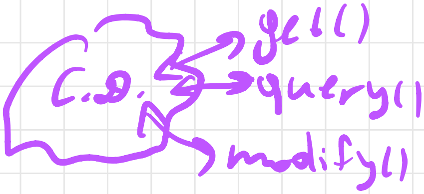
# Структур Данных




## Префиксные суммы

$a_0 \dots a_{n-1}$

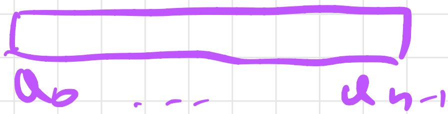
$$get(l,r) = \sum_{i=l}^{r} a_i = S_{r+1} - S_l$$

где $S_i = a_0 + \dots + a_{i-1}$

вопрос: можно ли сделать то же самое для min

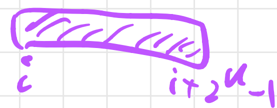
# Sparse Table

(разреж. таблица)

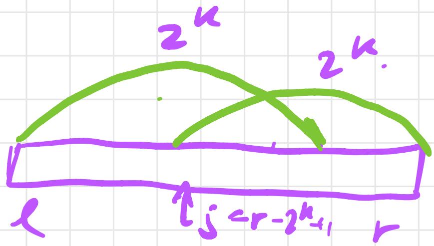



$$get(l, r) = \min_{i=l..r} a_i \qquad O(1)$$

$$f_{k i} \overset{def}{=} \min_{j=i..i+2^k-1} a_j$$




тогда: $$\min_{i=l..r} a_i \overset{(*)}{=} \min(f_{k,i}, f_{k,j})$$

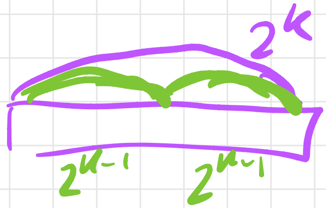



Пусть $k = \max\{k \mid 2^k \le r - l + 1\}$

Заметим что покрывают в объединении $[l; r]$

## Как искать $f_{k,i}$?

Д.П. $f_{k,i} = \min(f_{k-1,i};\ f_{k-1,i+2^{k-1}})$




$O(n\log n)$

## Как считать $k$?

В (*) нам нужно знать

$$k = \max\{k \mid 2^k \le r-l+1\}$$

Получается: $k = \lfloor \log_2 r-l+1 \rfloor$

Д.П.

$K[len] = \lfloor \log_2 len \rfloor$ ← предподсчёт

$K[1] = 0$

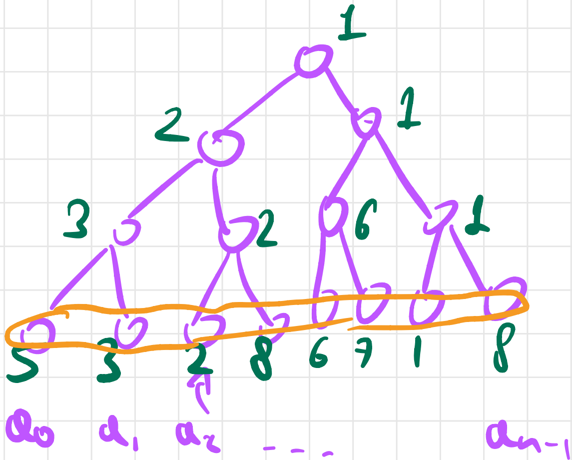
$k[len] = 1 + k[\lfloor len/2 \rfloor]$

RMQ: $\langle \underbrace{O(n\log n)}_{\text{предподсчёт}},\ \underbrace{O(1)}_{\text{ответ}} \rangle$

# Дерево отрезков

Умеет модификации

**Def**

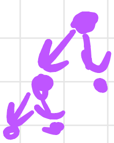




## Как хранить?

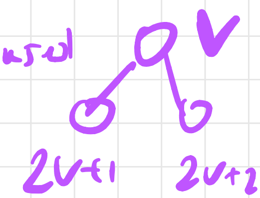
На ссылках

Node {
left
right
value
}

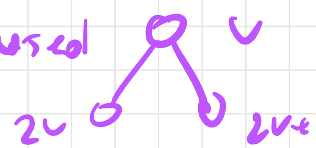


На массиве:

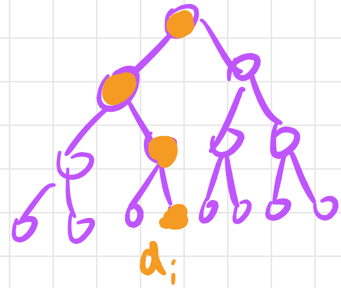

0-based




1-based



## Как поменять $a_i$?



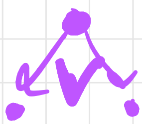

$O(\log n)$

$\Sigma$

## Как найти $\min\limits_{i=l..r} a_i$

node_l [      node_r )

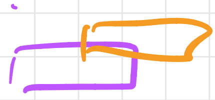
$[l; r)$



1) $[node\_l;\ node\_r) \subseteq [l; r)$

2) $[node\_l;\ node\_r) \cap [l; r) = \varnothing$

3) Нетрив. пересечение.



# Пример реализации

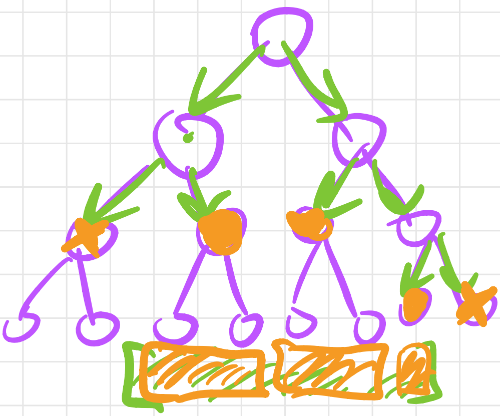
```
def get(node, node_l, node_r, l, r): # [node_l; node_r), [l, r)
    if l <= node_l and node_r <= r: # вершина целиком внутри запроса
1       return tree[node]

2   if node_r <= l or r <= node_l: # вершина целиком снаружи запроса
        return ~~X~~ +∞ # нейтральный элемент

    mid = node_l + (node_r - node_l) // 2
3   # вершина 2 * node + 1 отвечает за [node_l, mid)
    # вершина 2 * node + 2 отвечает за [mod, node_r)

    return min(get(2 * node + 1, node_l, mid, l, r),
               get(2 * node + 2, mid, node_r, l, r))
```

$get(0, 0, n, l, r)$

root

$a_0 \ldots a_{n-1}$

полуинтервал запроса

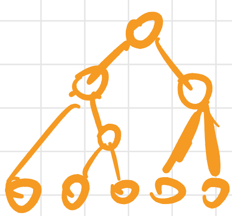


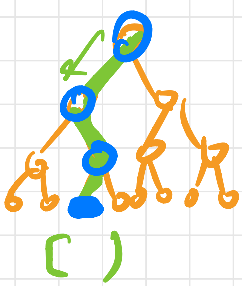
```python
def update(node, node_l, node_r, i, new_value):
    if node_r == node_l - 1: # лист
        assert i == node_l
        tree[node] = new_value
        return

    mid = node_l + (node_r - node_l) // 2

    if i < mid:
        update(2 * node + 1, node_l, mid, i, new_value)
    else:
        update(2 * node + 2, mid, node_r, i, new_value)
```

$O(\log n)$

tree[node] = min (tree[2 node + 1], tree [2 node + 2])

recalc





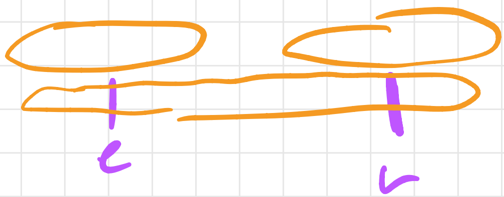
update(0, 0, n, i, a_i)

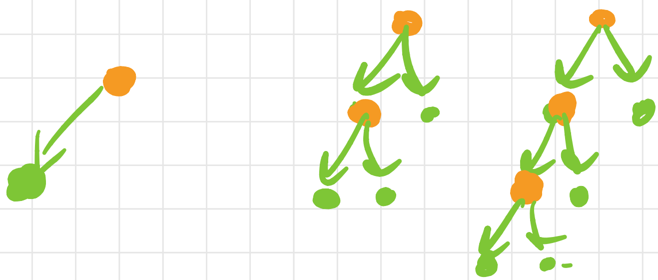
root [ )

Утв: get() работает за $O(\log n)$

Утв: get на каждом уровне дерева отрезков смотрит в не более 4 вершин.

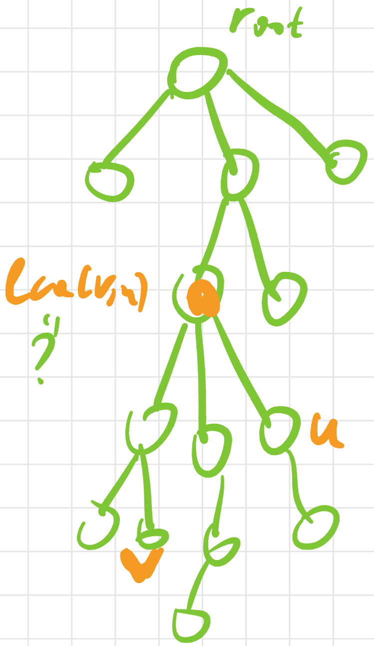
<u>Док-во:</u> $\leq 2$ интересные вершины на уровне



# LCA

(least common ancestor)



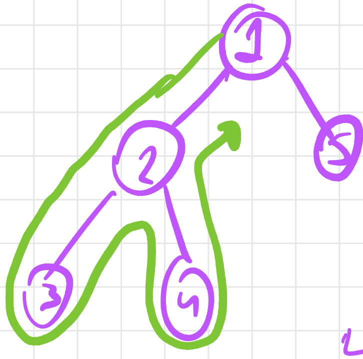

Решение LCA через сведение к RMQ

(range min query)

RMQ: $\langle O(n \log n), O(1) \rangle$ – Sp. Table

$\langle O(n), O(\log n) \rangle$ – дер. отрезков

$\langle O(n), O(1) \rangle$ ?



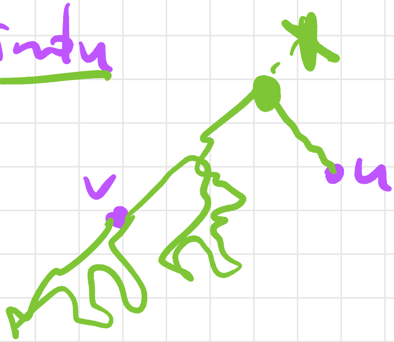

Эйлеров обход

$$\underbrace{1\,2\,3\,2\,4\,2\,1\,5\,1}_{O(n)}$$

<u>def</u>: $in_v$ – позиция $v$ в эйл. обходе

<u>Утв</u>: $lca(v,u)$ = наименее глубокая в-на на отрезке между $in_v$ и $in_u$

<u>Д-во</u>:



LCA $\to$ RMQ

$\langle depth_v, v \rangle$

$S(l; r) = \{l \dots r-1\}$

min на отрезке,

то это то же самое

что узел $u$: $depth_u \to \min$

RMQ $\langle O(n \log n), O(1) \rangle$

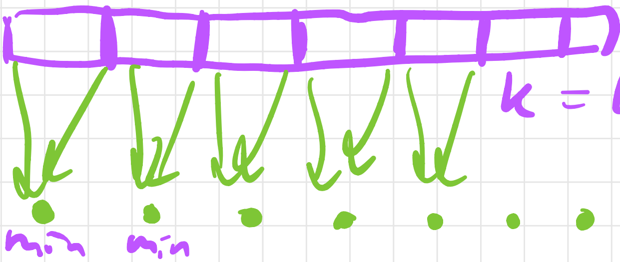
$\Downarrow$

LCA $\langle O(n \log n), O(1) \rangle$.

# RMQ за $\langle O(n), O(1) \rangle$

S.T: $\langle O(n \log n), O(1) \rangle$

Кусочков: $n/k$

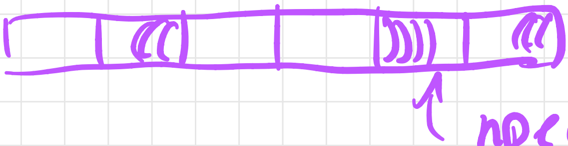



$k = \log_2 n / 3$

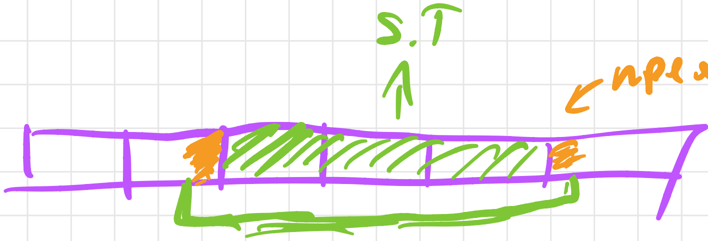
Построим на $n/k$ эл-тов Sp. Table

$$\frac{n}{k} \log \frac{n}{k} = O\left(\frac{n}{\log n} \cdot \log \frac{n}{\log n}\right) = O(n)$$

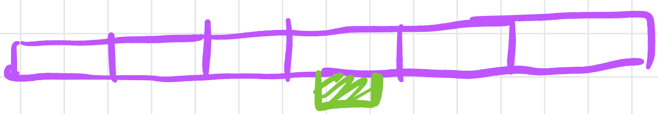
— может ответить на полные запросы



префиксные и
суфф min
на каждом кусочке



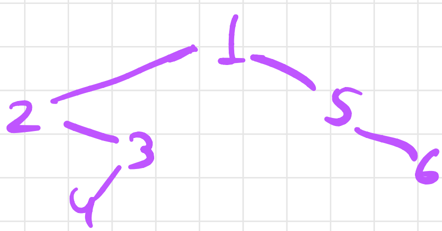

Как ответить на такие запросы ???



LCA → RMQ
RMQ → LCA

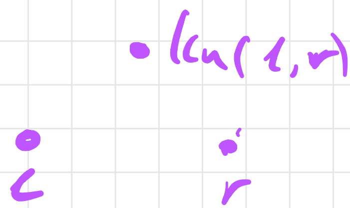
4-русские

$\boxed{2\ 4\ 3\ 1\ 5\ 6}$ → дерево.

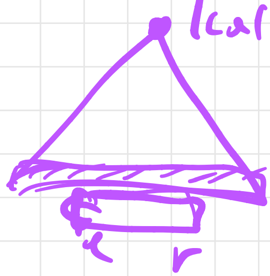


Утв: $\min\limits_{i=l..r} a_i = a_{lca(l,r)}$

Д-во: ① $lca(l,r) \in [l; r]$




② $a_{lca(l,r)}$ — min на $[l,r]$.



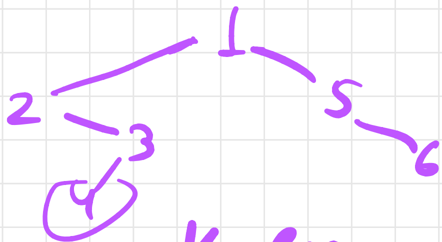

lca(l,r) - min во всём её поддереве

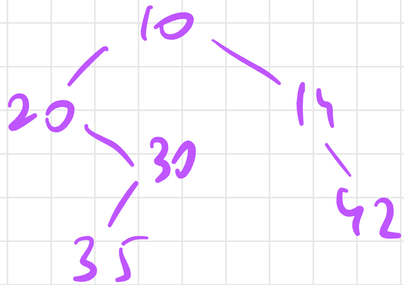
$\square$

Для решения RMQ нам не хватало научиться делать RMQ на кусочках.

RMQ → LCA

Нахождение LCA зависит только от структуры дерева.




$k$ вершин.

<u>Утв</u>: деревьев такого вида $\leq C_k$.

<u>Утв</u> деревьев такого вида $\leq 4^k = 2^{2k}$

$$C_k \leq 4^k$$

предподсчёт за $4^k \text{poly}(k)$ —

— lca всех возможных пар

во всех возм. деревьях

$$4^k \text{poly}(k) = 2^{2k} \text{poly}(k) = 2^{2\log_2 n/3} \text{poly}(n) =$$

$$= O(n^{2/3} \log^t n)$$

$$= O(n)$$

$$k = \frac{1}{3} \log_2 n$$

$$\frac{1}{4} \log_2 n$$

$$2^{2k} \operatorname{poly}(k)$$

$$= n^{1/2} \operatorname{poly}(k)$$