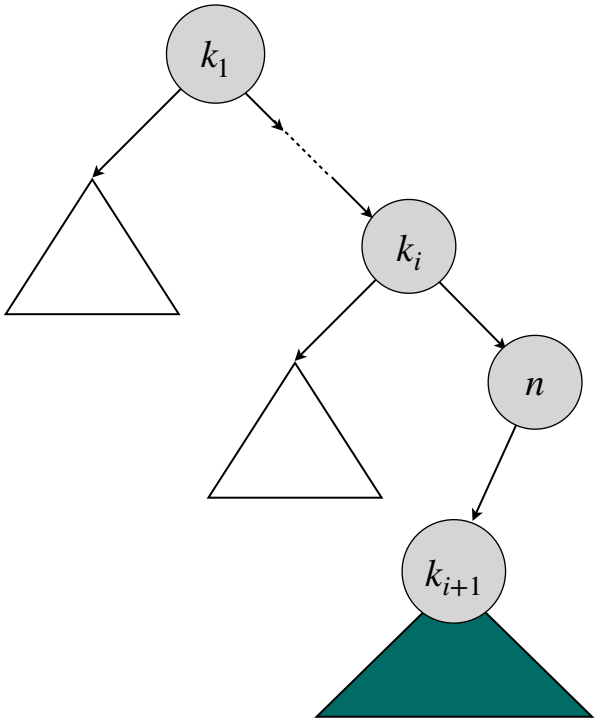
Последния елемент по най-десния път винаги ще бъде най-десния индекс в масива, който разглеждаме (по индукция)

(3) От (1) получаваме, че добавянето на n към дървото $\mathscr{D}_A(0,n-1)$ се свежда до вмъкването на $A[n]$ в сортирания списък $A[k_1] \leq A[k_2] \leq \ldots \leq A[k_j]$.

По-точно, намираме такъв индекс i :

$A[k_i] \leq A[n] < A[k_{i+1}] \quad (A[k_0] = -\infty,\ A[k_{j+1}] = +\infty)$

След което:



Но освен от началото към края, може да се опитаме да добавим елемент наивно от края на списъка.
В нашия случай, ако сменим една наивна идея с друга - ще получим качествена промяна.

Ако се окаже, че $A[k_j] > A[n]$, то тогава k_j никога повече няма да попада в най-десния път след пренареждането, тъй като той ще увисне наляво от този път. Освен това, всички елементи от k_i до k_j, които са се оказали по-малки от$A[n]$ ще увиснат в ляво на най-десния път и никога повече няма да имат възможност да се върнат на най-десния път.

Т.е. всяка стъпка, която не съответства на сравнението $A[i] < A[n]$ ще събаря върха с i -тия индекс в ляво от пътя.

Имплементация:

Поддържаме кореново дърво $T(V[0...n-1], p, r)$, което представя декартово дърво за масива $A[0...n-1]$.

Добавяне на пореден елемент:

$T \to \mathscr{D}_A(0,k-1), \quad k \geq 1$, добавяме k.

procedure $insert(T, A, k)$ {
    $v \leftarrow k-1$ // k_j
    $c \leftarrow NULL$ // ∞
    while $v \neq NULL$ **and** $A[k] < A[v]$ **do**
        $c \leftarrow v$
        $v \leftarrow p(v)$
    done
    // $v = k_i, c = k_{i+1} : A[k_i] \leq A[k] \leq A[k_{i+1}]$
    $p(k) \leftarrow v$
    if $c \neq NULL$ **then**
        $p(c) \leftarrow k$
    if $v = NULL$ **then**
        $r \leftarrow k$
}

procedure $cartesianTree(A, n)$ {
    $T \leftarrow (\{0\}, \emptyset, 0)$
    for $k = 1$ **to** $n-1$ **do**
        Insert(T, A, k)
    done
    return T
}

Коректност: След всяка итерация имаме декартово дърво $T = \mathscr{D}_A(0,k)$. Следователно от първото твърдение $RMQ_A(i,j) = LCA_T(i,j), \quad (i \le j)$.

Сложност: Нека $T_k = \mathscr{D}_A(0...k)$ и нека $d_k = d_{T_k}(k)$. Тогава добавянето на $k+1$ към T_k има следната сложност:

1. $while$-цикъла: $depth_{T_k}(k) - depth_{T_k}\big(p(k+1)\big) + 1$
2. $O(1)$ време извън $while$-цикъла за добавяне на роднински връзки

Да забележим, че: $depth_{T_k}\big(p(k+1)\big) = depth_{T_{k+1}}(k+1) - 1 = d_{k+1} - 1$

Следователно $while$-цикъла има сложност $d_k - (d_{k+1-1}) + 1 = d_k - d_{k+1} + 2$. Тогава общото време в $while$-циклите при итерациите от 1 до $n-1$ е:

$$\sum_{k=0}^{n-2}(d_k - d_{k+1} + 2) = 2(n-1) + \sum_{k=0}^{n-2} d_k - \sum_{k=1}^{n-1} d_k =$$

$$= 2(n-1) + d_0 - d_{n-1} \le 2(n-1) + d_0 - 0 = 2n - 1 = O(n).$$