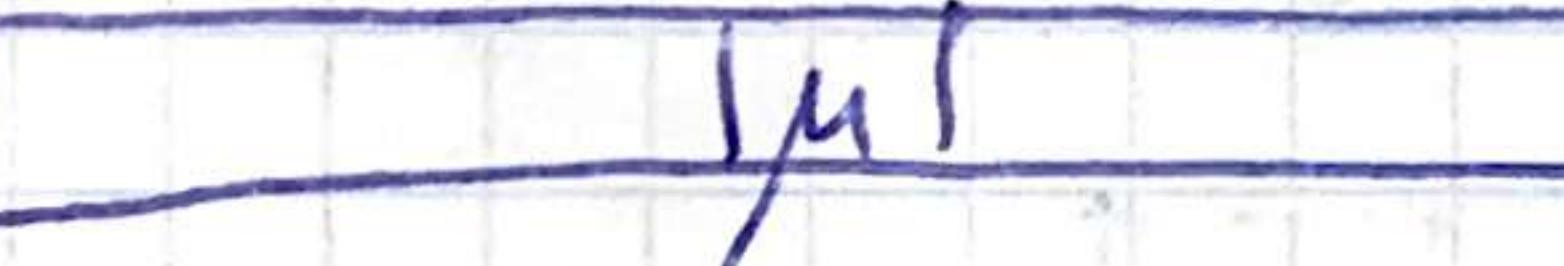


~~2 2 3 3 3 2 ②~~

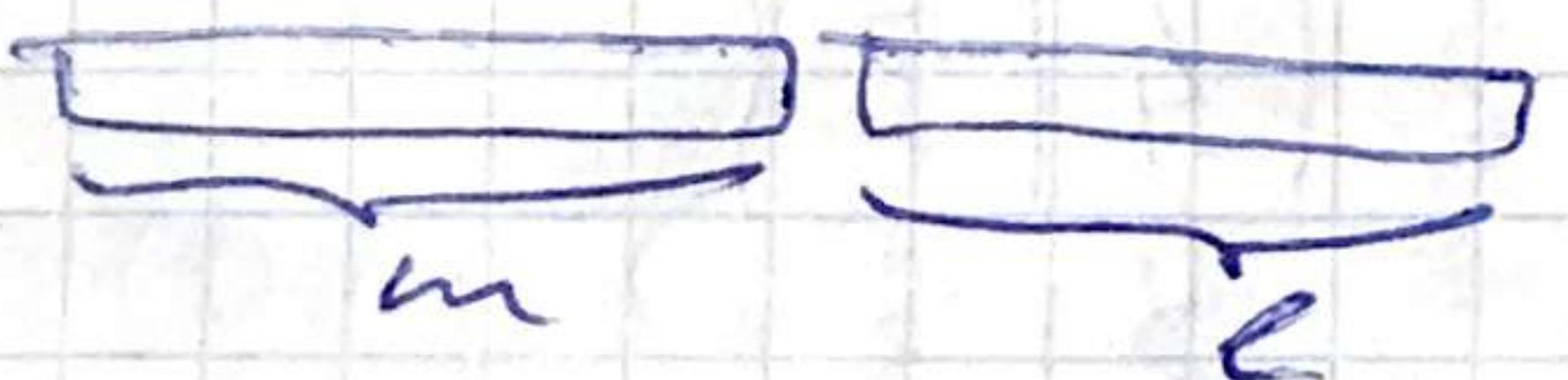
~~② 2 3 ③~~

i)   $n$ -tereftest,  $\mu$   
 $\mu$ -vorfälle

Kaxodgerne seeguanan ze  $O(n)$

std::nth\_element

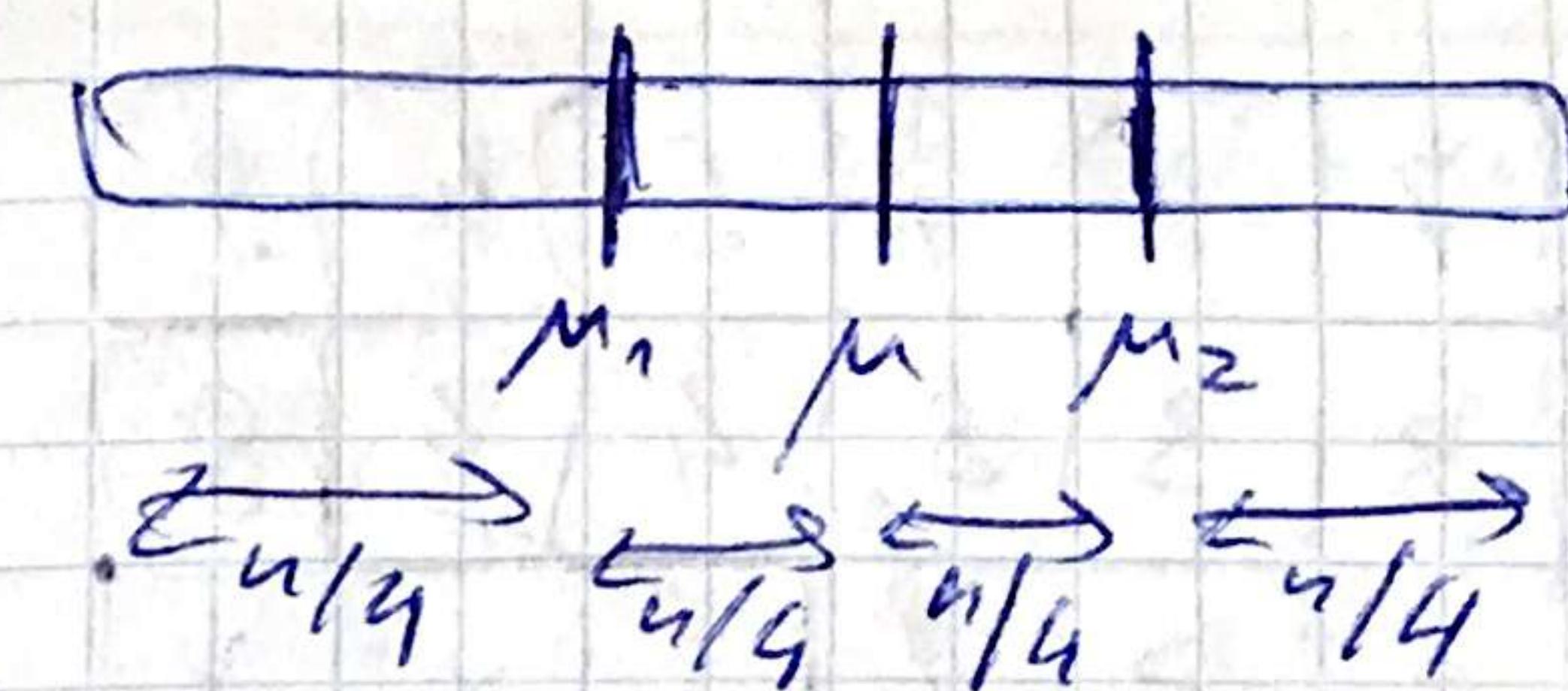
$k^{th}$  element



$m > k$

$m < k$

2)



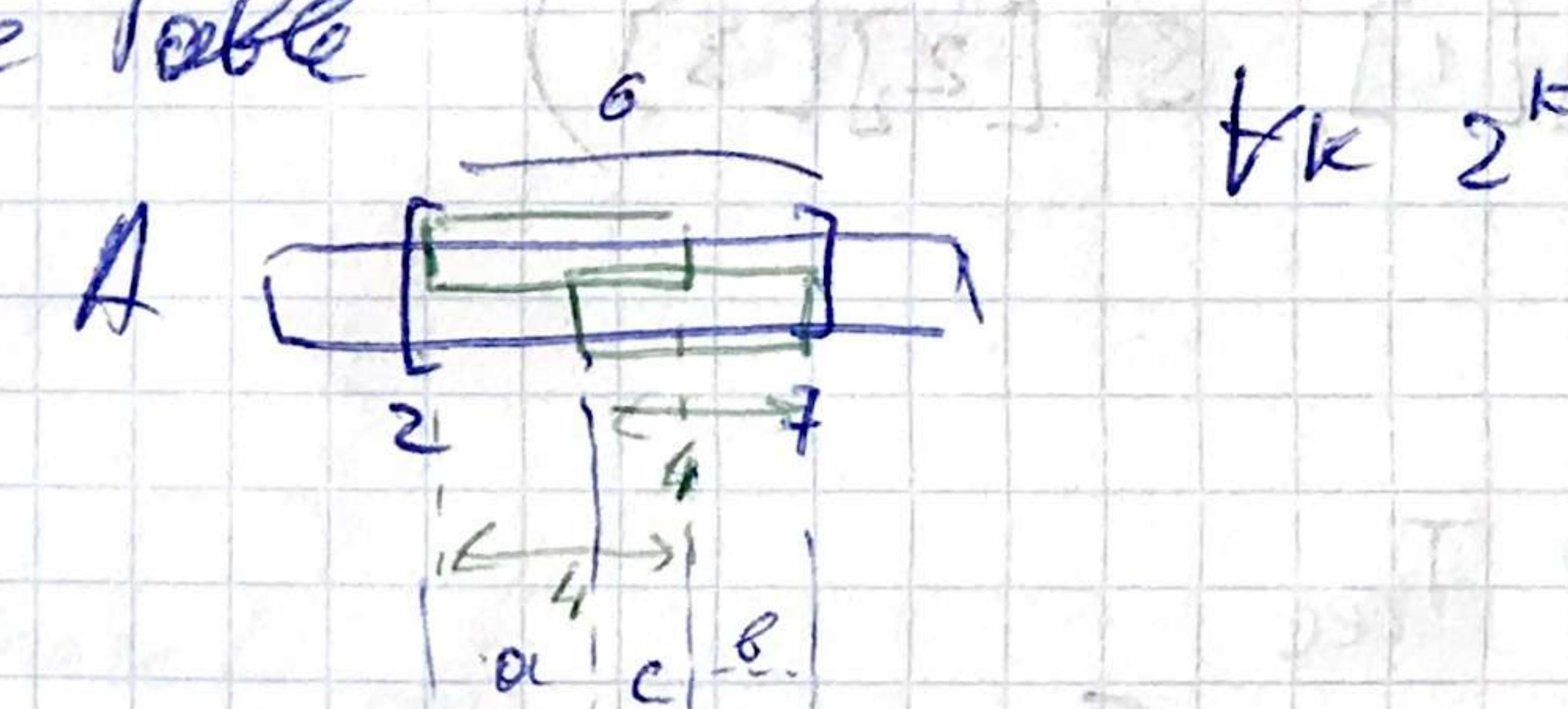
RMQ / RSQ

→ static  
→ dynamic



min & updatorefekte

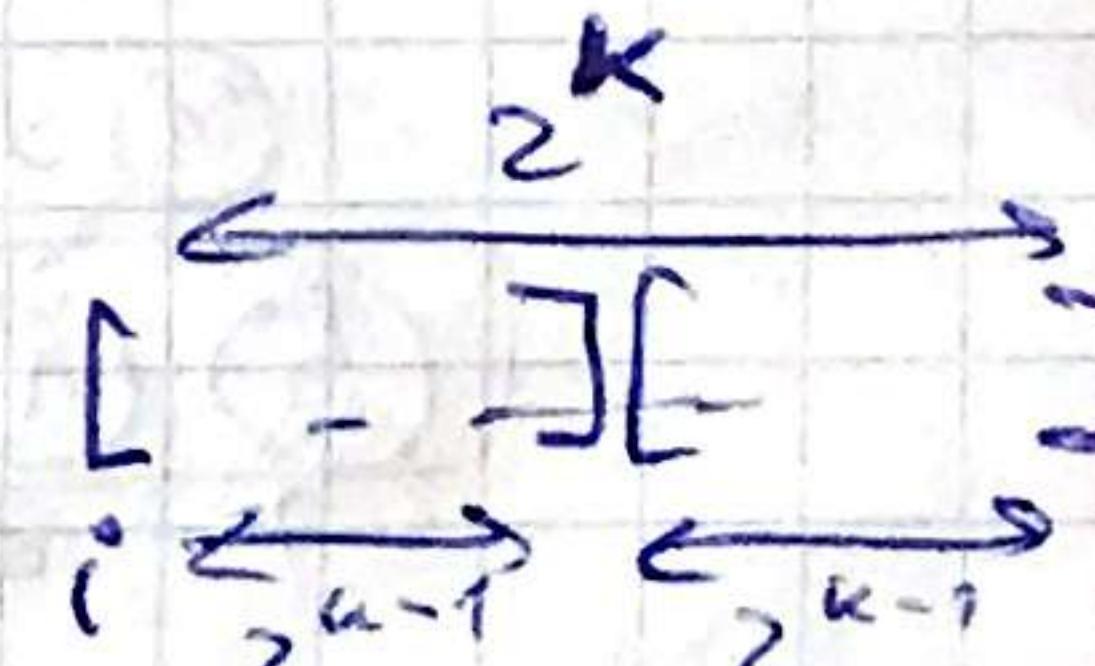
Sparse Table



$$ST[k][i] = \min(A[i, i+2^k-1])$$

$$ST[0][i] = A[i]$$

$$ST[k][i] = \min(ST[k-1][i], ST[k-1][i+2^{k-1}])$$



ST	9	1	4	2	7	3	7	5	4	9
$k=0$	1	1	1	2	3	3	5	4	4	X
$k=1$	1	1	2	2	3	3	5	4	4	X
$k=2$	1	1	2	2	3	3	3	4	X	X
$k=3$	1	1	2	X	X	X	X	X	X	X

$[1 \dots 6]$

$[1 \dots 4]$

$[3 \dots 6]$

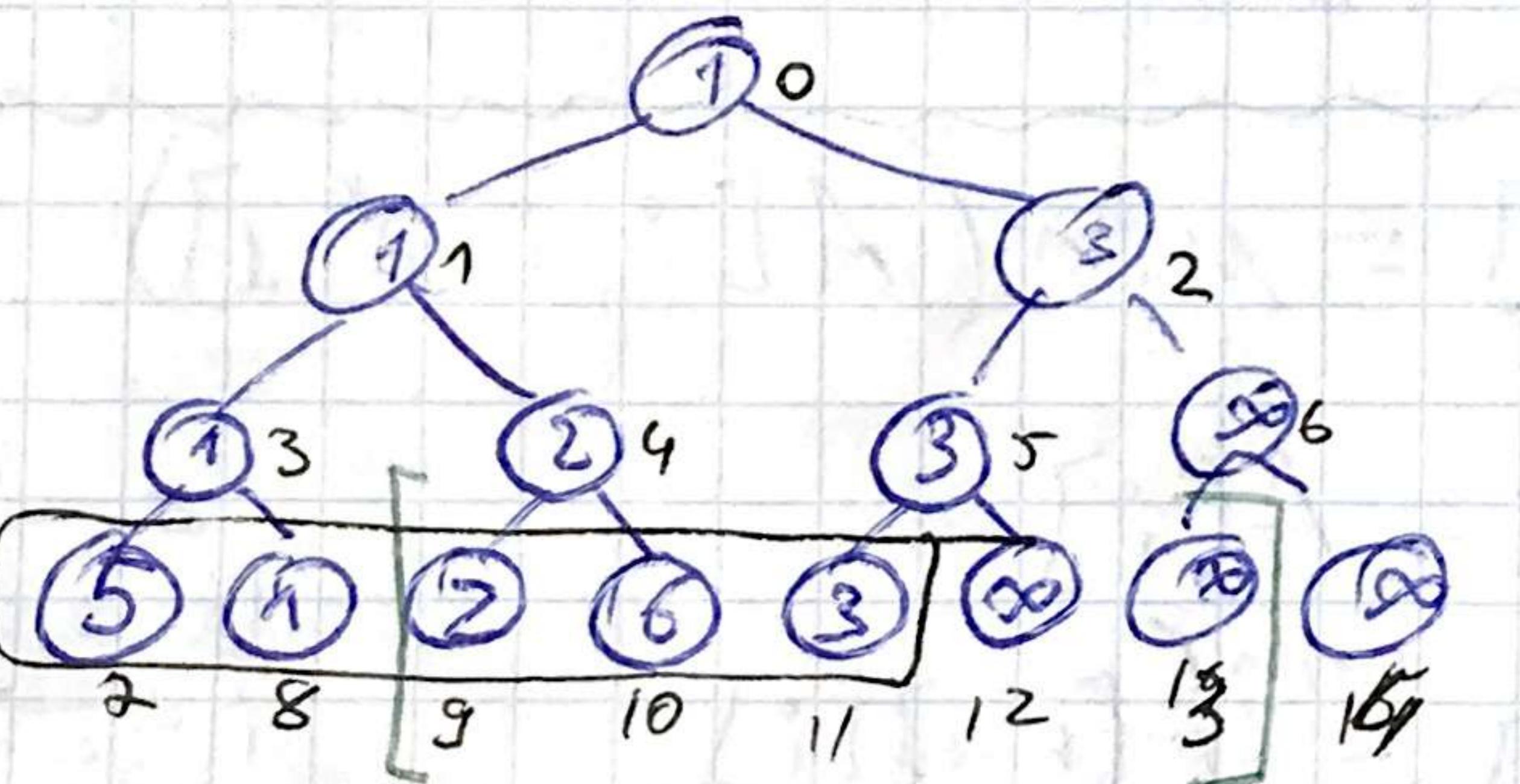
$[i, j]$

$[i, i+2^k]$

$[j-2^k, j]$

$\min(ST[2][1], ST[2][3])$

Segment Tree



$\text{tree}[i:j] = \min(\text{tree}[2i+1], \text{tree}[2i+2:j])$

```

int Query(int left, int right) {
    left += tree.size() / 2;
    right += tree.size() / 2;
    while (left < right) {
        if (left == right) break;
        left_res = min(left_res,
                      tree[left]);
    }
}
  
```

```

    right = parent(right);
    if (right == left) break;
    right_res = min(right_res,
                     tree[right]);
}
  
```

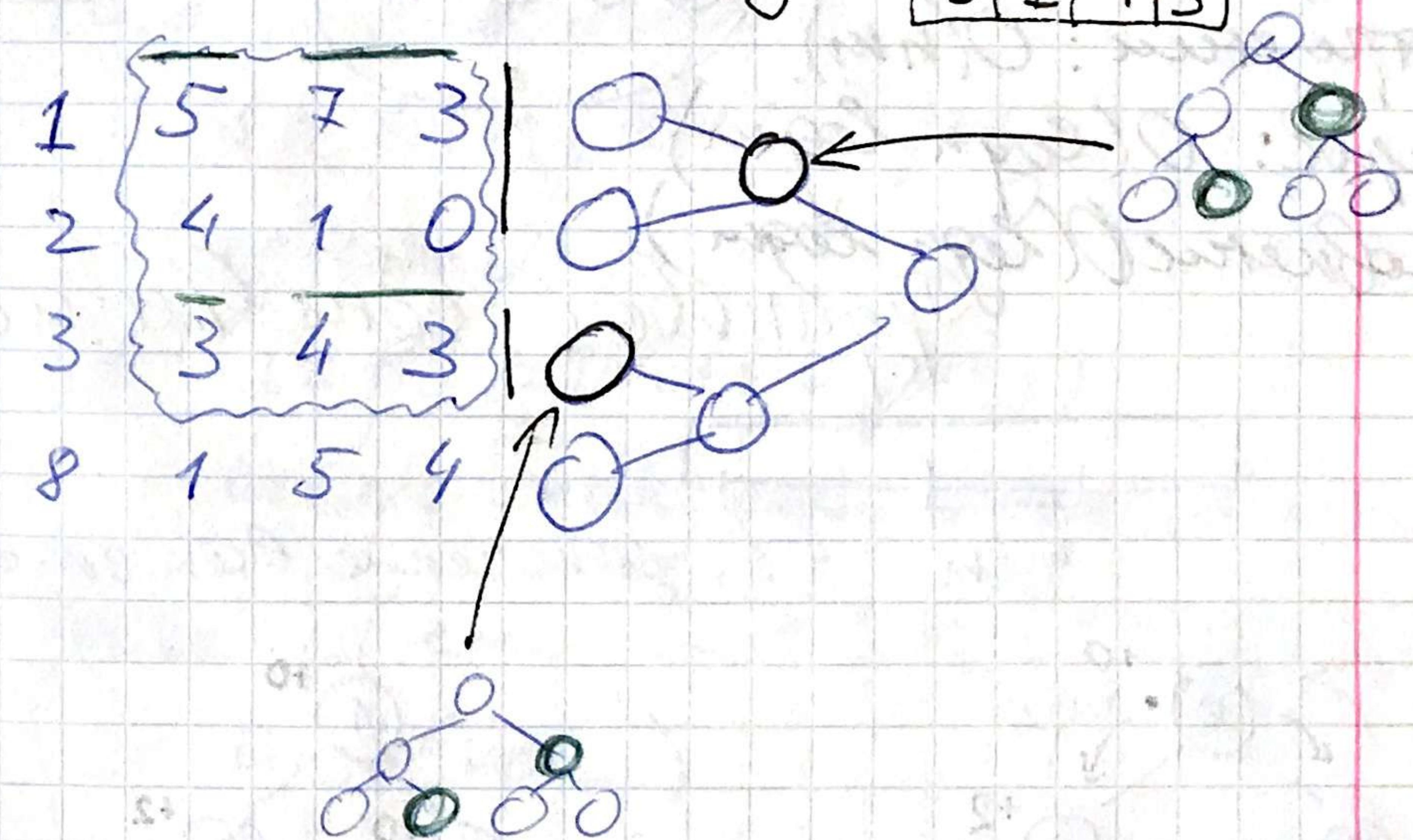
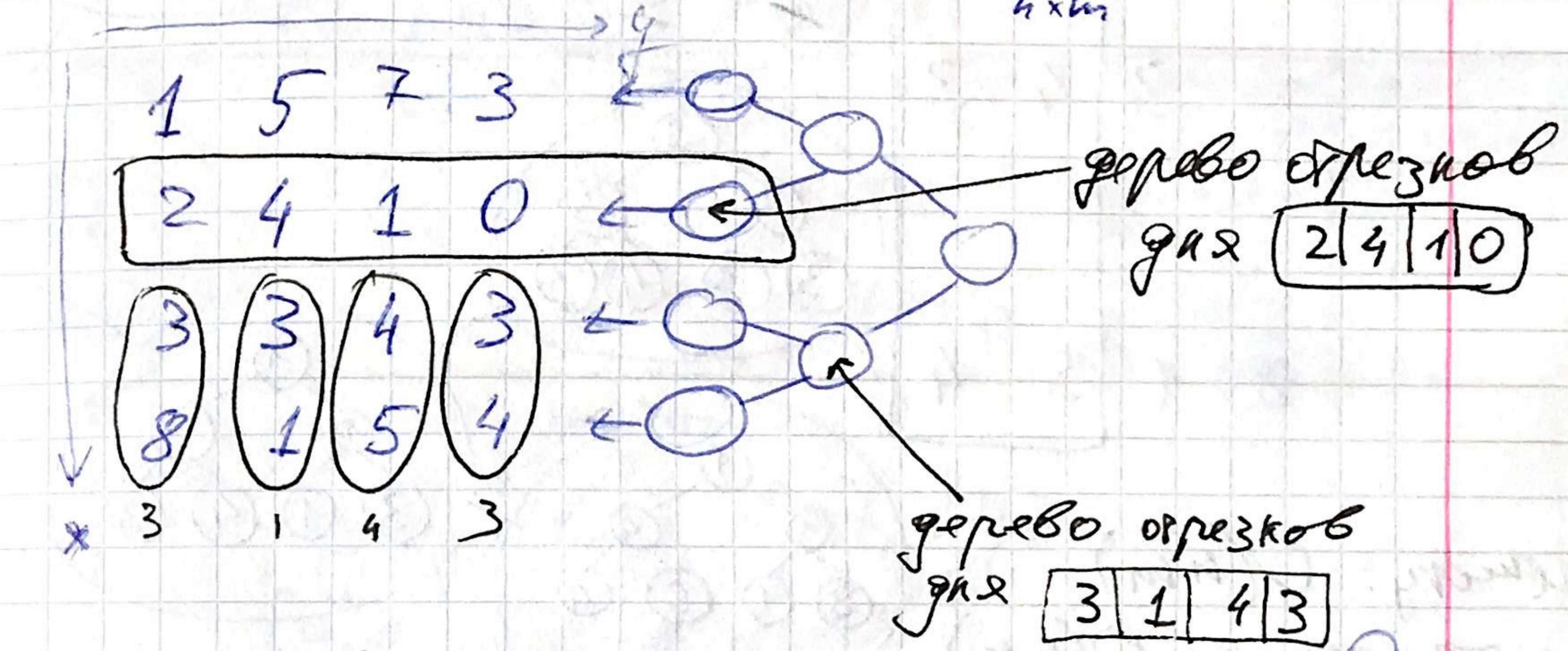
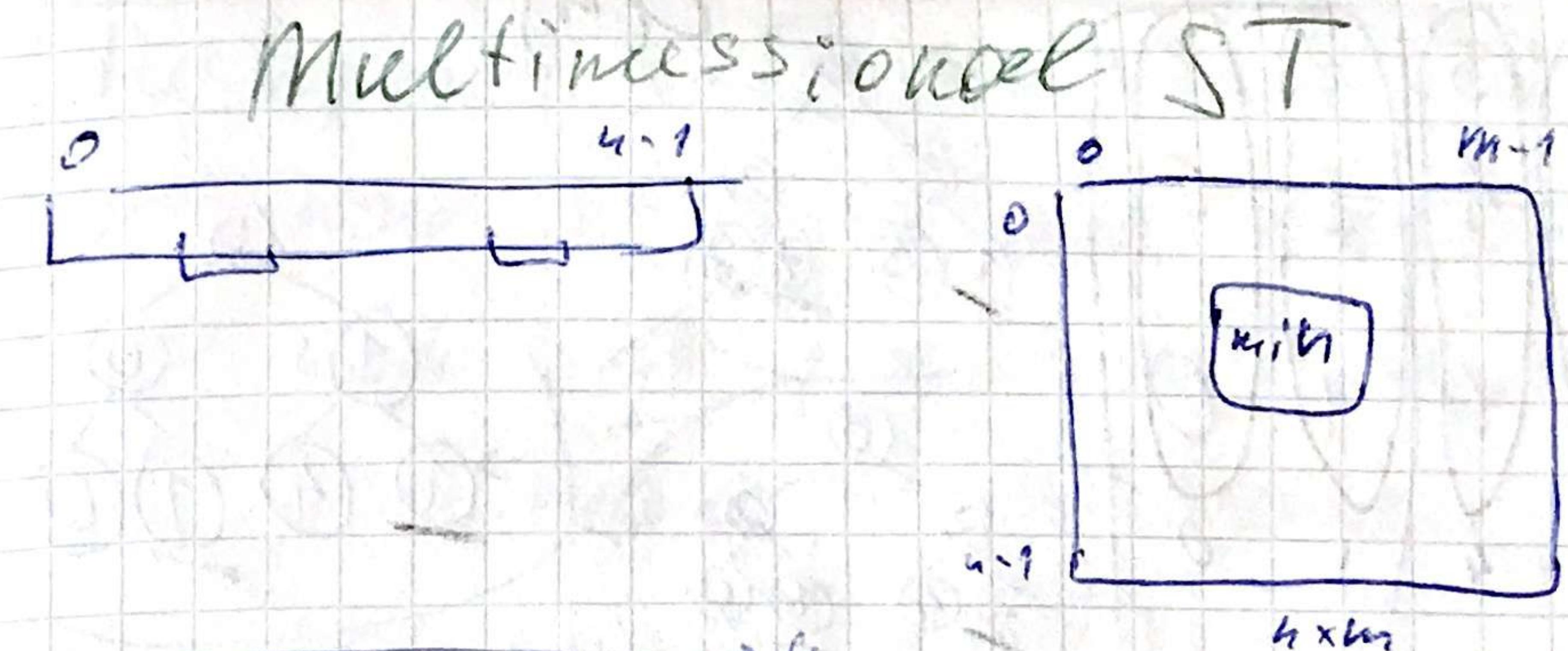
```

    right = parent(right - 1);
    if (left == right) break;
    left_res = min(left_res, tree[left]);
}
return min(left_res, right_res);
}
  
```

```

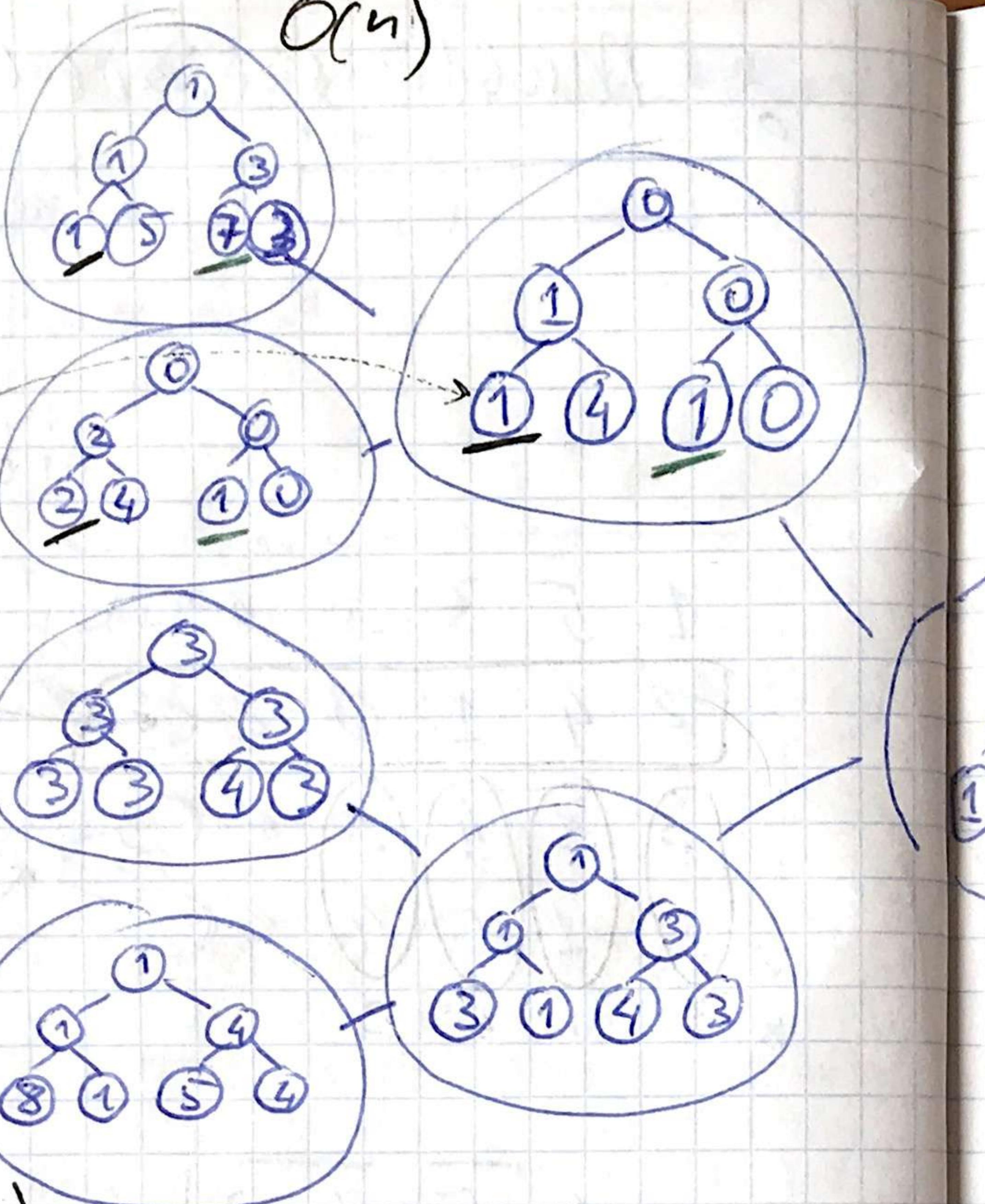
void Update (size_t idx, int new_value) {
    i += free_size / 2;
    tree[i] = new_value;
    do {
        i = parent(i);
        tree[i] = min(tree[left[i]], right[i]);
    } while (i != 0);
}

```





$O(n)$



Memory:  $O(nm)$

Построение:  $O(nm)$

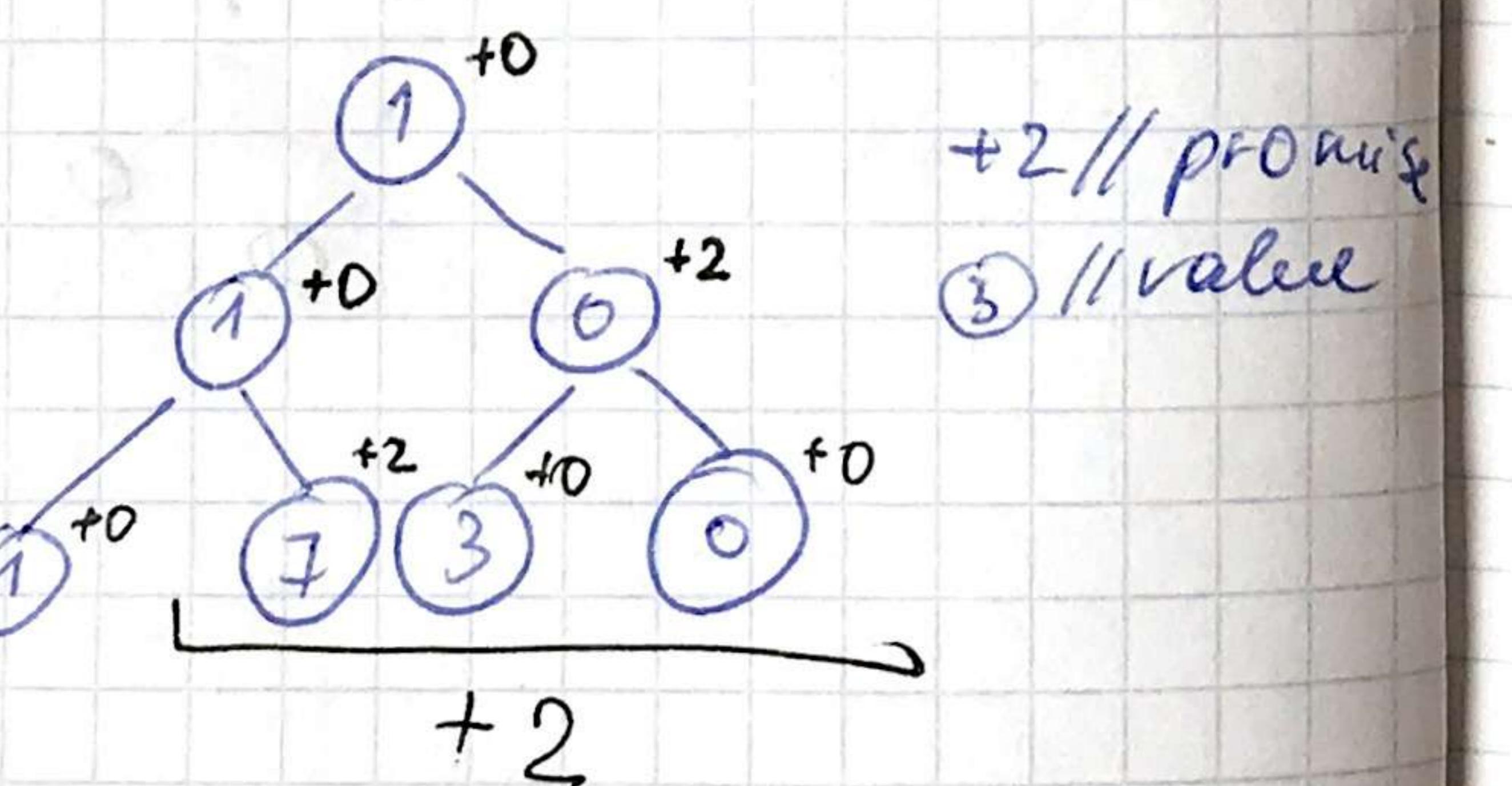
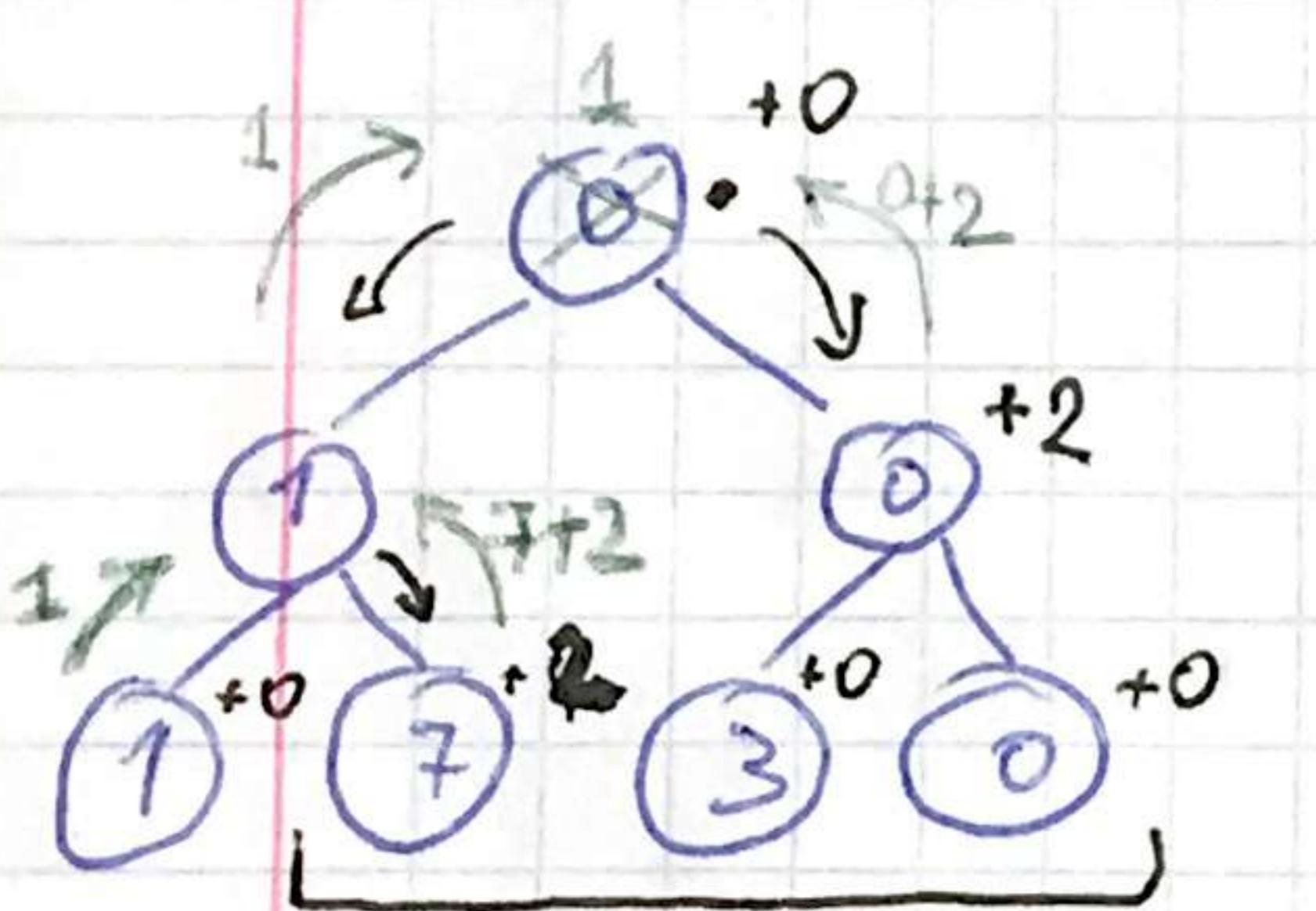
Запрос:  $O(\log n \log m)$

Одновременное обновление

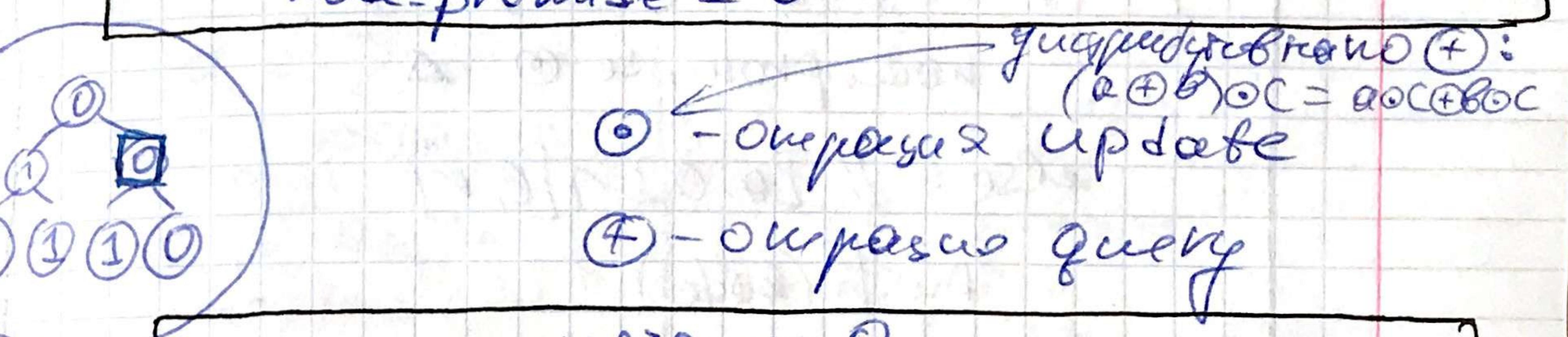
Параллельное одновременное обновление



min + 3, где  $a = 0, b = 3$



Push(node) : // параллельное обновление  
 $\text{node.left.promise} \odot = \text{node.promise}$   
 $\text{node.right.promise} \odot = \text{node.promise}$   
 $\text{node.value} \odot = \text{node.promise}$   
 $\text{node.promise} = \odot$



updateQuery()

Query(node, left, right):

$[a, b] = \text{node.borders}$

if  $([a, b] \not\subseteq [\text{left}, \text{right}])$

return 0;

if  $([a, b] \subseteq [\text{left}, \text{right}])$

return node.value  $\odot$  node.promise

else: (если  $[a, b] \cap [\text{left}, \text{right}]$ )

Push(node)

return Query(node.left) + Query(node.right)

Query(node.right)

Update(node, l, r;  $\Delta$ ):

$[a, b] = \text{node\_borders}$

if  $([a, b] \cap [l, r])$

return

if  $([a, b] \subseteq [l, r])$

node.promise  $\Theta = \Delta$

else: //  $[a, b] \cap [l, r]$

push(node)

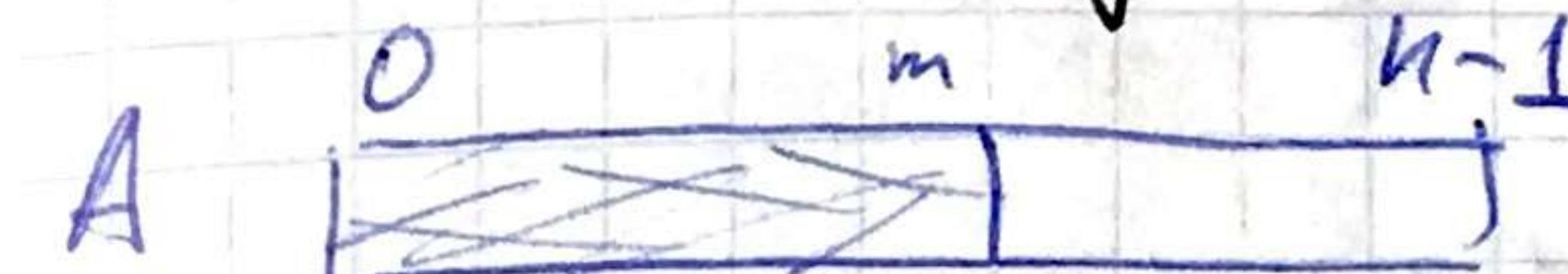
update(node.left, l, r,  $\Delta$ )

update(node.right, l, r,  $\Delta$ )

node.value = Query(node.left)

⊕  
Query(node.right)

Derevo Perbaiki  
(Binary Indexed Tree)  
dynamic RSQ



$$\sum_{k=0}^m \alpha_k = \sum_{k=F(m)}^m \alpha_k + \sum_{k=F(m)+1}^{F(m)-1} \alpha_k + \dots + \sum_{k=F(F(\dots)-1)+1}^{F(F(F(\dots)-1)-1)} \alpha_k + \dots$$

T-masalah.

$$T[i] = \sum_{k=F(i)}^i \alpha_k,$$

$F: N \rightarrow N$

$F(i) \leq i$

Query(m): //  $[0 \dots m]$

res = 0

for ( $i=m; i \geq 0; i=F(i)-1$ )  
 $res += T[i]$        $i \& (i+1)$

return res

Query(l, r) //  $[l \dots r]$

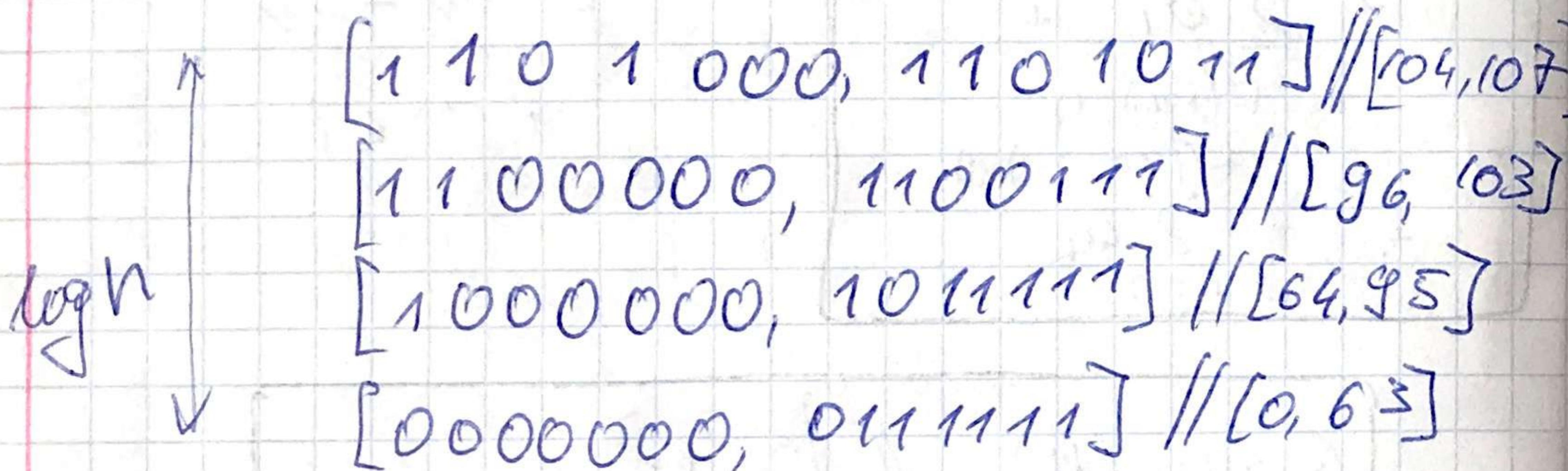
return  $Q(r) - Q(l-1)$

$F(m)$  - функция, которая засчитывает  
последовательность в соответствии  
с некоторыми правилами

$[0, 107]$

$$107_{10} = 11010 \underline{11}_2$$

$$F(107) = 1101000_2 = 104$$



$$F(m) = m \otimes (m+1) \quad \text{- функция Рекурсии}$$

Update: Допустим хранить излишество  $a_k$  на  $\Delta$  ( $a_k += \Delta$ )

Как корректны  $T[i:]$  при накоплении?

$$\text{тогда } \rightarrow F(i) \leq k \leq i \quad (T[i:] = \sum_{j=F(i)}^i a_j)$$

Построение безрасходной последовательности  $i$ , которую можно однозначно

$$i_0 = k$$

Допустим есть  $i_m$ , хотим найти  $i_{m+1}$

$$\begin{aligned} i_m &= \overline{x} 01 \dots 1_2 \\ i_k &= \overline{x} 0? \dots ?_2 \\ F(i_m) &= \overline{x} 00 \dots 0_2 \end{aligned}$$

~~$x$~~

$$i_{m+1}$$

Но что же  $\overline{x}_{m+1} > \overline{x}_m$ ? - Нет

$$i_{m+1} = \overline{x}_{m+1 \dots \infty}$$

если  $x_{m+1} > x_m$ ,

то это несложно.  
но требуется  $i$ ,  
т.к. неизвестно

$$F_{i_{m+1}} = \overline{x}_{m+1}$$

$$k = \overline{x}_m$$

следующее значение  $i_{m+1} = \overline{x}_{m+1}$ .

То есть надо убрать из  $i$   $\overline{x}_m$

$$i_{m+1} = \overline{x}_m 1 (1 \dots 01 \dots 1)$$

могут ли засчитать  $\overline{x}_m$ ? - Нет,

$$\text{т.к. } F(i_{m+1}) = \overline{x}_m 11 \dots 00 \dots 0 > k = \overline{x}_m 0 \rightarrow$$

$$\rightarrow i_{m+1} = \overline{x_m} 11\dots1$$

$$k = \overline{x_m} 0? \dots ?$$

$$F(i_m) = \overline{x_m} 00\dots0$$

$$x_m' < x_m$$

$\Rightarrow i_{m+1}$  накрещащ възима здравеен  
номера 0 или 1.  
(съдържащ  
надежда)

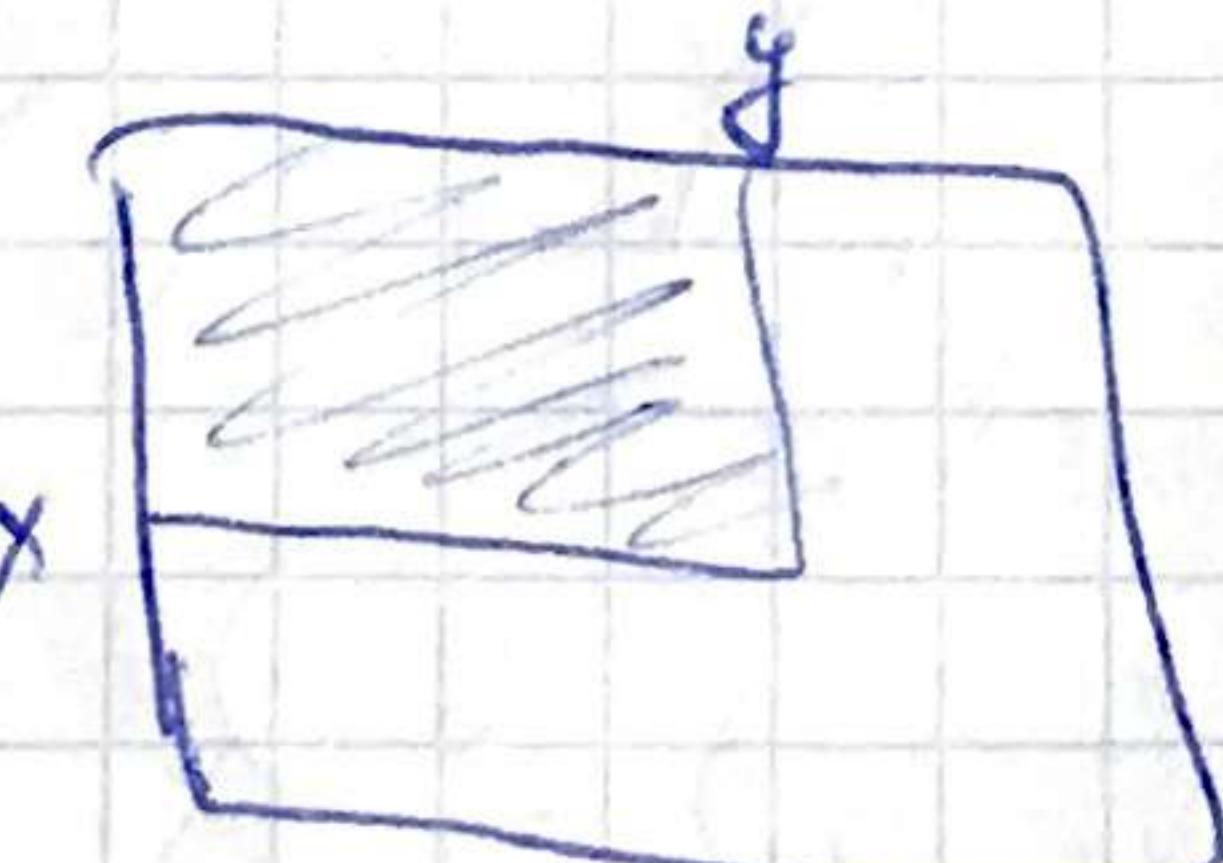
$$i_{m+1} = i_m | (i_{m+1} + 1), i_0 = k$$

Update ( $k, \Delta$ ):

```
for (i=k; i<n; i = i + 1)  
    T[i] += Δ
```

Общо съвръзка

$$T[i, j] = \sum_{x=F(X)}^X \sum_{y=F(y)}^y a[x, y]$$



Query ( $x, y$ ):

res = 0

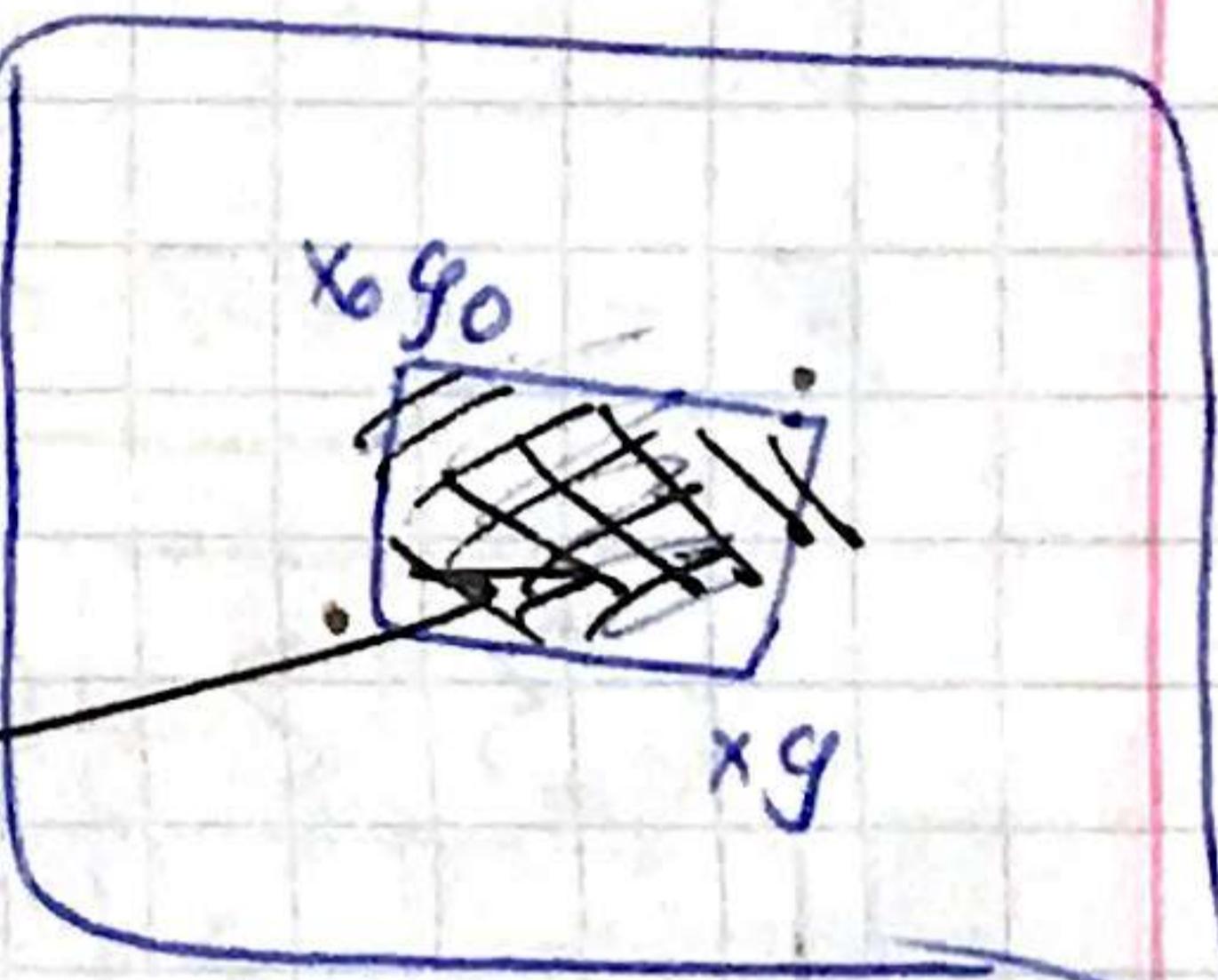
```
for (i=x; i>=0; i = i | (i+1))
```

```
    for (j=y; j>=0; j = j | (j+1))  
        res += T[i][j]
```

return res

$$Q(x, y) = Q(x_0, y) - Q(x_0, y_0)$$

$$+ Q(x_0, y_0)$$



Update ( $x, y, s$ )

```
for (i=x; i<n; i = i | (i+1))
```

```
    for (j=y, j<m; j = j | (j+1))
```

$T[i][j] += s$

⊕ - обновяване,  
изменение,  
обратуване

21.11.20

СеминарPrefixSumArray

Query ⊕	Update ◐
accogact. одобрение	-

SparseTable

Query ⊕	Update ◐
accogact. одобрение	-

Segment Tree

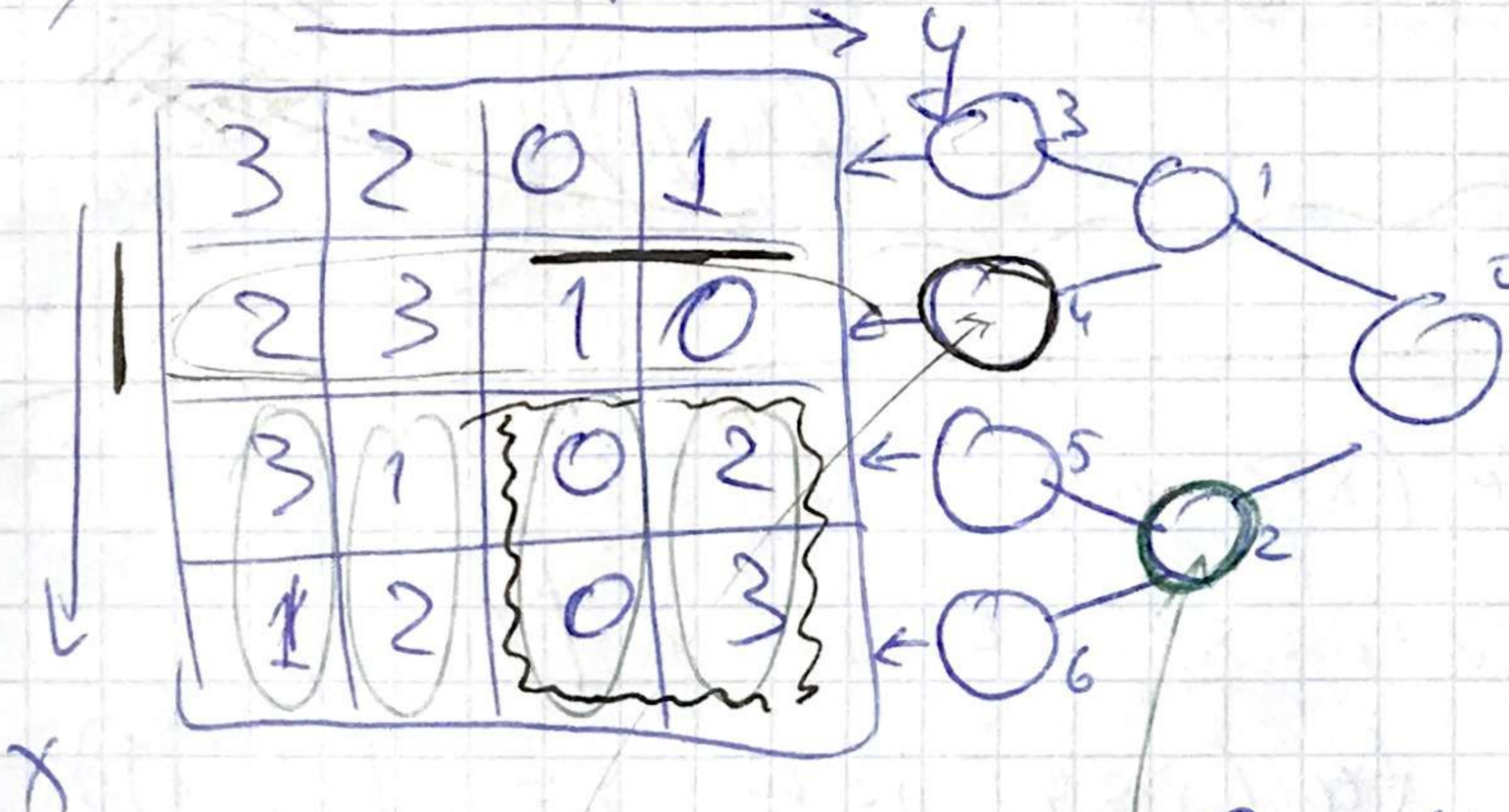
Query ⊕	Update ◐
accogact. одобрение	∅

Segment Tree (with GM)

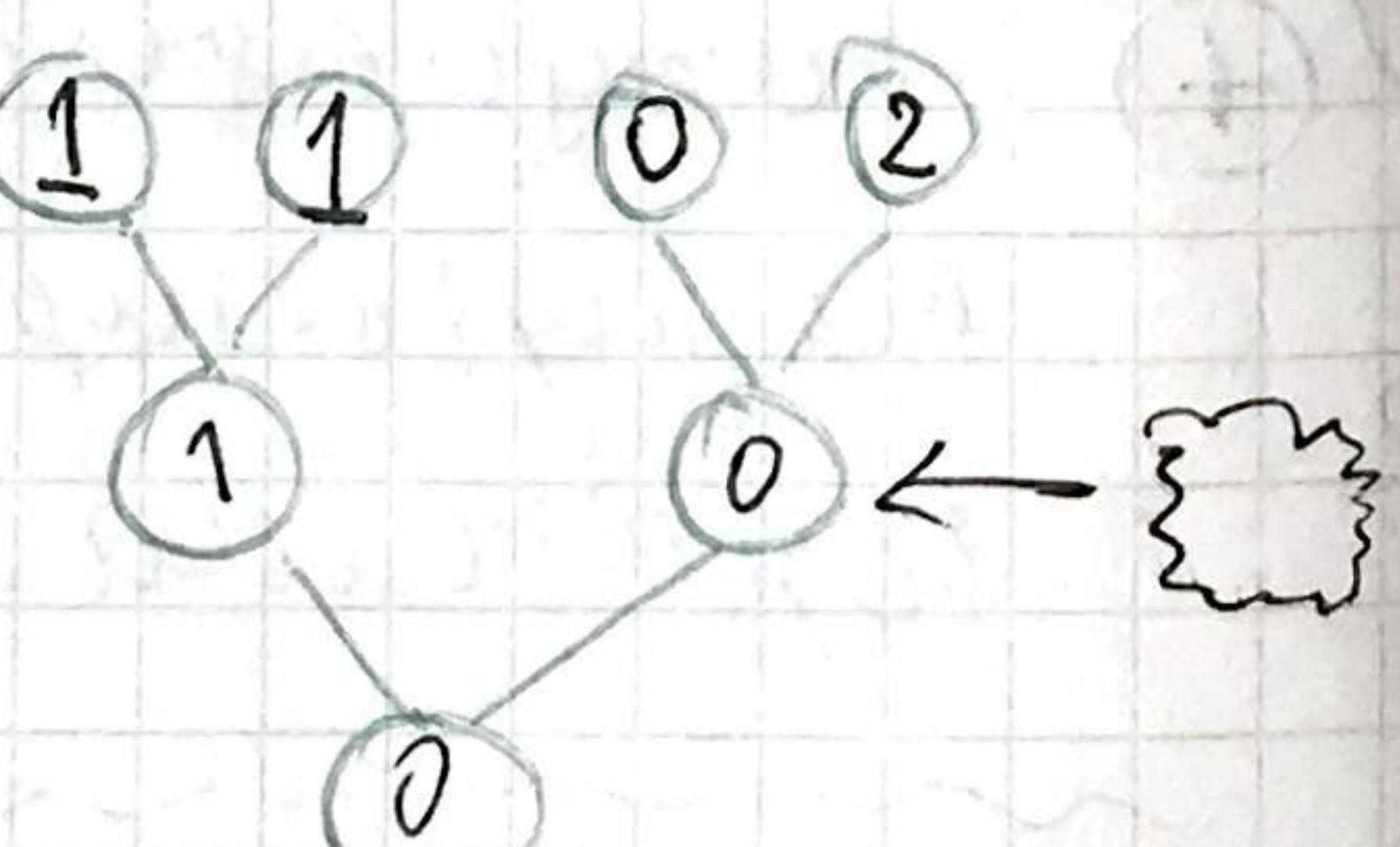
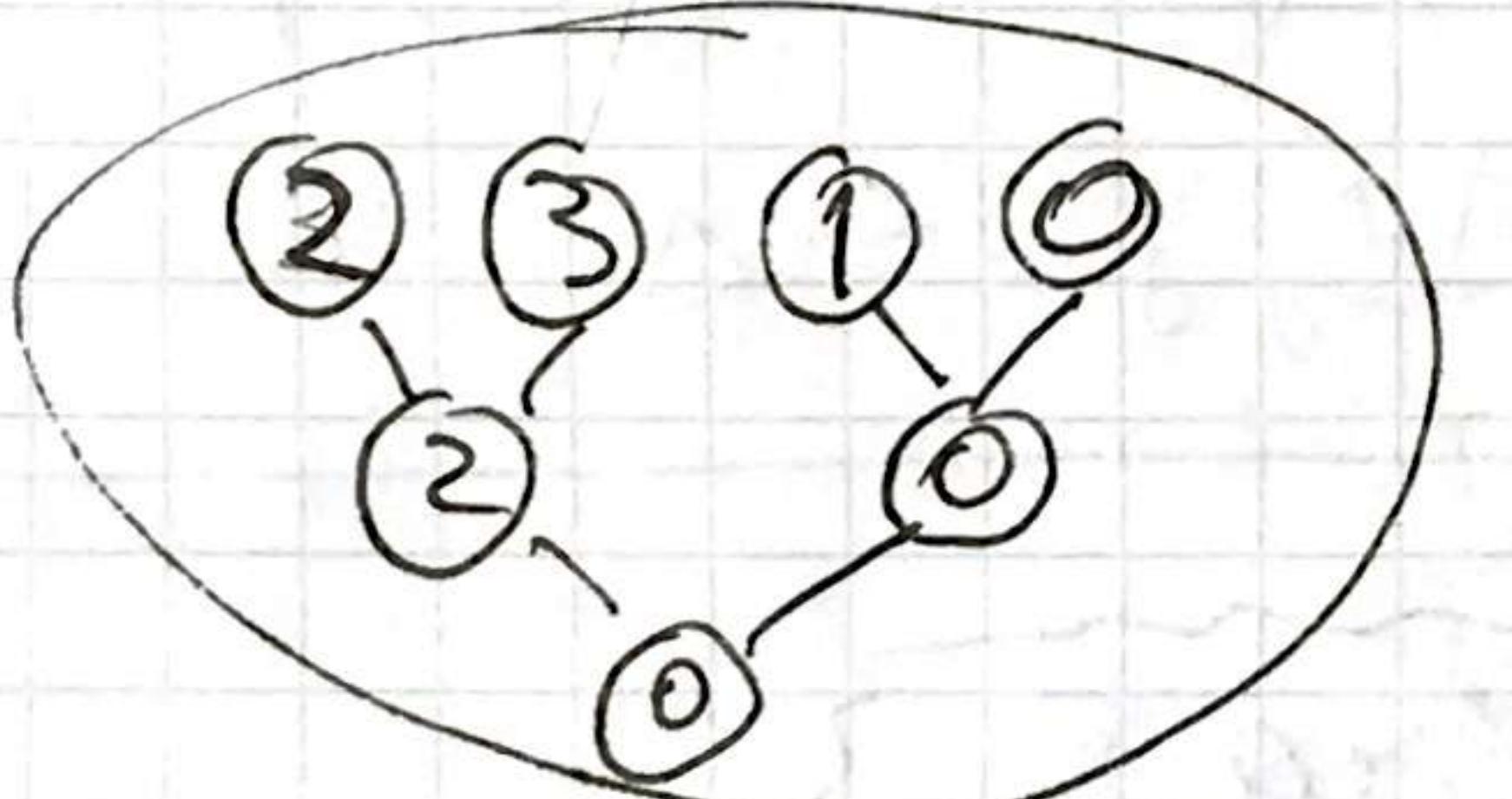
Query ⊕	Update ◐
accog., одобр.	accogact. одобрение

BIT (Binary Indexed Tree)

Query ⊕	Update ◐
accogact., одобрение	∅

 $\emptyset$  - нет оператора

Query([1, 1], [2, 3])



Building Time	Query Time	Update Time	Memory
$O(n)$	$O(1)$	-	$O(n)$
$O(n \log n)$	$O(1)$	-	$O(n \log n)$
$O(n)$	$O(\log n)$	$O(\log n)$	$O(n)$
$O(n)$	$O(\log n)$	$O(\log n)$	$O(n)$
$O(n \log n)$	$O(\log n)$	$O(\log n)$	$O(n)$

} static

} dynamic

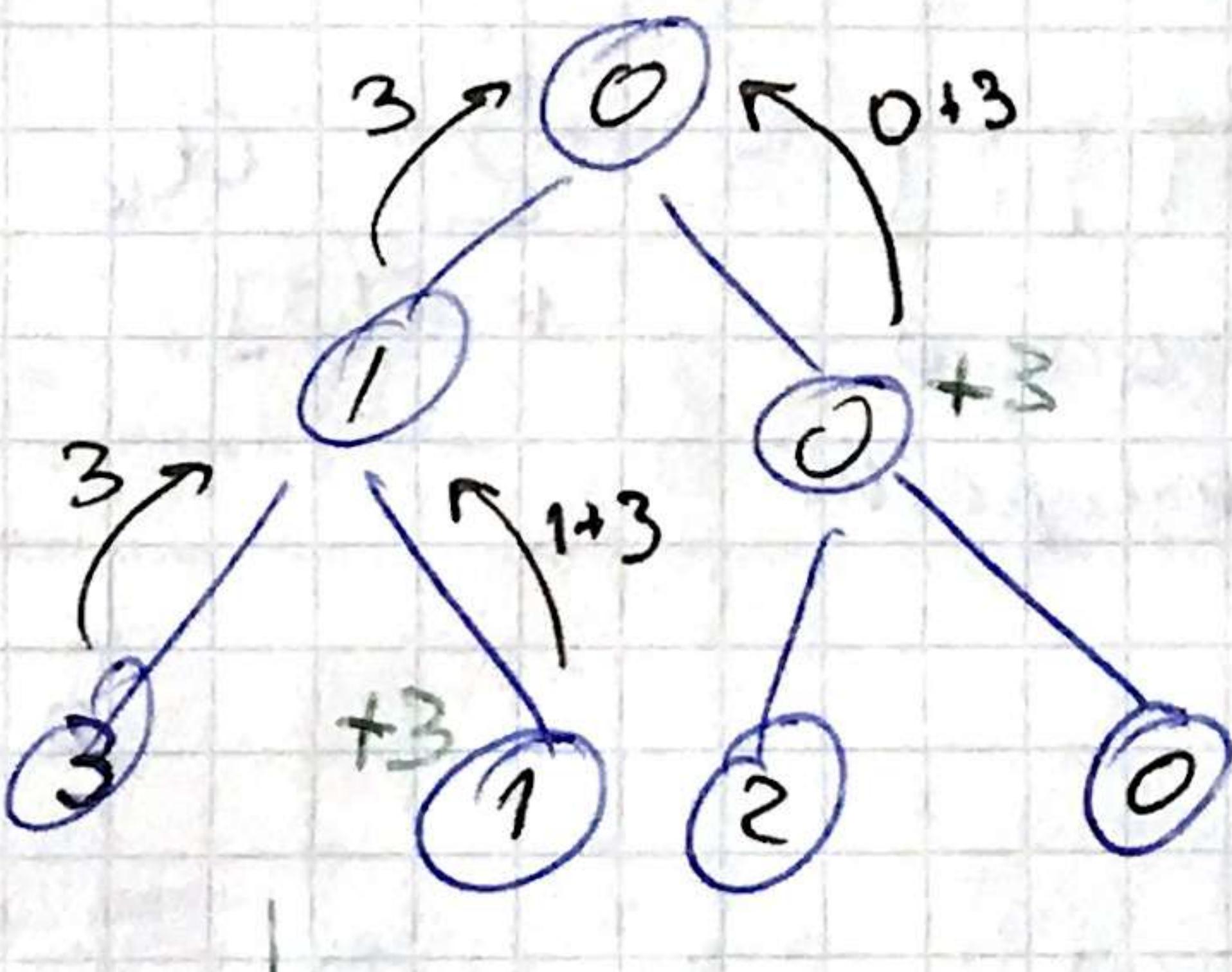
Тривиальные Операции

[compl.]

+2

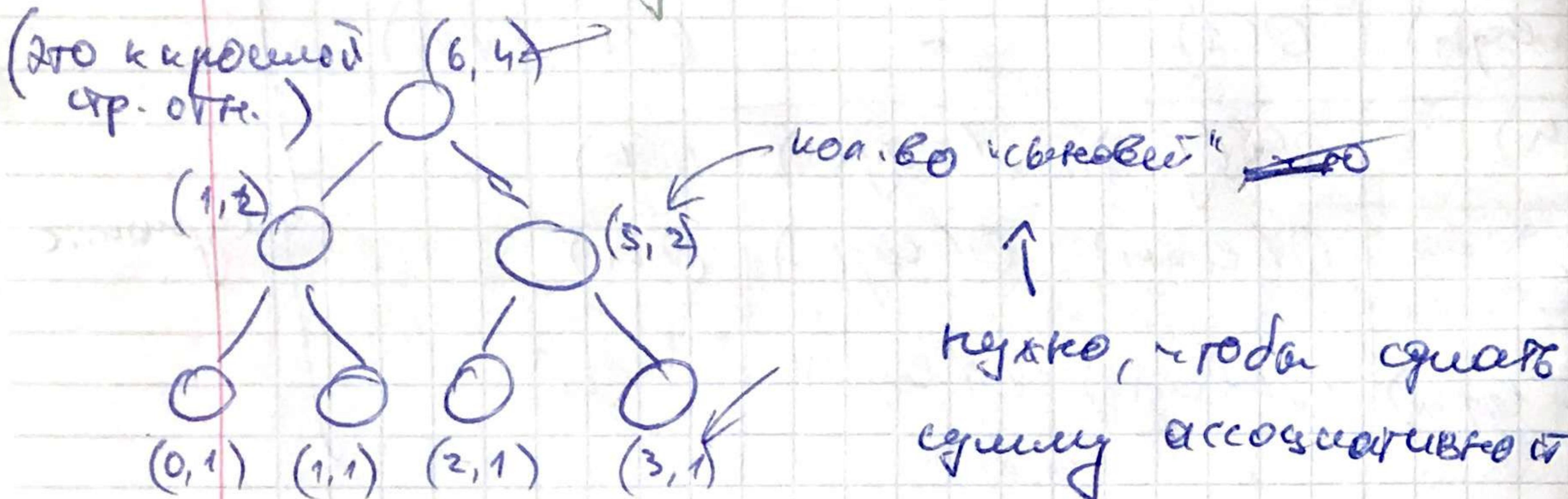
 $O(n)$ 

min



+3

# Derebe Perbeck (Binary Indexed Tree)



(Perbeck)

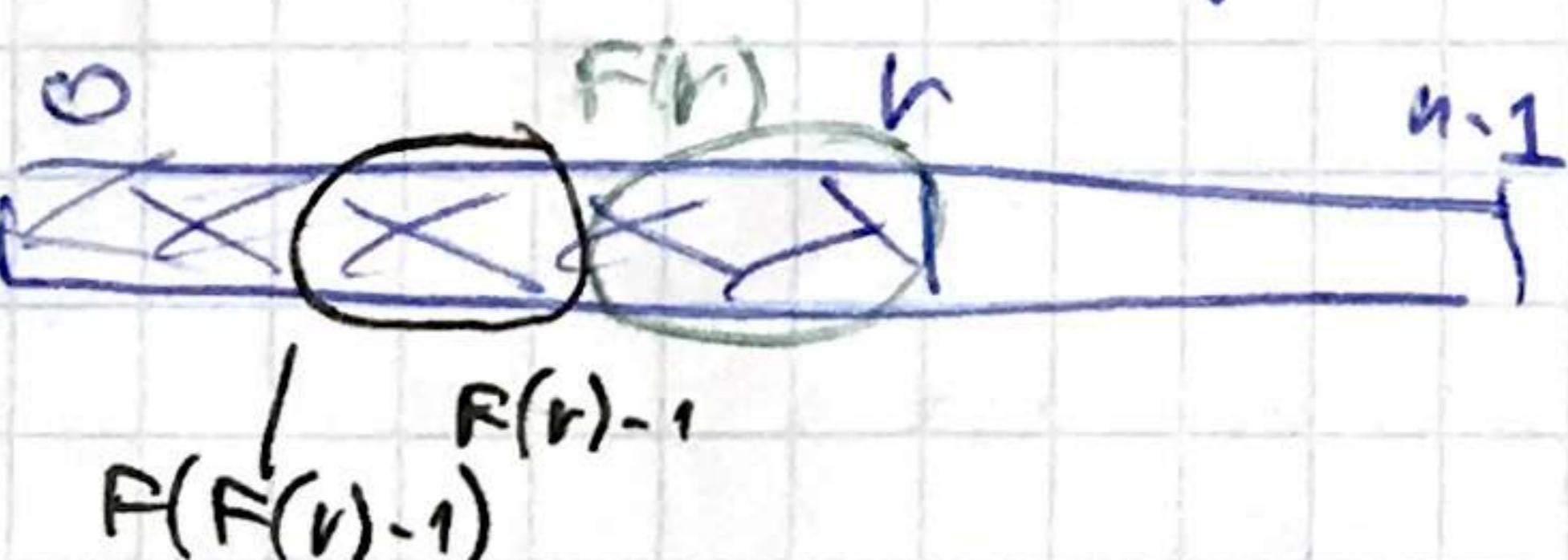
$$T[i] = \sum_{k=F(i)}^i a_k$$

нормал  
измерені



$$F(i) = i \& (i+1) - \text{для зберігання попередніх елементів}$$

$$\sum_{k=0}^r a_k$$



$$\sum_{k=0}^r a_k = T[r] + T[F(r)-1] + T[F(F(r)-1)] + \dots$$

$\log n$

Query( $r$ ):  $\oplus = +$  unsigned int

```
int res = 0
for (int i=r; i>=0; i = i & (i+1)-1)
    res += T[i]
return res
```

~~shakobae!~~

$F(i)$

Update( $a_k$ )  $\rightarrow T[i_0], T[i_1], \dots$

```
res += T[i]
while (i = F(i)-1)
    sum[i] += a_m
    i = i // (i+1)
```

$i_0 = k$

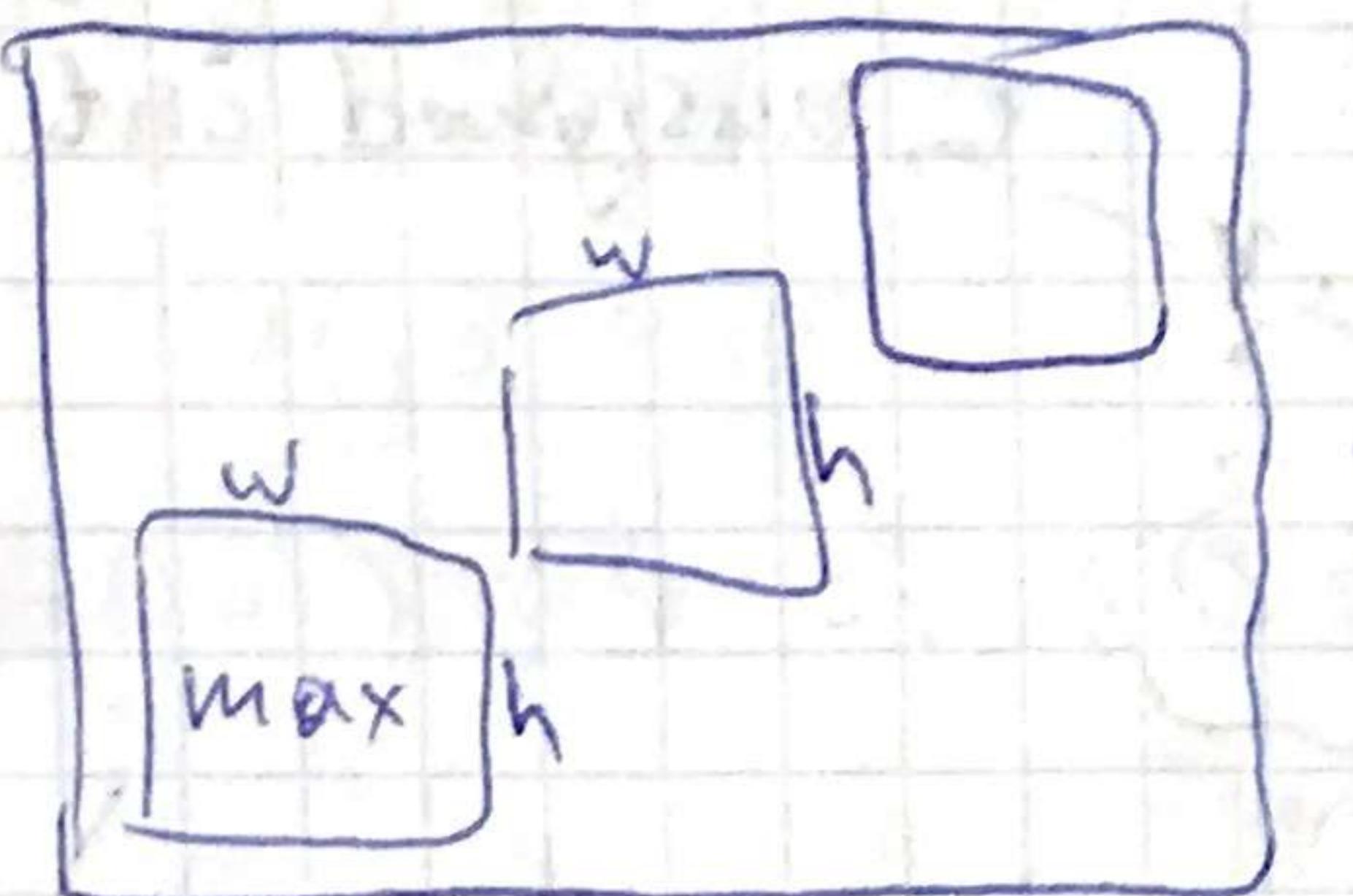
$i_m \rightarrow i_{m+1} : i_{m+1} = i_m // (i_m + 1)$

Update( $k, \Delta$ ):

```
for (int i=k; i<T.size(); i = i // (i+1))
    T[i] += Delta;
return
```

Build( $arr$ ):

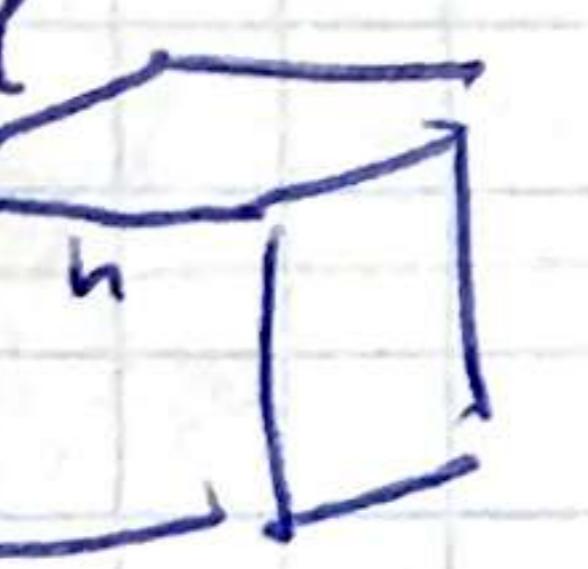
```
T = vector<int>(arr.size(), 0)
for i in range(arr.size()):
    update(i, arr[i])
```



static

$O(n \cdot m \cdot l)$  - носкоство

$O(1)$  - засыпок

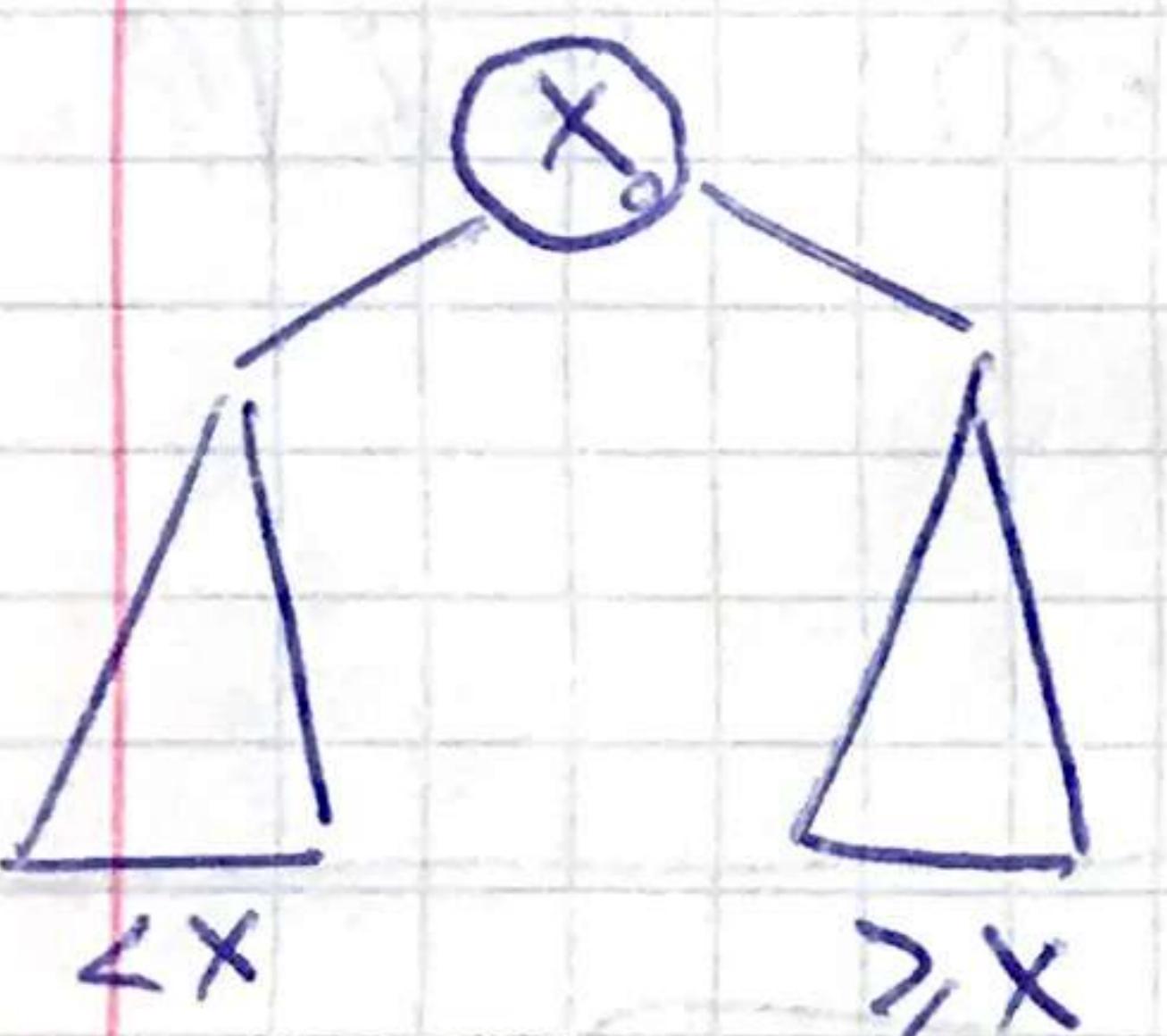


#Pinguinare Copywriting Datastructure

26.11.20

Бинарное дерево поиска  
(BST)

$x_1, \dots, x_n$

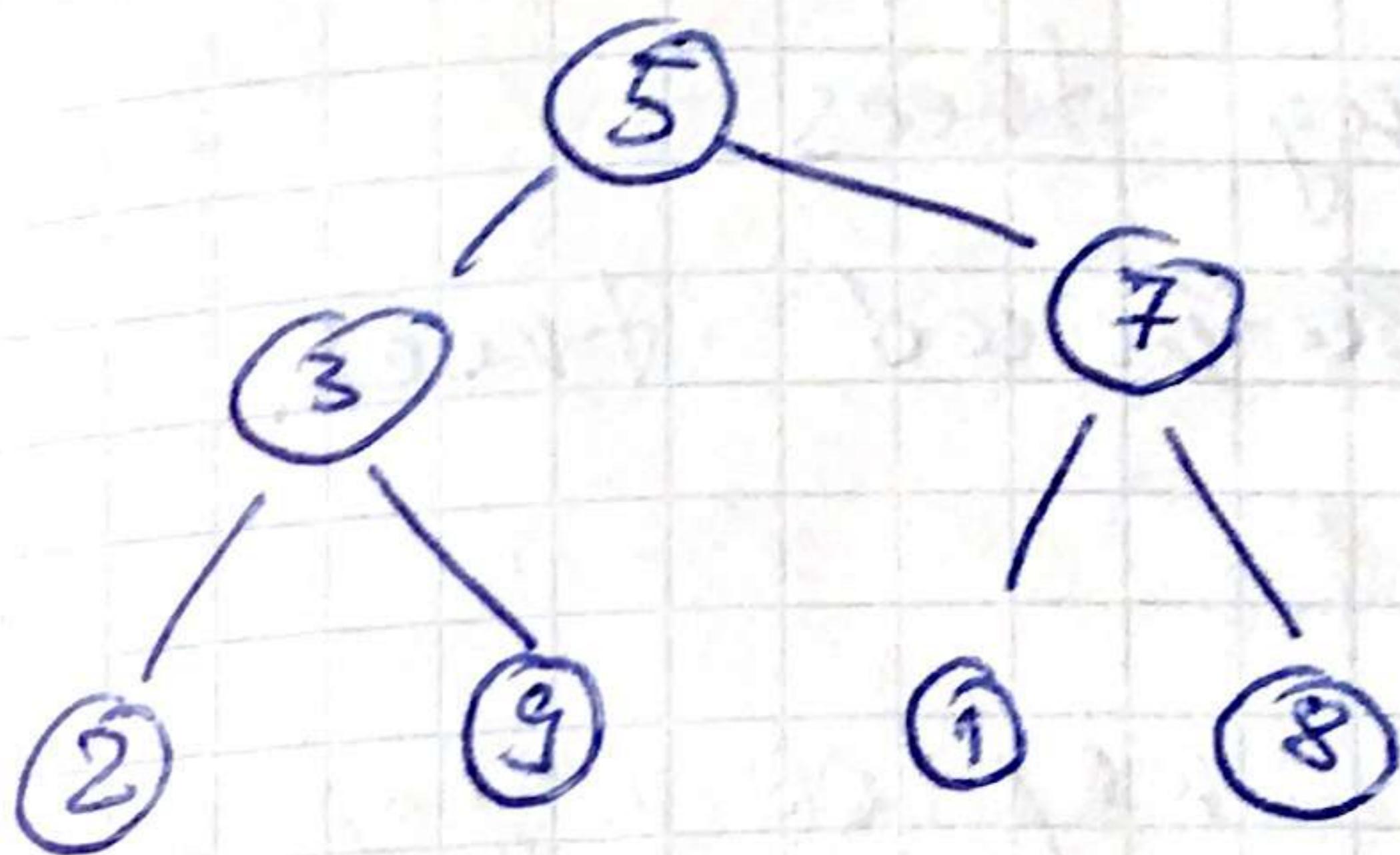


	xем-Реди
Insert( $x$ )	$O(d+1)$
Erase ( $x$ )	$O(1+1)$
Find ( $x$ )	$O(d+1)$ $d = \frac{h}{m} - \text{коэф. делн}$ $\leq \frac{h}{m} - \text{коэф. копийн}$

```

Find(x):
    if (root == null) return False
    if (root > x):
        Find(root->left, x)
    else:
        return True
    elif (root < x):
        Find(root->right, x)

```



// tree reader

BST

Insert  $O(\log n)$

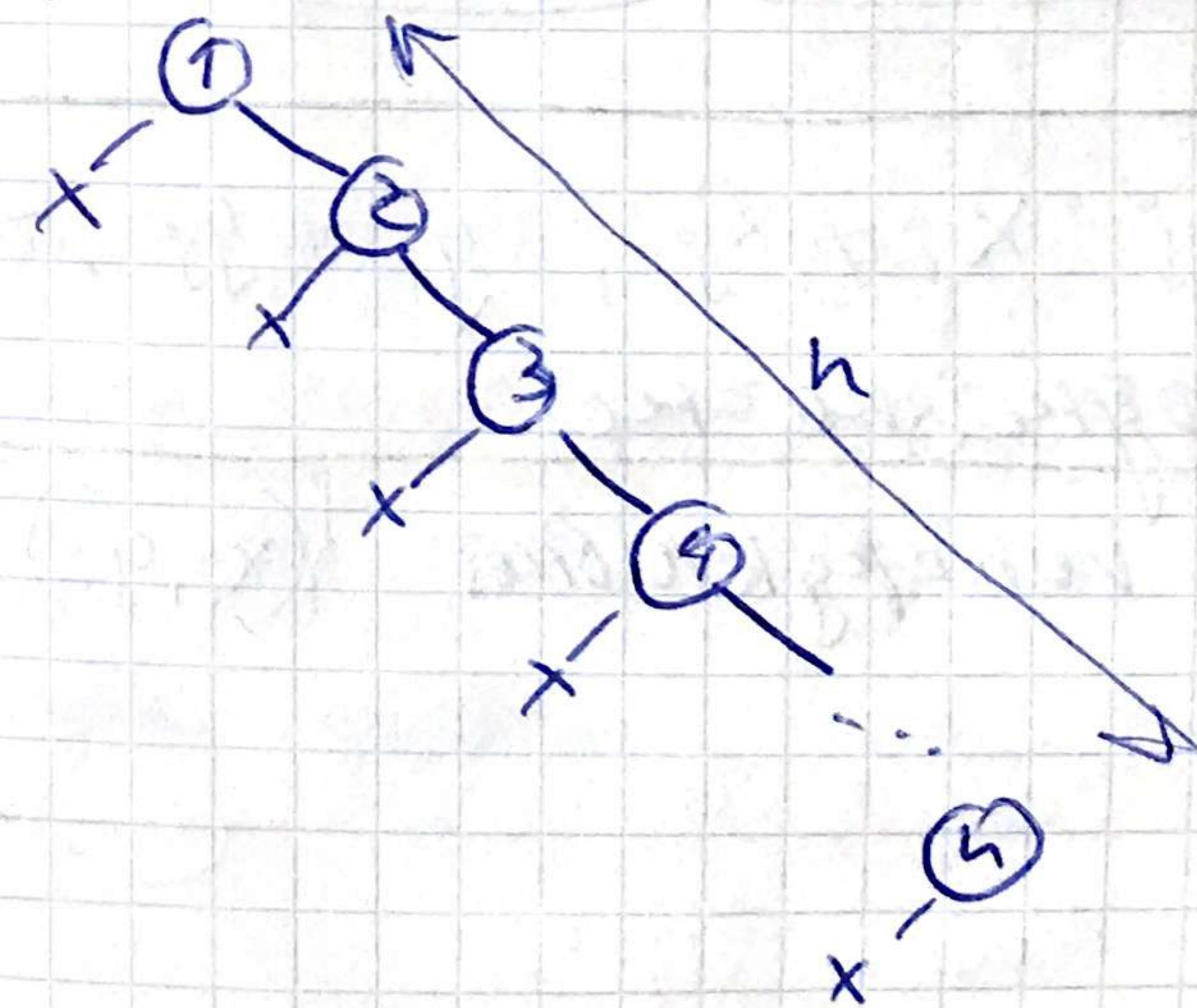
Erase  $O(\log n)$

Find  $O(\log n)$

• BST с динамической высотой, если это гайдеке

$O(\log n)$   
 $(\leq C \log n)$

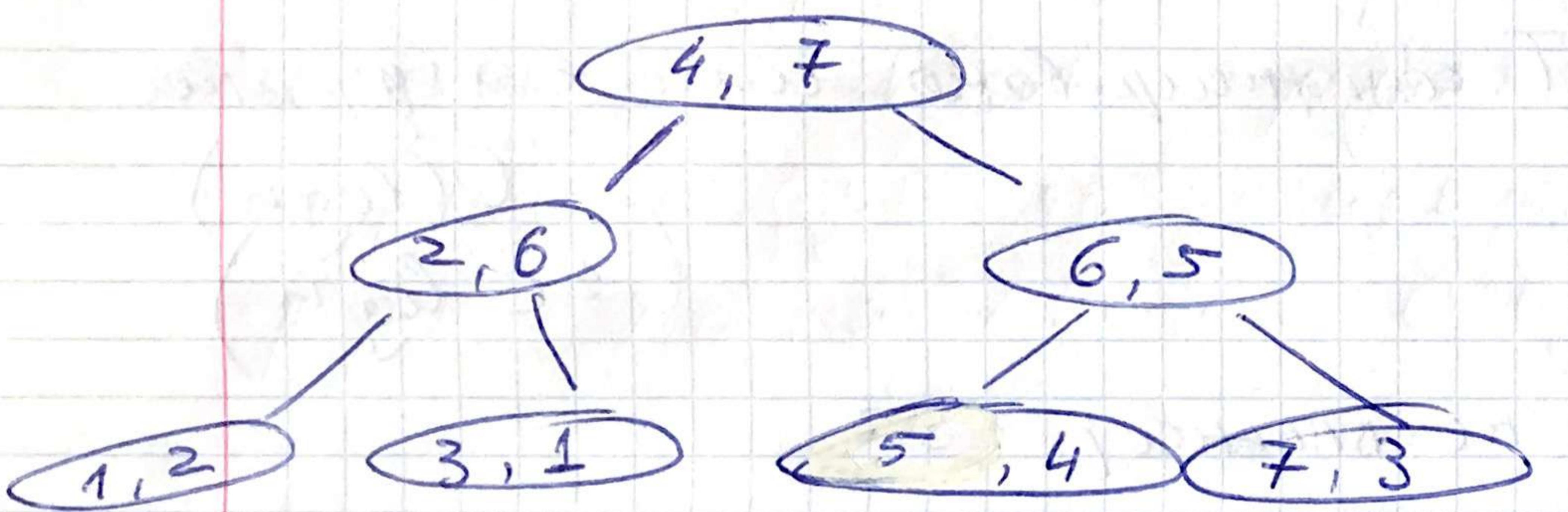
Пример: tree с динамич. высотой BST



Red-black, AVL, Splay trees -  
- balanced trees.

Cartesian Tree (Декартово дерево)

Он  $\Delta\Delta$  - структура данных, представляющая пару  $(x, y) \parallel x$ -координаты,  $y$ - приоритет и обладающая BST по  $x$  и реализацией декартовой крест (использование) по  $y$ .

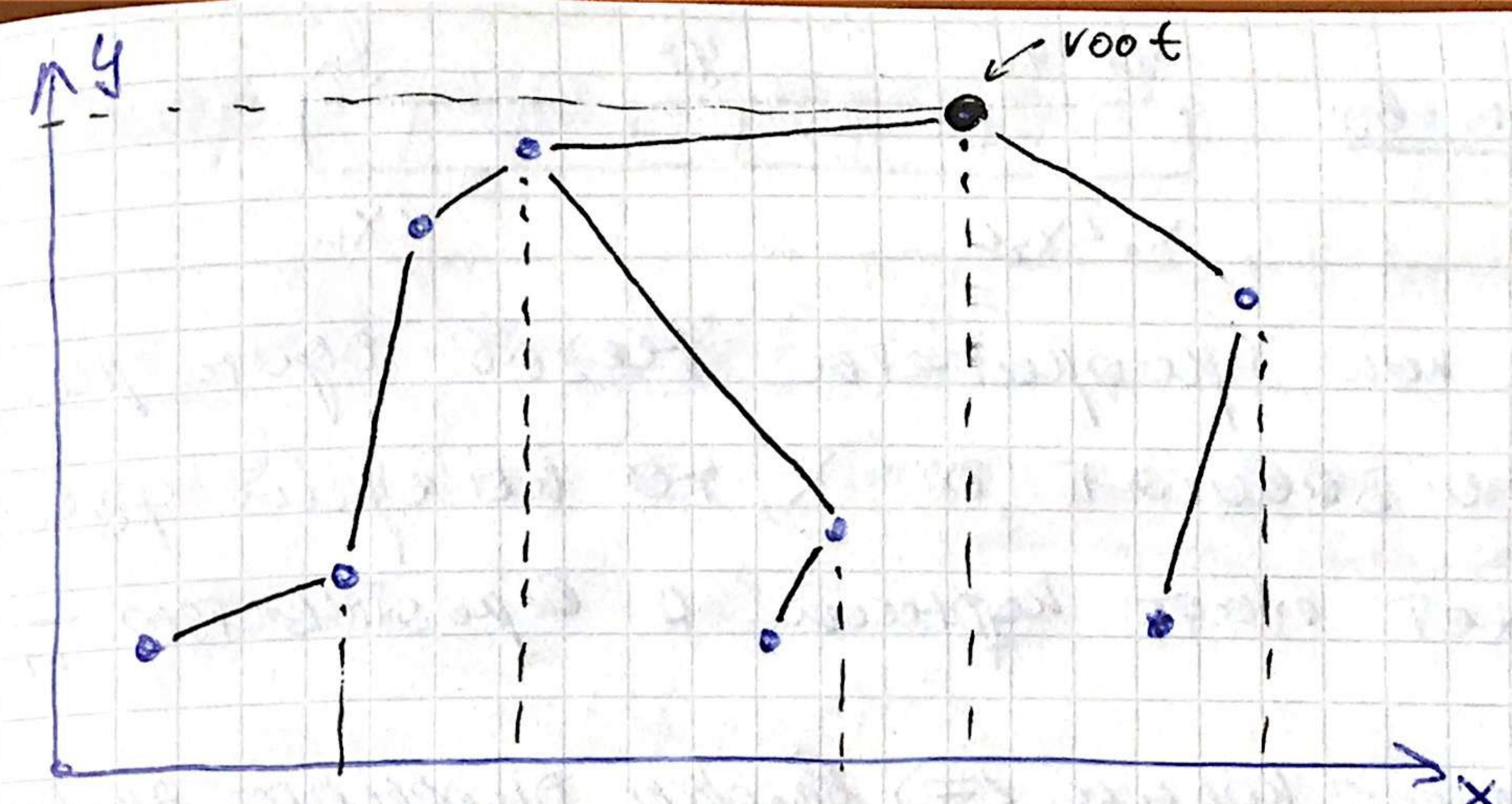


П Если  $\forall i, j \quad x_i \neq x_j, \quad y_i \neq y_j$ , то

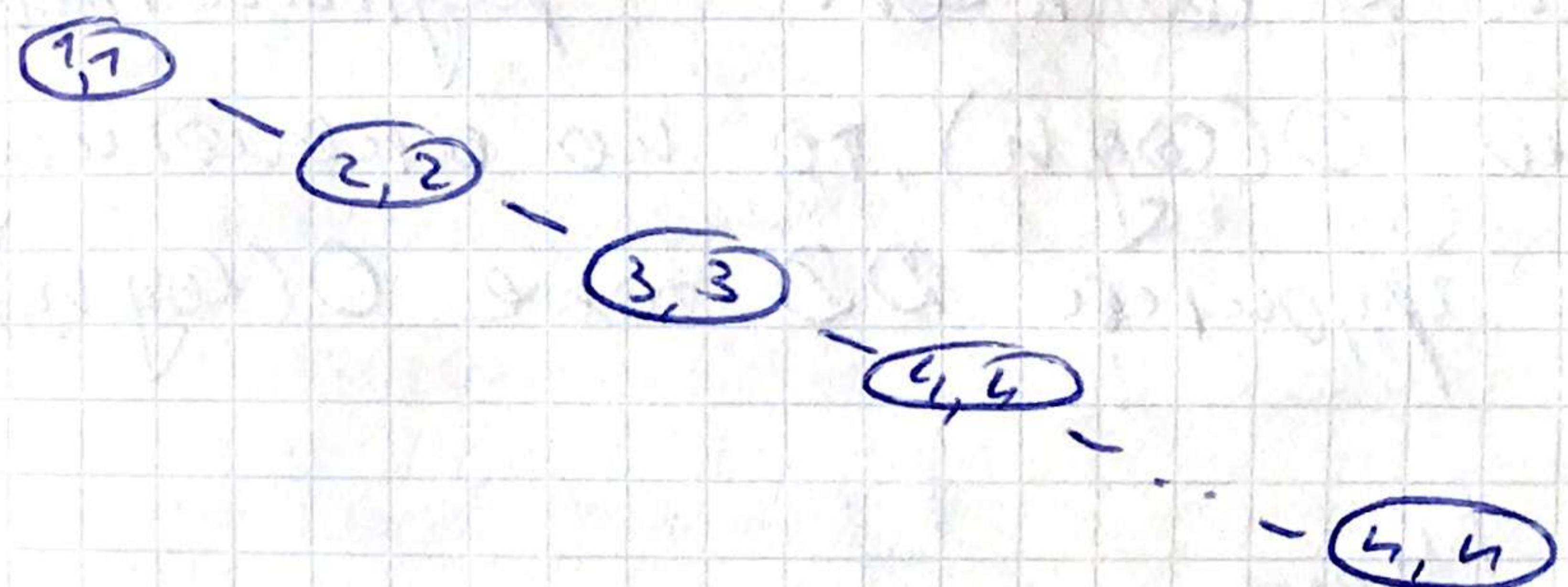
DD останется однозначным.

Д Докажем корректность.  $\{(x_i, y_i)\}_{i=1}^{n-1}$

Доказана  
Для  
Картинка

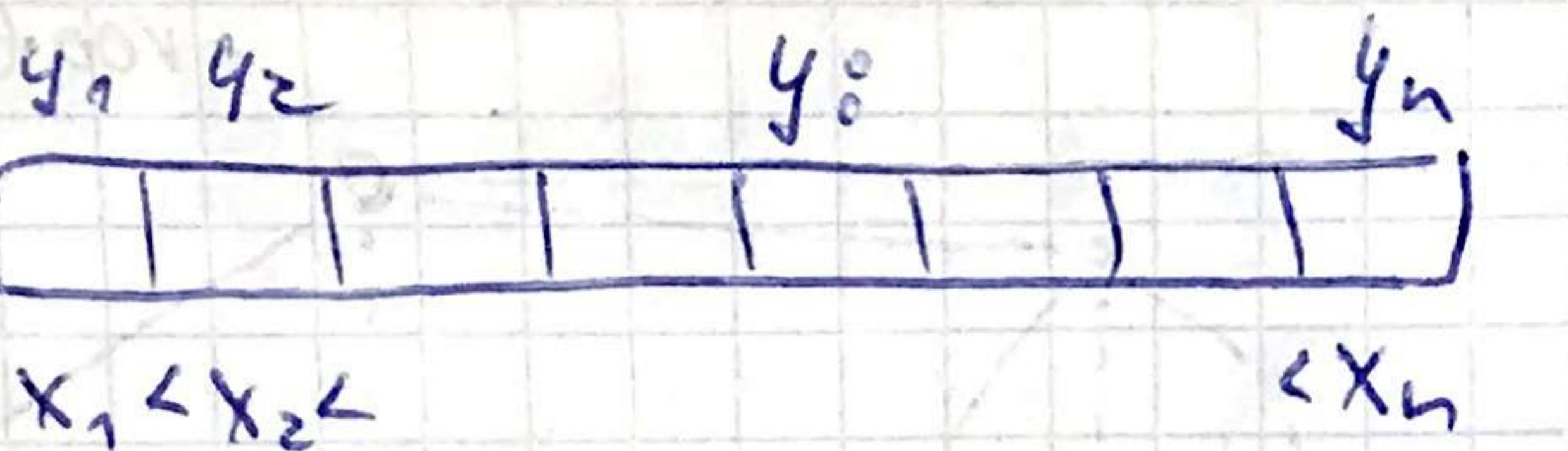


$$\{(1,1), (2,2), (3,3), \dots, (n,n)\}$$



Теорема: Рассмотрим  $y_i: y_{i=0}^{n-1}$  - независимые определения координат. Свойство:  $\forall i \neq j \quad P(y_i \neq y_j) = 1$  и  $\vec{y}$  не зависит от  $\vec{x}$ .  
Тогда средняя сложность DD -  $O(\log n)$

DOK - 60:



Так как производитель не зоб. огни. расход  
и не зависит от  $X$ , то налоги изъят  
налог спадет кратно с перестановки  $\frac{1}{n}$ .

Baedop kopra  $\Leftrightarrow$  Baedop ouaporo mra  
8 Quick Sort

Рекурсивно симметричне та в. ч. кратк. задачи.

# Split / Merge

Детали: детально бюджет председательство в  
Сенате Чубаров

struct Node {

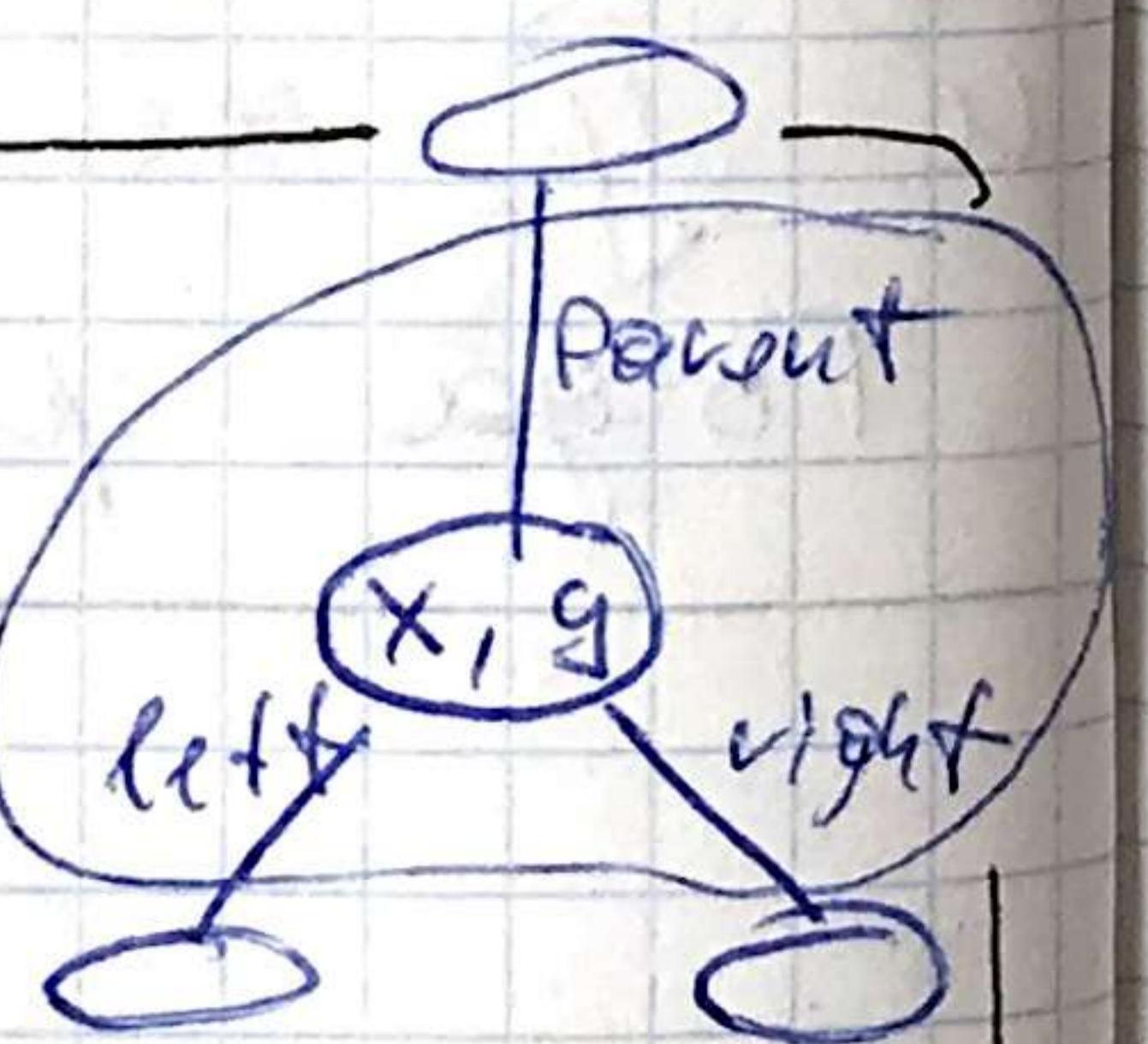
```
struct Node {
```

key T x; node \* left;

PrT(y); Node\* right;

PrT y; Node\* right;

Note to parents



Treeap = Node or root;

SetLeft(node, left) :

Мы можем добавить в стек а узла

if(note != acceptor):

node->left = left

```
if (left != null &gt;
```

update(node)

Split( $T, x_0$ ):

if ( $T == \text{nullptr}$ )

`push(T)` refers to (`newleftptr`, `newrightptr`)

if (root  $\rightarrow$   $x < x_0$ )

$T_1, T_2 = \text{Split}(\text{root} \rightarrow \text{right}, x_0)$

$$\text{self-right}(T, T_1) \leq \text{ulipper size}$$

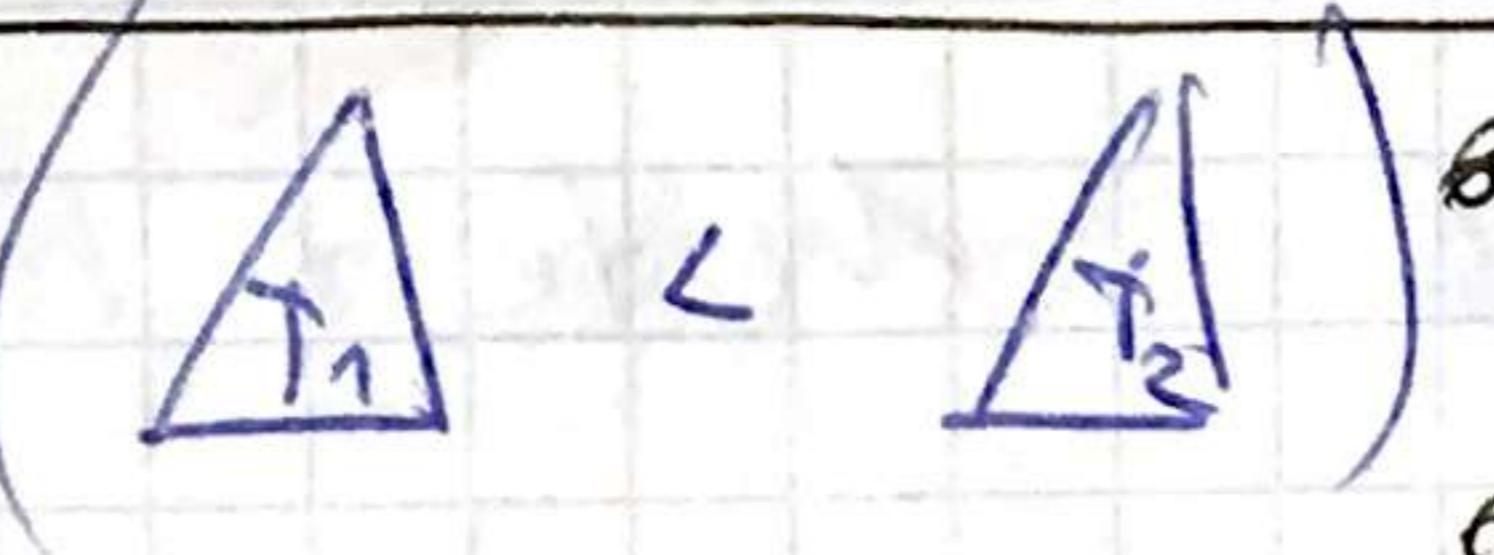
referen  $(\tau, \tau_\Sigma)$

else if \*root  $\rightarrow x \geq x_0$

$T_1, T_2 \in \text{Split}(T \rightarrow \text{left}, x_0)$

`SetLeft(T,T2)` ← member size

reference ( $T_0, T$ )

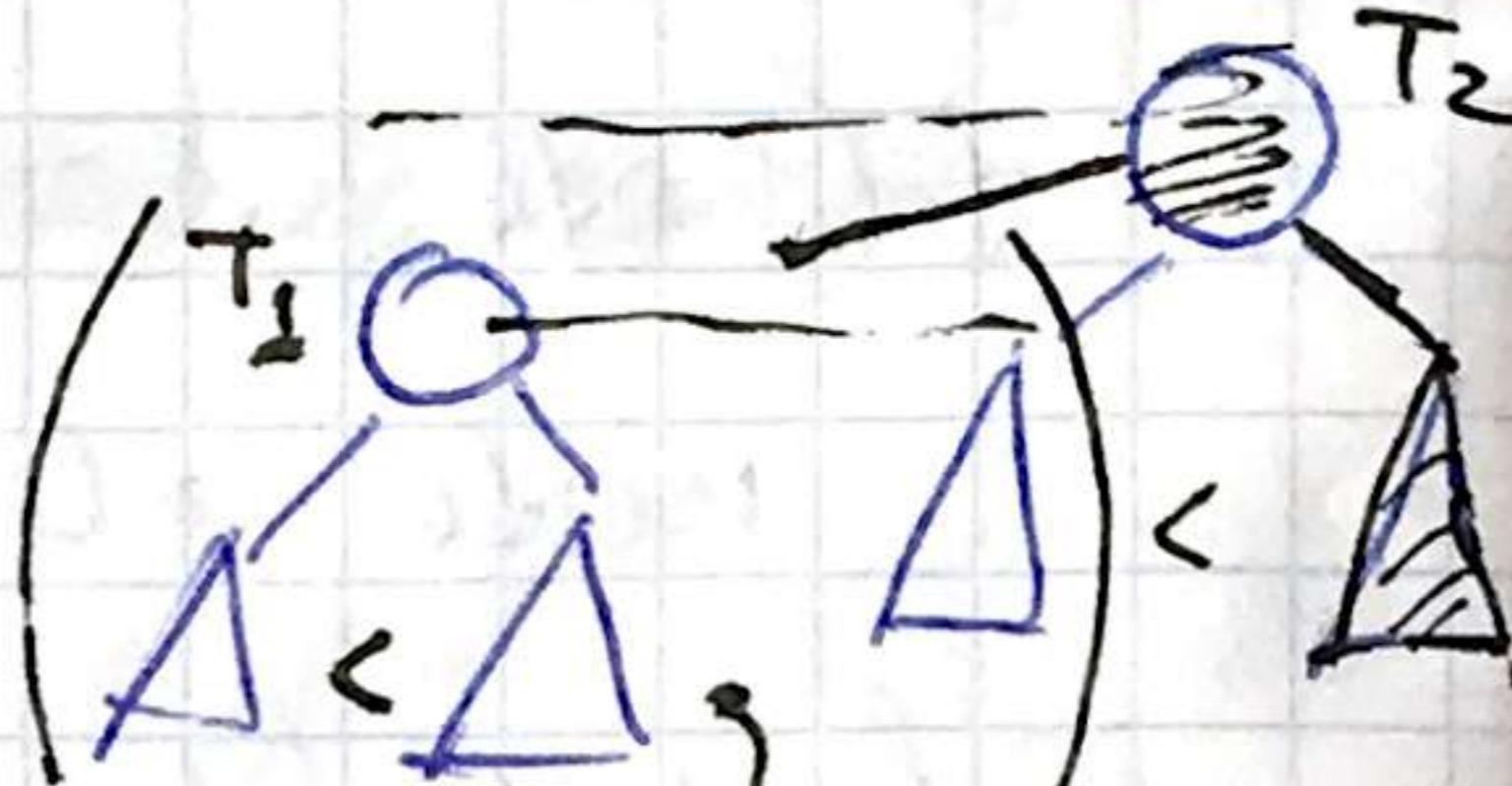
Merge( $T_1, T_2$ ):  $(T_1 < T_2)$  

if ( $T_1 == \text{nullptr}$ )  
Push( $T_1$ ) return  $T_2$   
Push( $T_2$ )

if ( $T_2 == \text{nullptr}$ )  
return  $T_1$

if ( $T_1 \rightarrow g < T_2 \rightarrow g$ )  
SetLeft( $T_2, \text{Merge}(T_1, T_2 \rightarrow \text{left}))$   
return  $T_2$

else //  $T_1 \rightarrow g \geq T_2 \rightarrow g$   
SetRight( $T_1, \text{Merge}(T_1 \rightarrow \text{right}, T_2))$   
return  $T_1$



Merge operation  
 $O(\log n)$

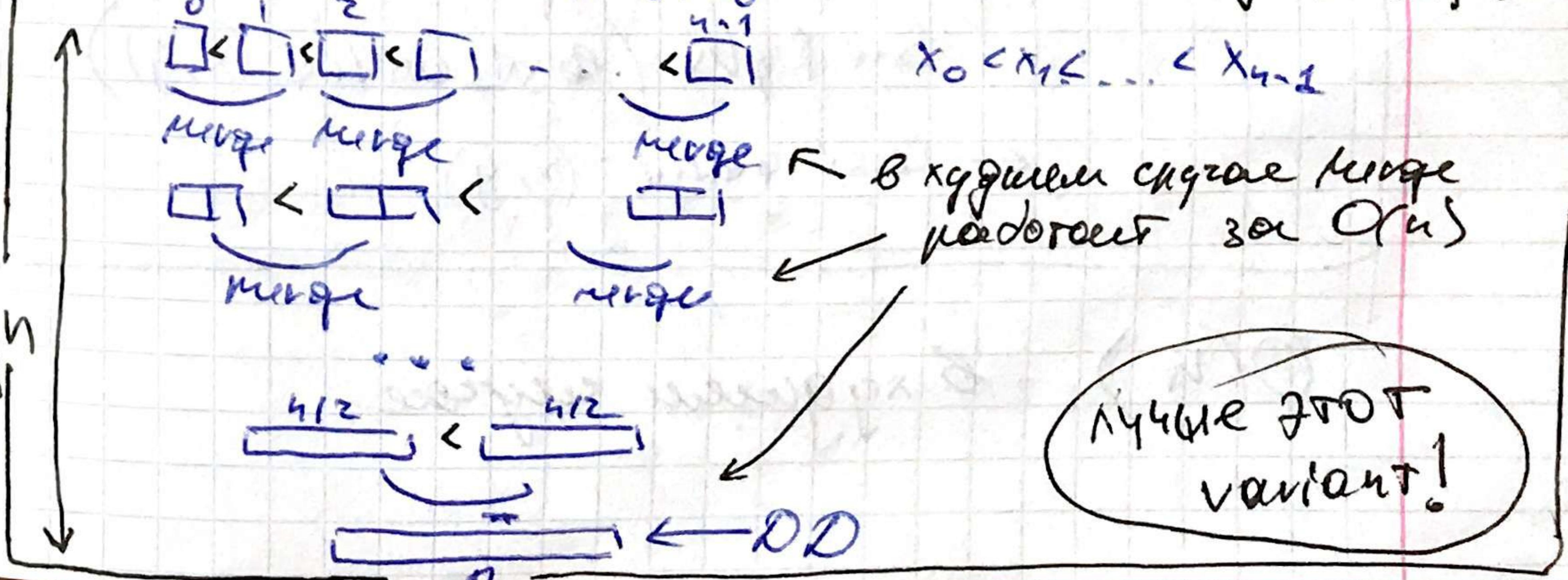
Merge operation  
 $O(n)$

Nootkae picee (Bailed)

Build ( $\{(x_i, y_i)\}_{i=0}^{n-1}$ ):  $\leftarrow O(n \log n)$  space  
root = nullptr  
for  $(x, y)$  in  $\{(x_i, y_i)\}$ :  
root = Insert (root,  $(x, y)$ ) /  $O(n^2)$

Доказательство  $\{f(x_i, y_j)\}_{i,j=0}^{\infty}$  симметрично по  $x$

1)  $\text{Build}(\{h(x_i, y_i)\}_{i=0}^{n-1})$ :  $\leftarrow O(n \log n)$  by quick  
sort

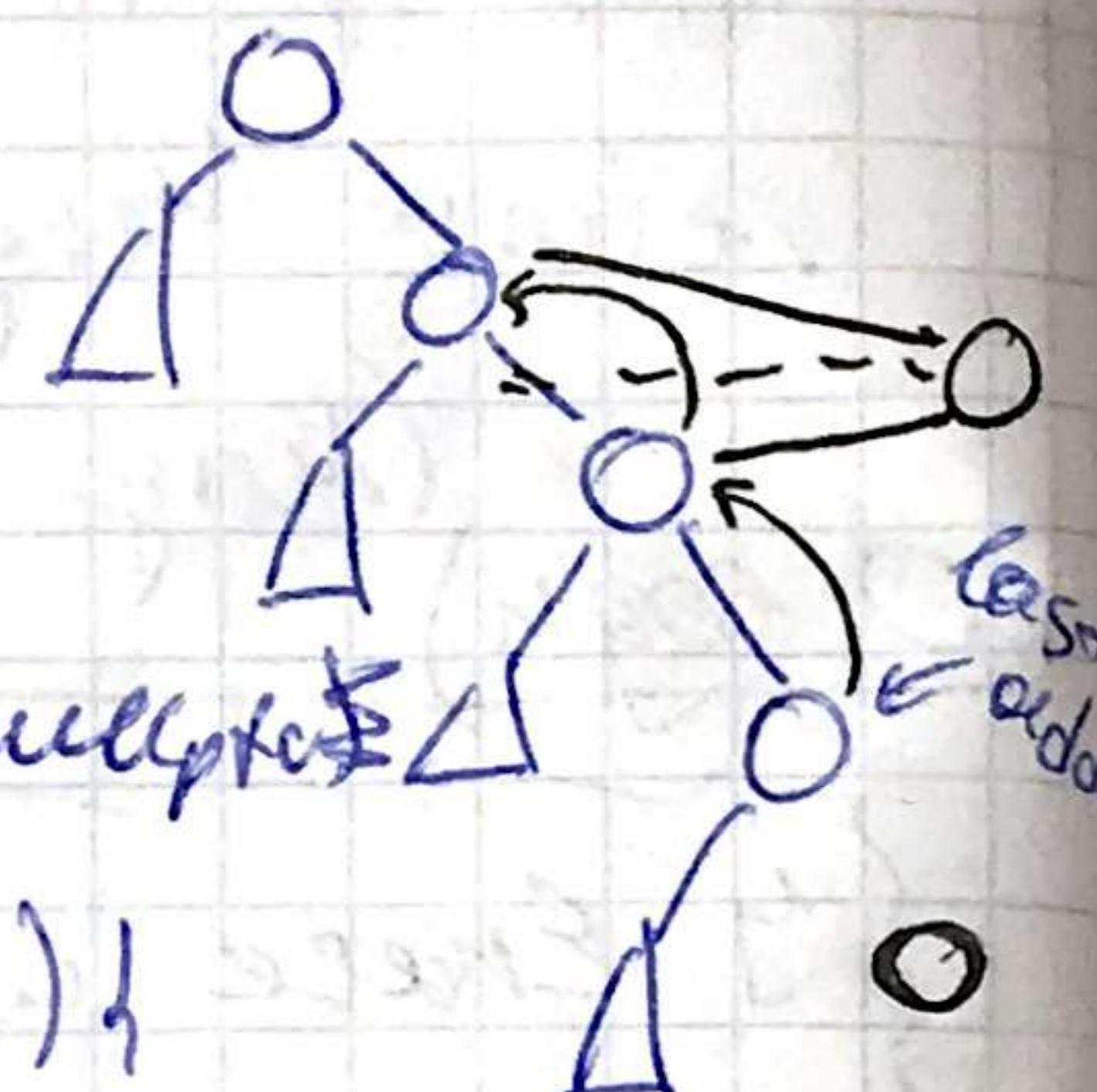


2) Build  $\left( \left\{ (x_i, y_i) \right\}_{i=0}^{n-1} \right)$   $\left\{ (x_i, y_i) \right\}$  oropr.  
root = nullptr

last-added = nullptr

for  $(x, y)$  in  $\{(x_i, y_i)\}$ :

$\left\{ \begin{array}{l} \text{if } x \text{ max eg} \\ \text{min,} \\ y \text{ not yet} \\ \text{appeared} \\ \text{done} \end{array} \right. \begin{array}{l} \text{while (last-added} \\ \text{!= nullptr} \\ \text{and} \\ \text{last-added} \rightarrow y < y) \end{array} \right.$



if ( $last\_added == nullptr$ )

SetLeft( $(x, y)$ , root)

root =  $(x, y)$

else:

SetLeft( $(x, y)$ ,  $last\_added \rightarrow \text{right}$ )

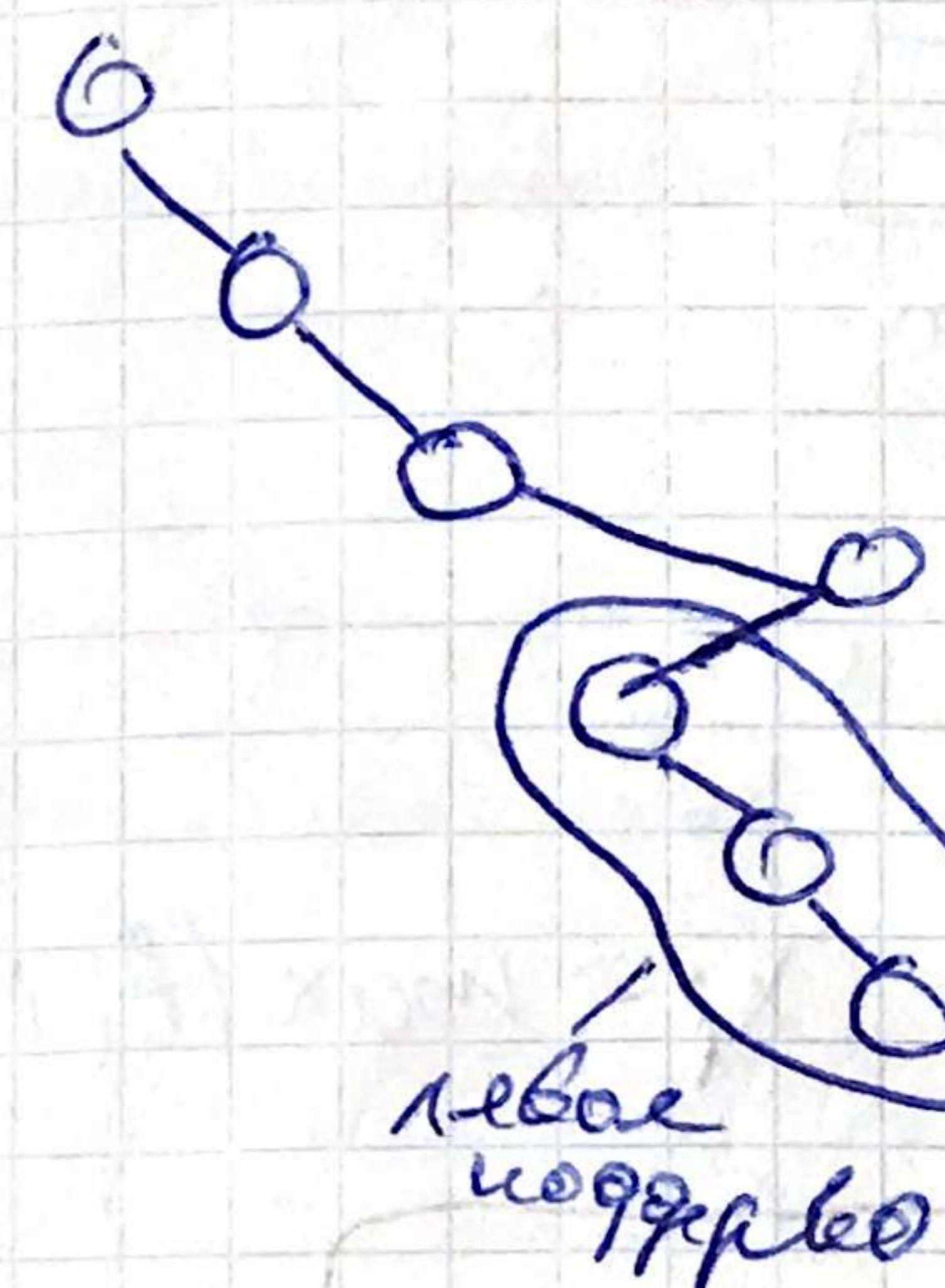
SetRight( $last\_added, (x, y)$ )

$\leftarrow last\_added = (x, y)$

$O(n)$  -  $\theta(n^2)$  queries worse

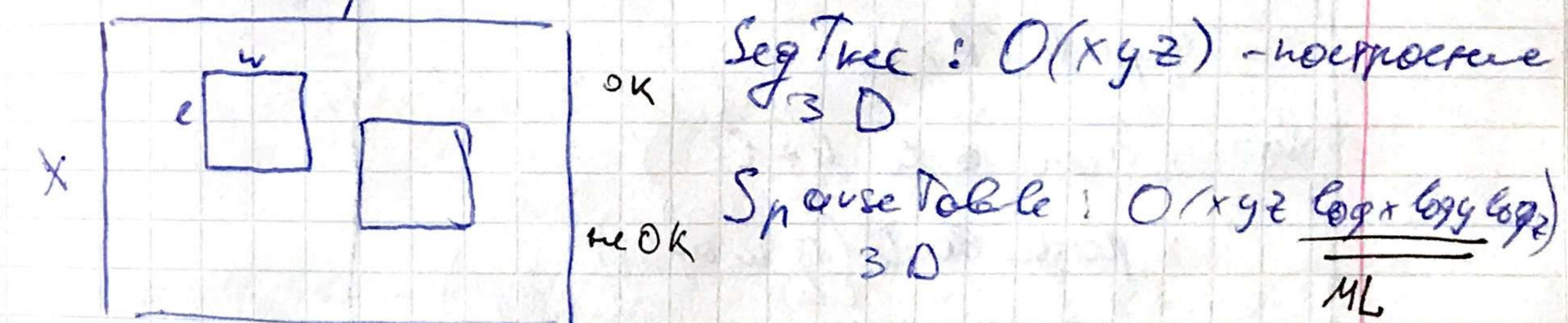
Kortgant Shereer godabane 1 pay u, yez  
horake "ogate pay".

$$\sum_{i=0}^{n-1} O(i) = O(n)$$

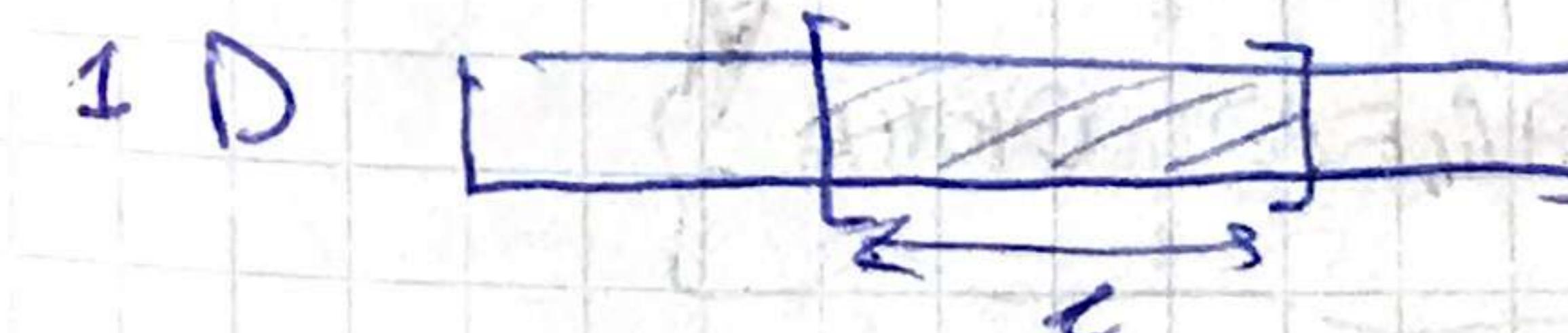


CEMIKAP

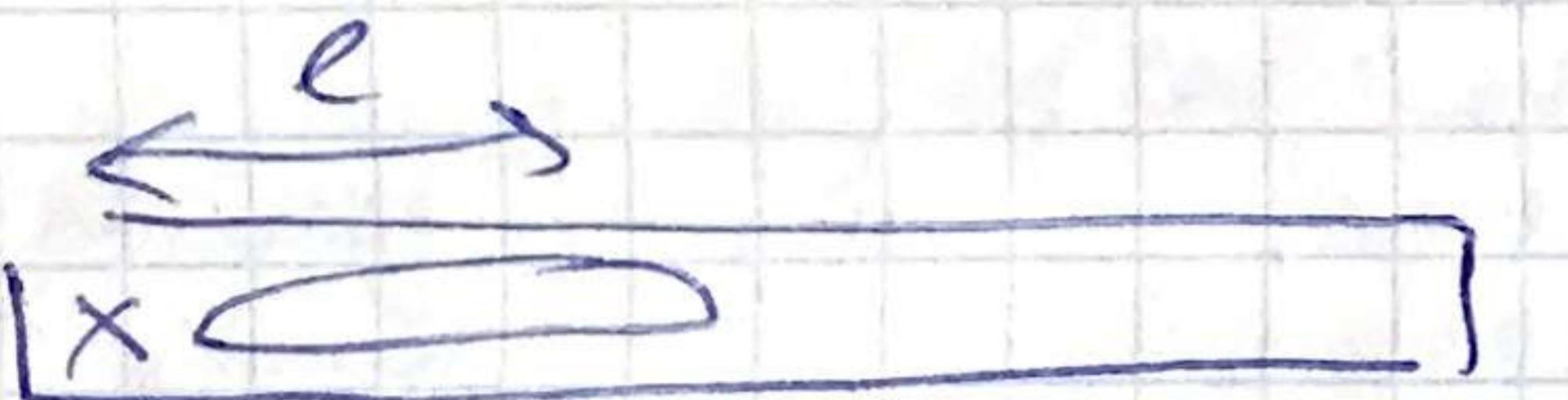
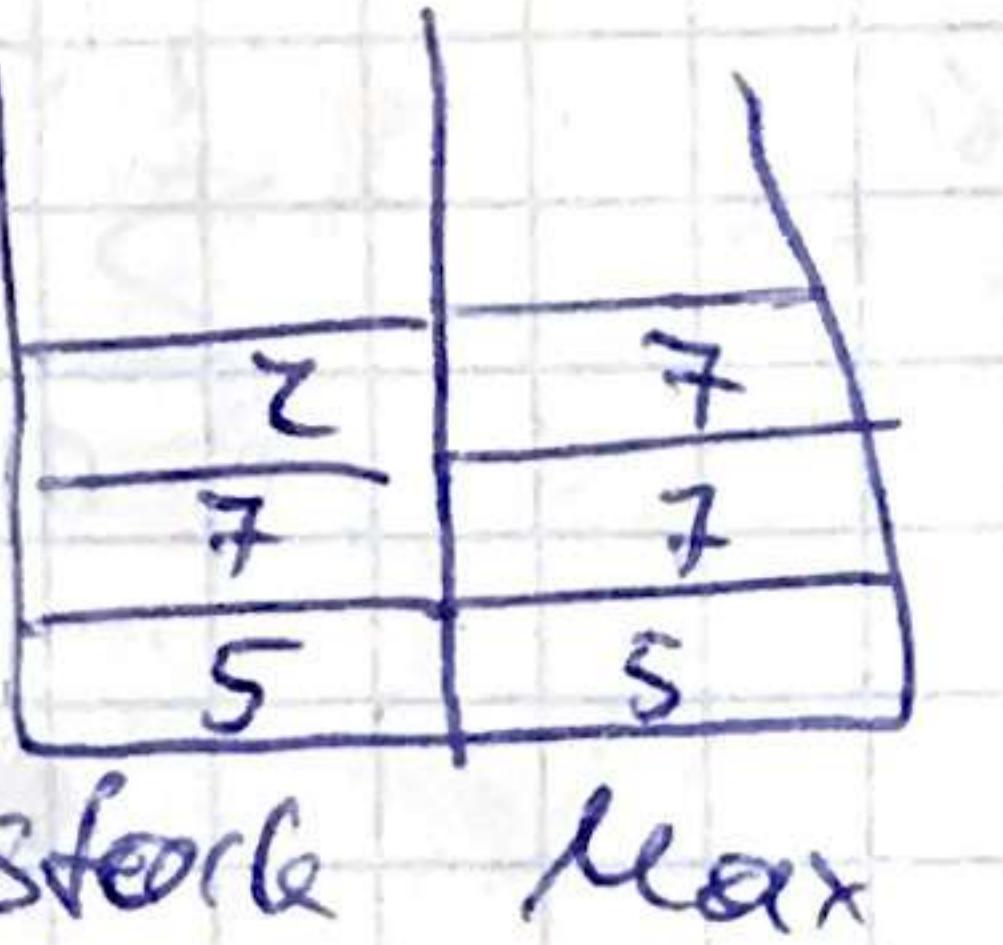
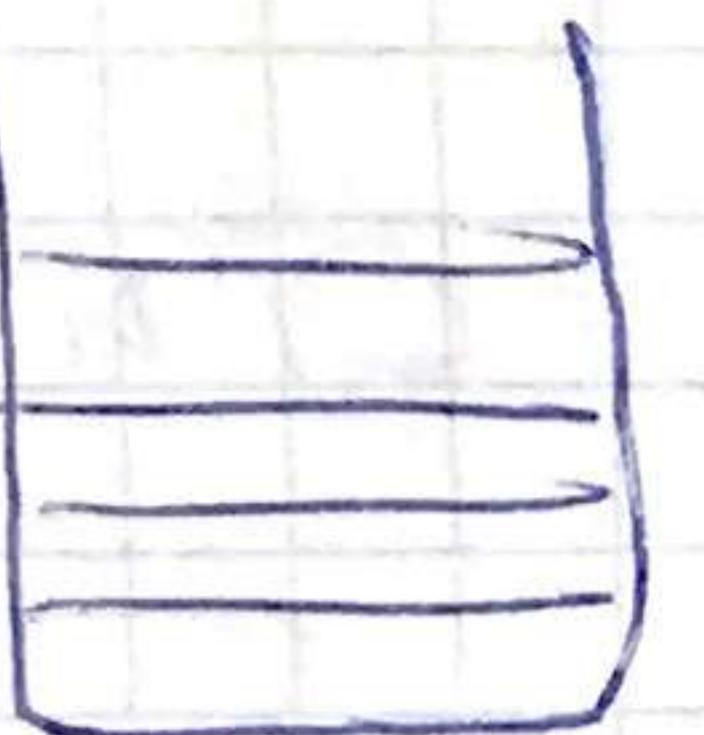
28.11.20



Xorpre:  $\left\{ \begin{array}{l} O(nyz) - \text{Build} \\ O(z) - \text{Query} \end{array} \right.$



Stack  $\rightarrow$  StackMax  $\rightarrow$  QueryMax



$$x_i^* = \max(A[i:i+l])$$

Query  $q_i$

for  $i$  from 0 to  $l$ :

$q_i$ .push( $A[i]$ )

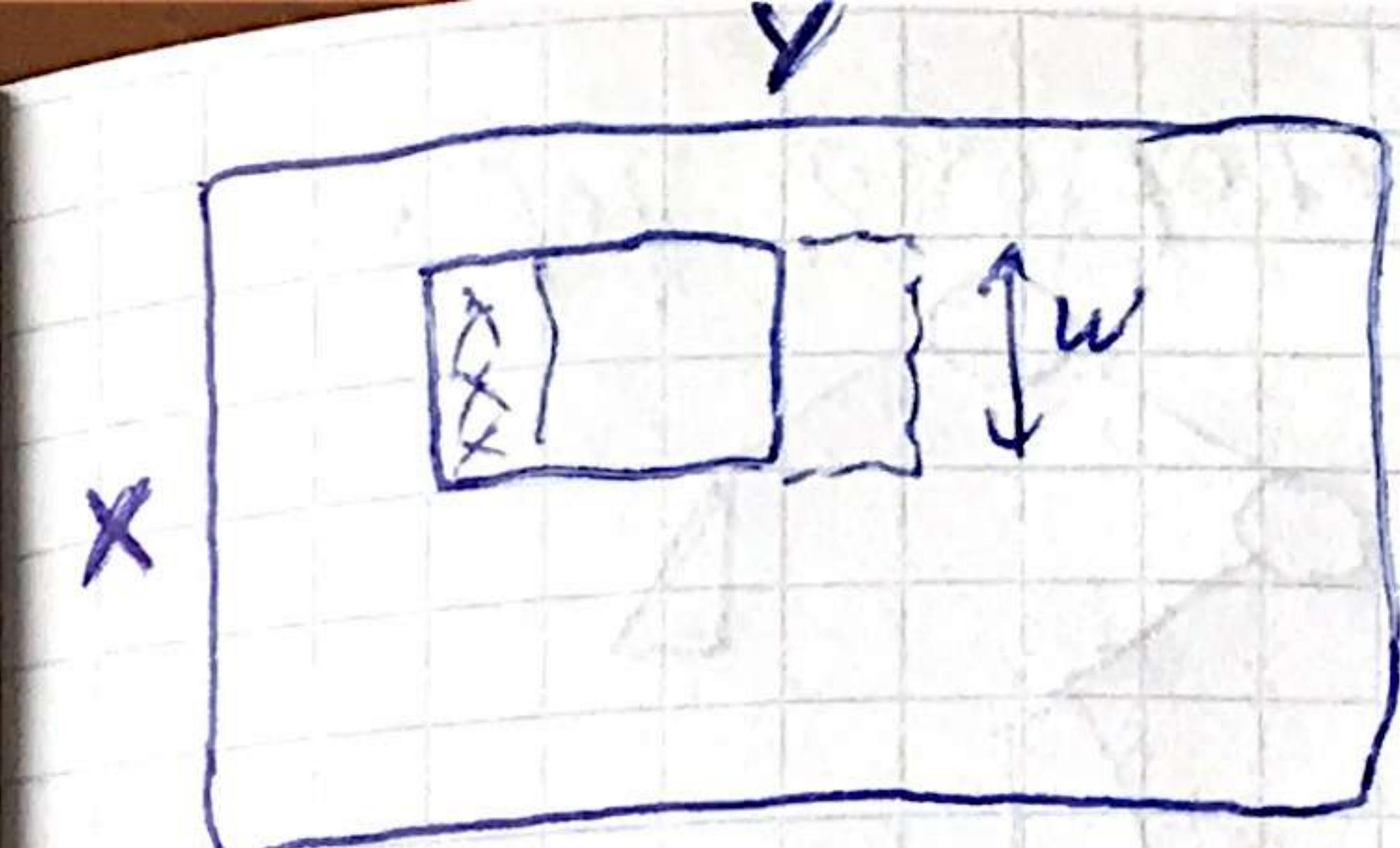
for  $i$  from 0 to  $n-l$ :

$x_i$ .push-back( $q_i$ .max())

$q_i$ .pop-front()

$q_i$ .push-back( $A[i+l]$ )

Algorithm сконструирован верно!

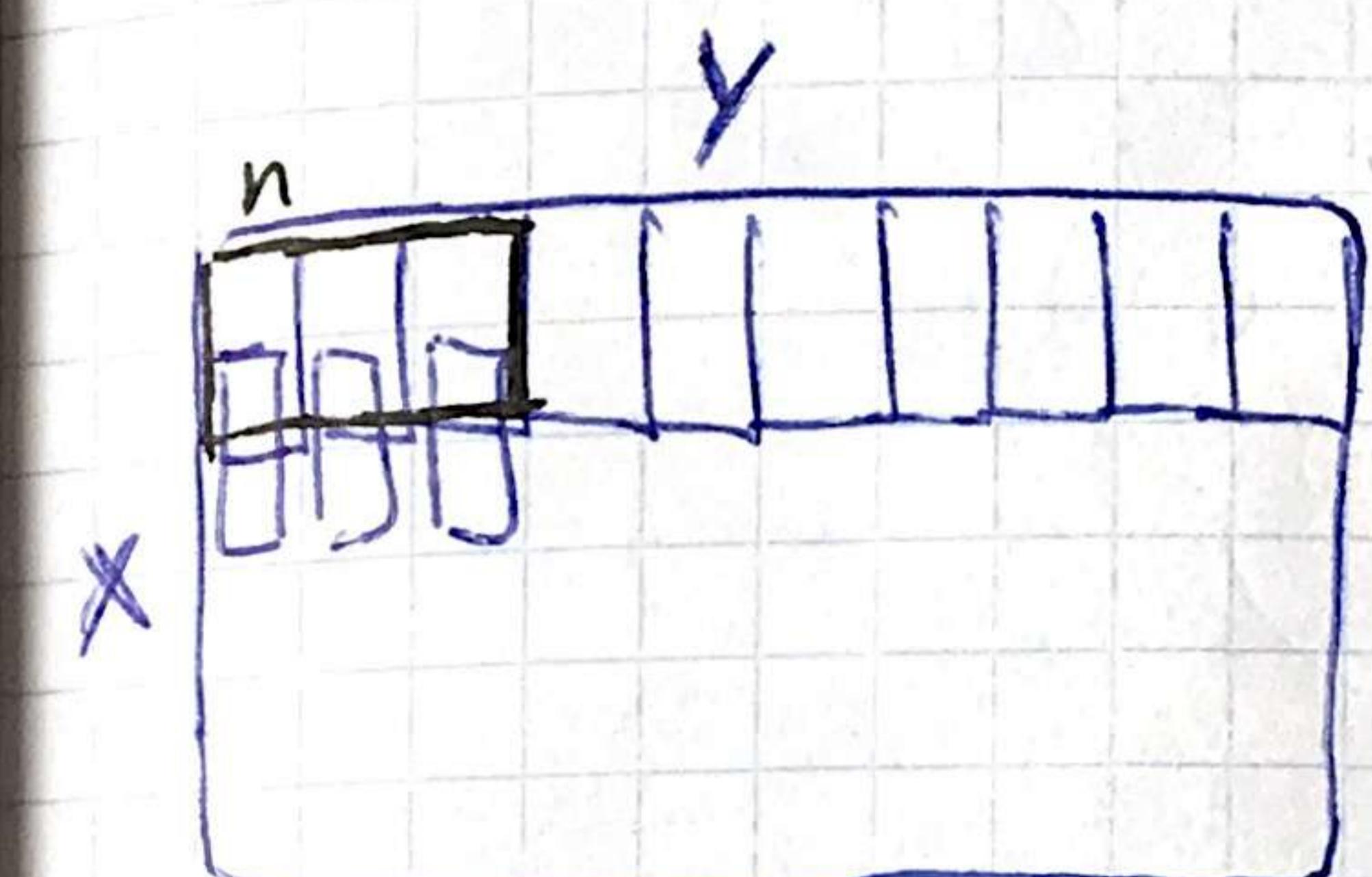


$-w$   $O(w)$   
 $+w$   $O(w)$

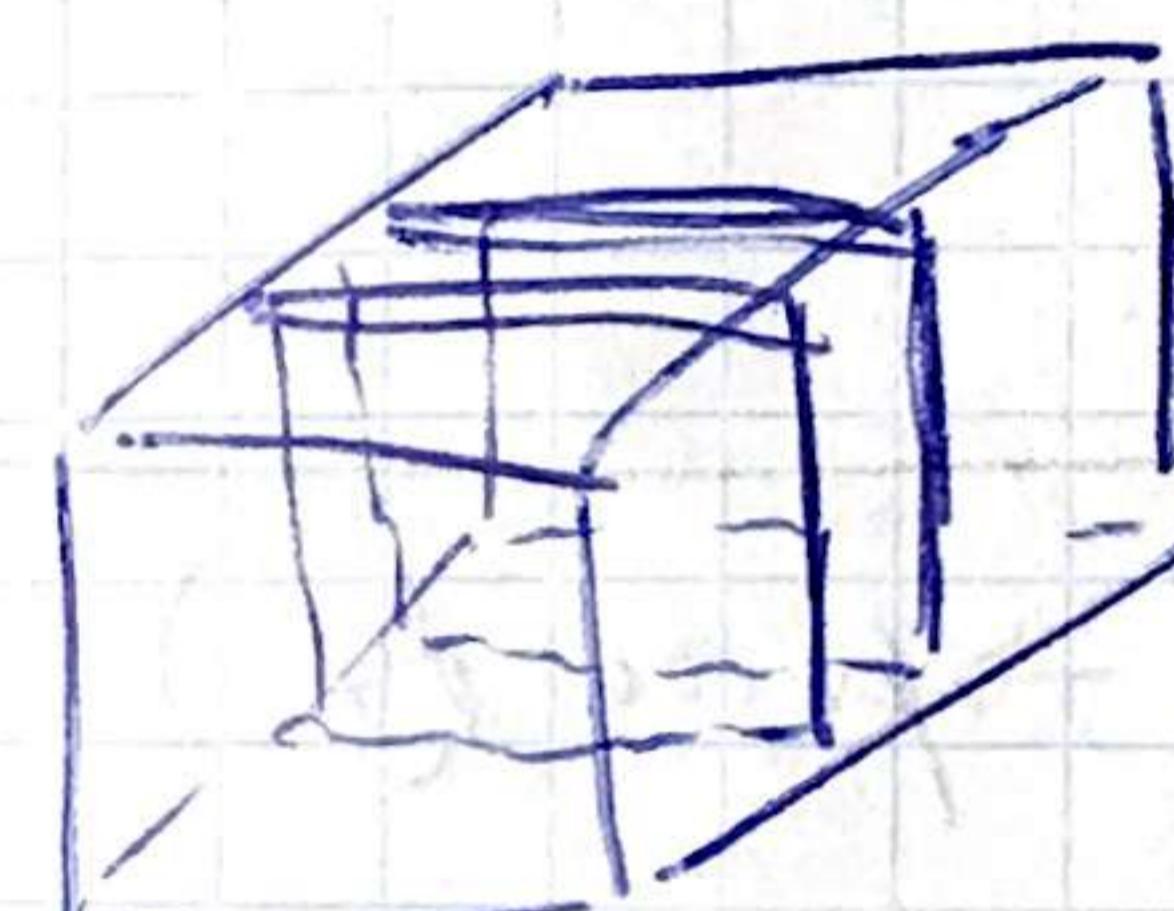
$O(xy) \cdot O(w)$

Сложно O(nm)

Однако в этом случае  
время выполнения

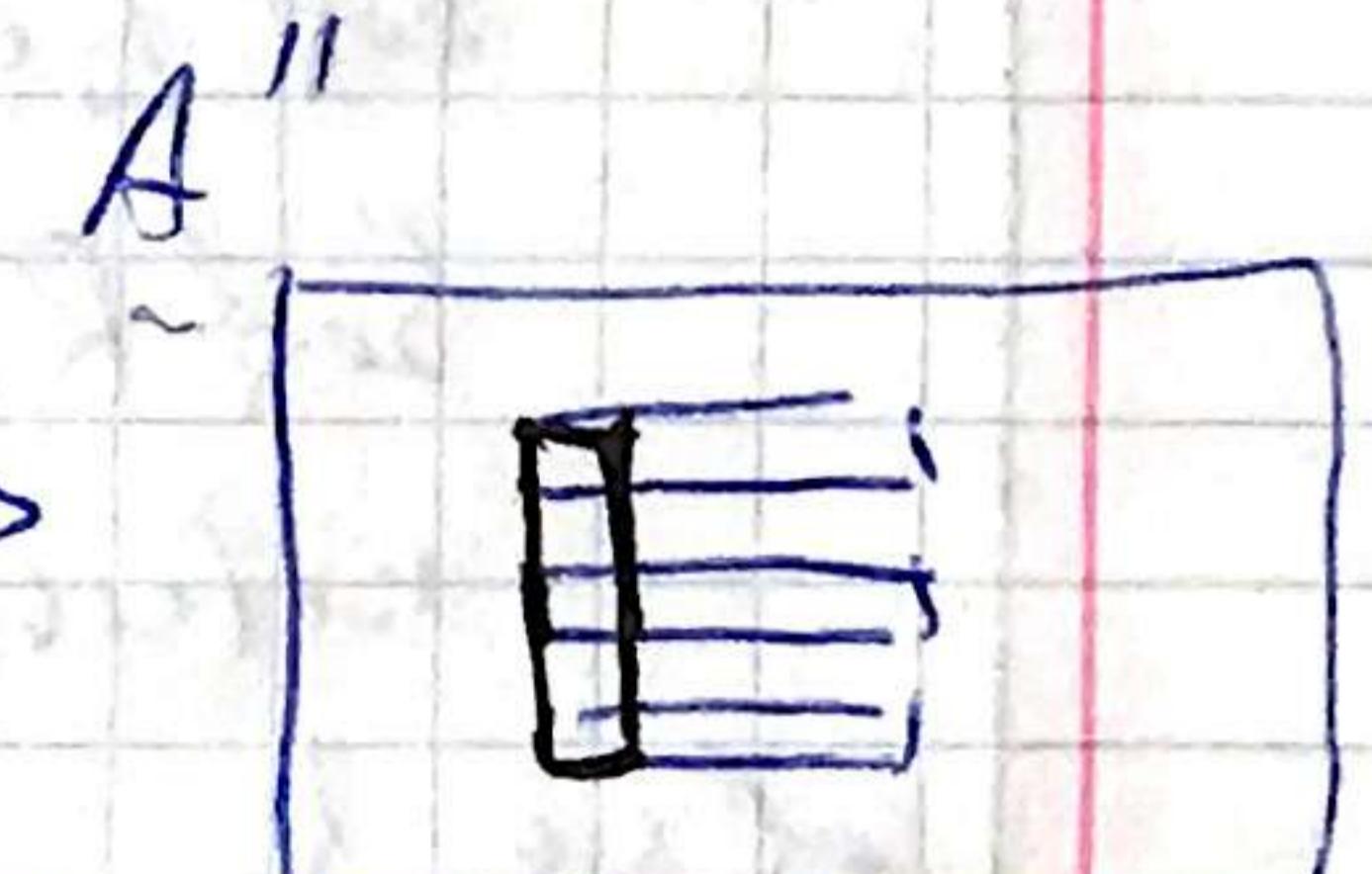
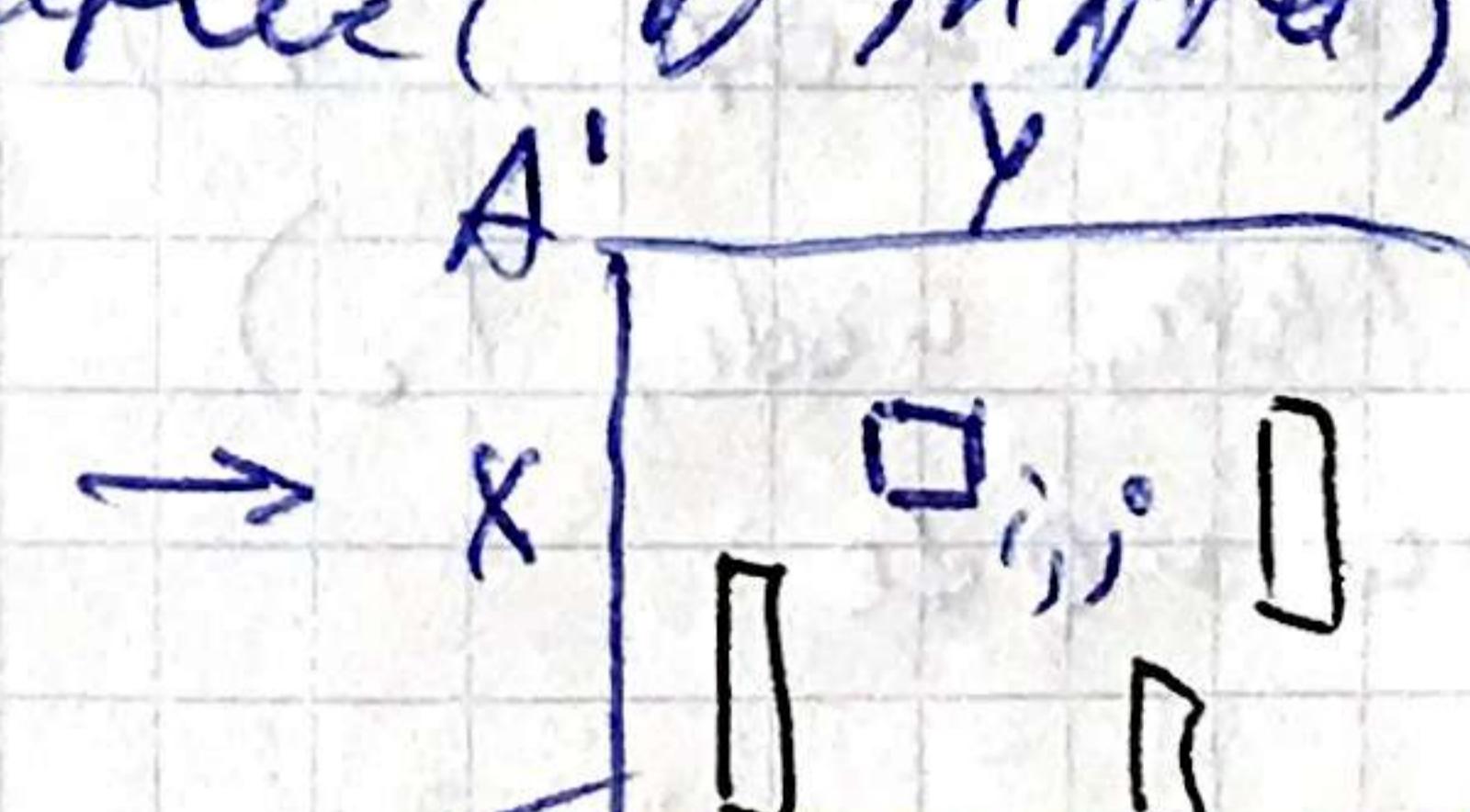
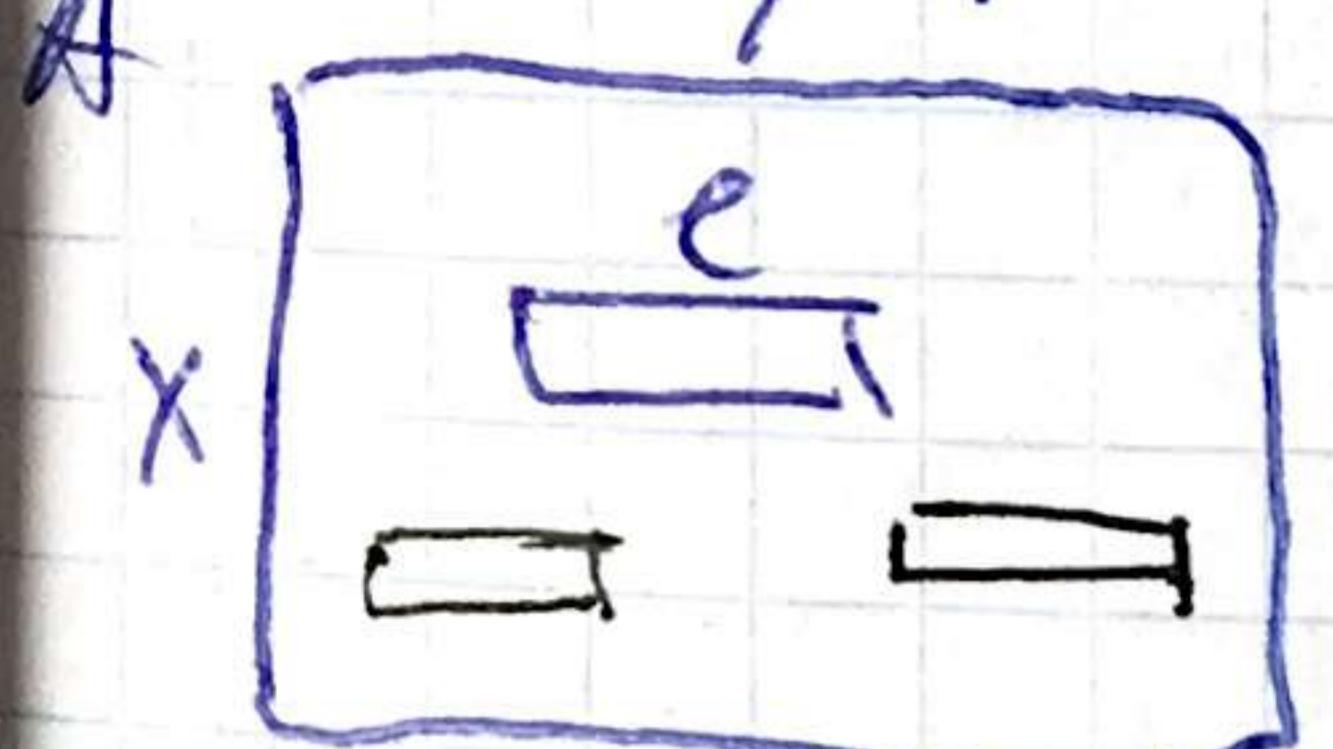


$O(xy)$



$O(xyz)$

Однородные параллелизмы (бинарно)



Скользящие окна по X  
Скользящие окна по Y,  
однозначно

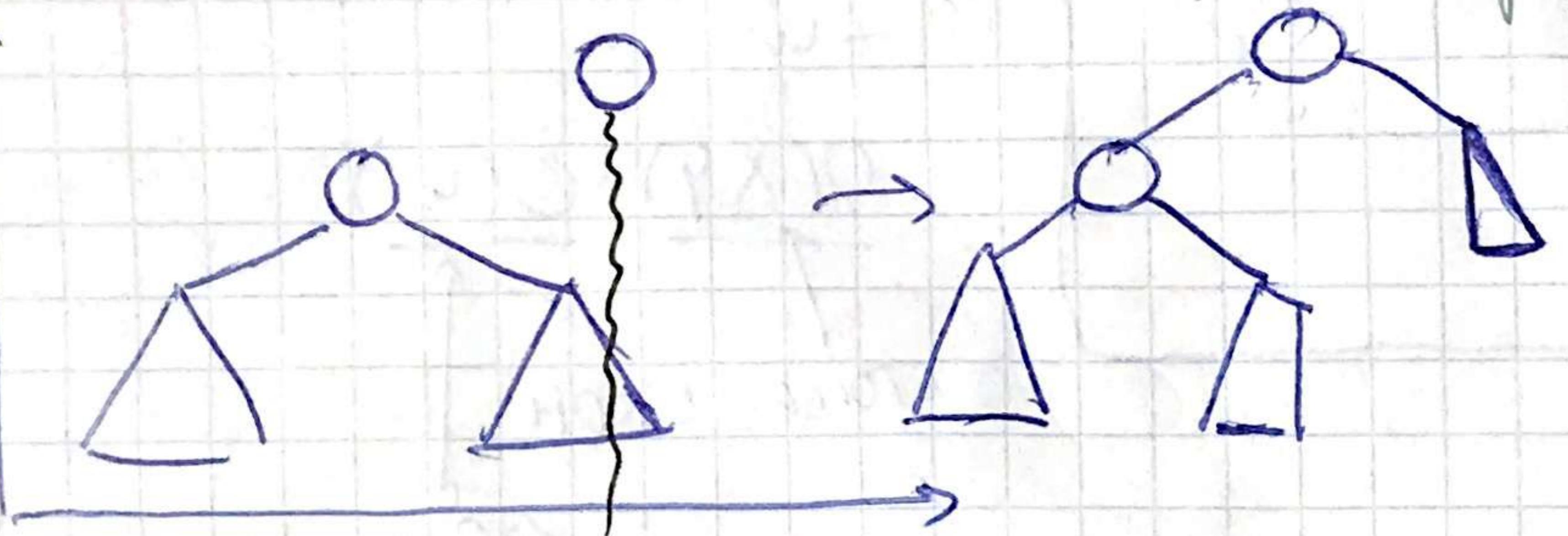
некоторые  $O(xyw)$   
Query -  $O(1)$

$$A'[i,j] = \max(A[i:i+c][j:j+w])$$

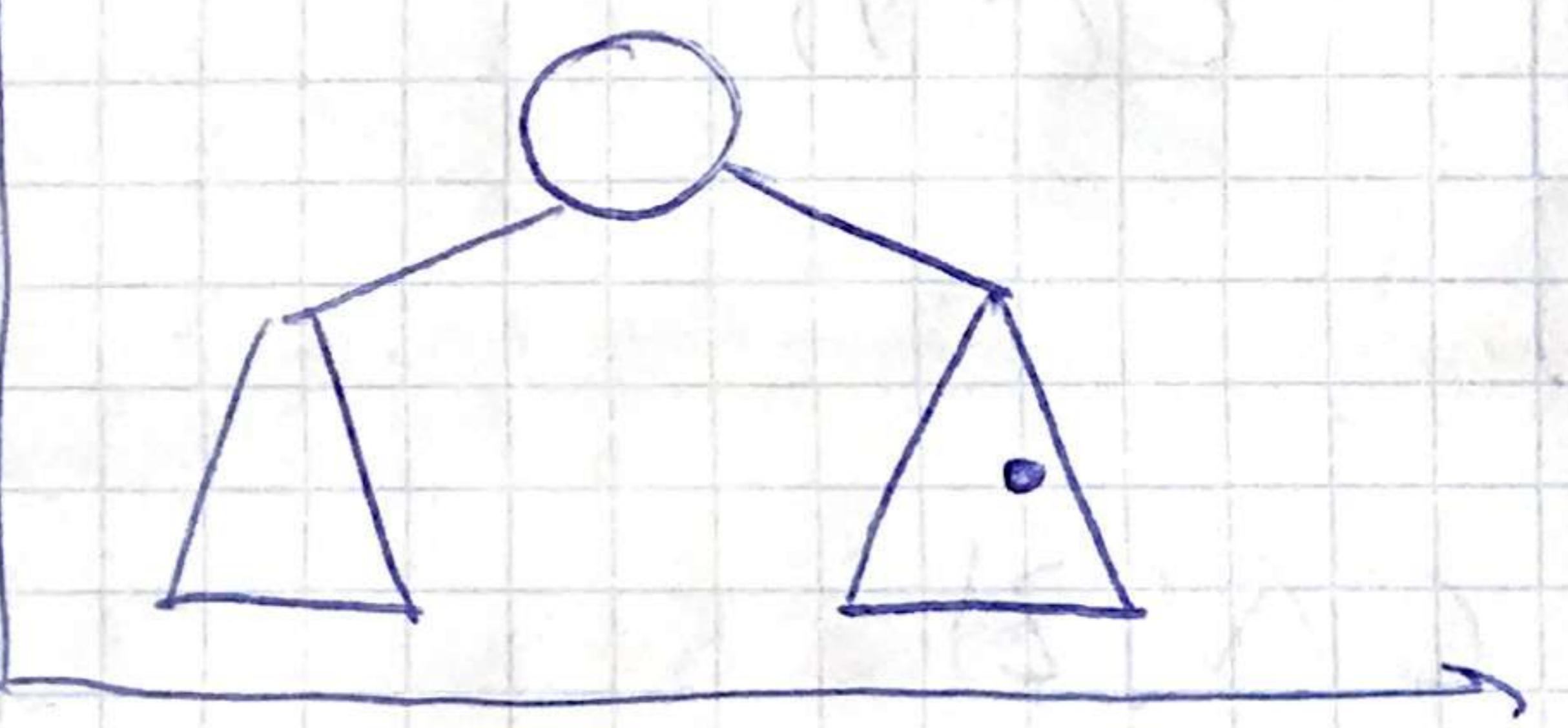
$$A''[i,j] = \max(A[i:i+c][j:j+w])$$

Insert

Cartesian Tree (Dekoprobo jepoko)



~~if newnode → priority~~



Insert

if (new\_node → priority  $\geq$  root → priority)

$T_1, T_2 = \text{Split}(\text{root}, x)$

$\text{SetLeft}(new\_node, T_1)$

$\text{SetRight}(new\_node, T_2)$

return new\_node;

else {

if (new\_node → key  $>$  root → key)

$\text{SetRight}(\text{root}, \text{Insert}(\text{root} \rightarrow \text{right}, new\_node))$

return root;

else ... //An error case

Erase

Erase(x)  $\leftarrow$  if (root = null)  $O(h) + O(T(\text{Merge}))$

if (root  $\rightarrow$  x  $=$  x)

return merge (T-left, T-right)

if (root  $\rightarrow$  x  $<$  x)

$\text{SetRight}(\text{root}, \text{Erase}(\text{root} \rightarrow \text{right}, x))$

return root

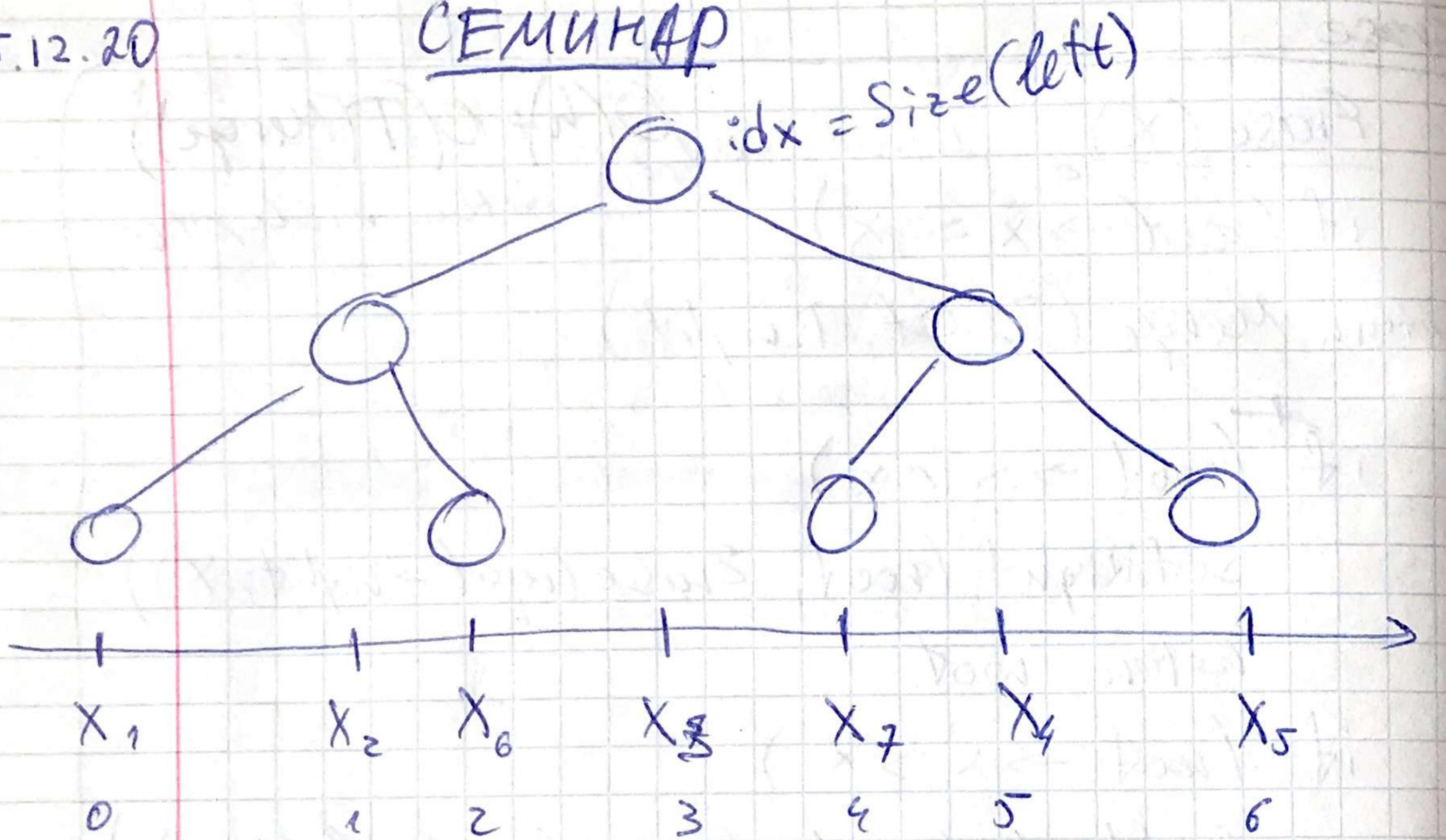
if (root  $\rightarrow$  x  $>$  x)

$\text{SetLeft}(\text{root}, \text{Erase}(\text{root} \rightarrow \text{left}, x))$

return root

5.12.20

## СЕМУНДАР

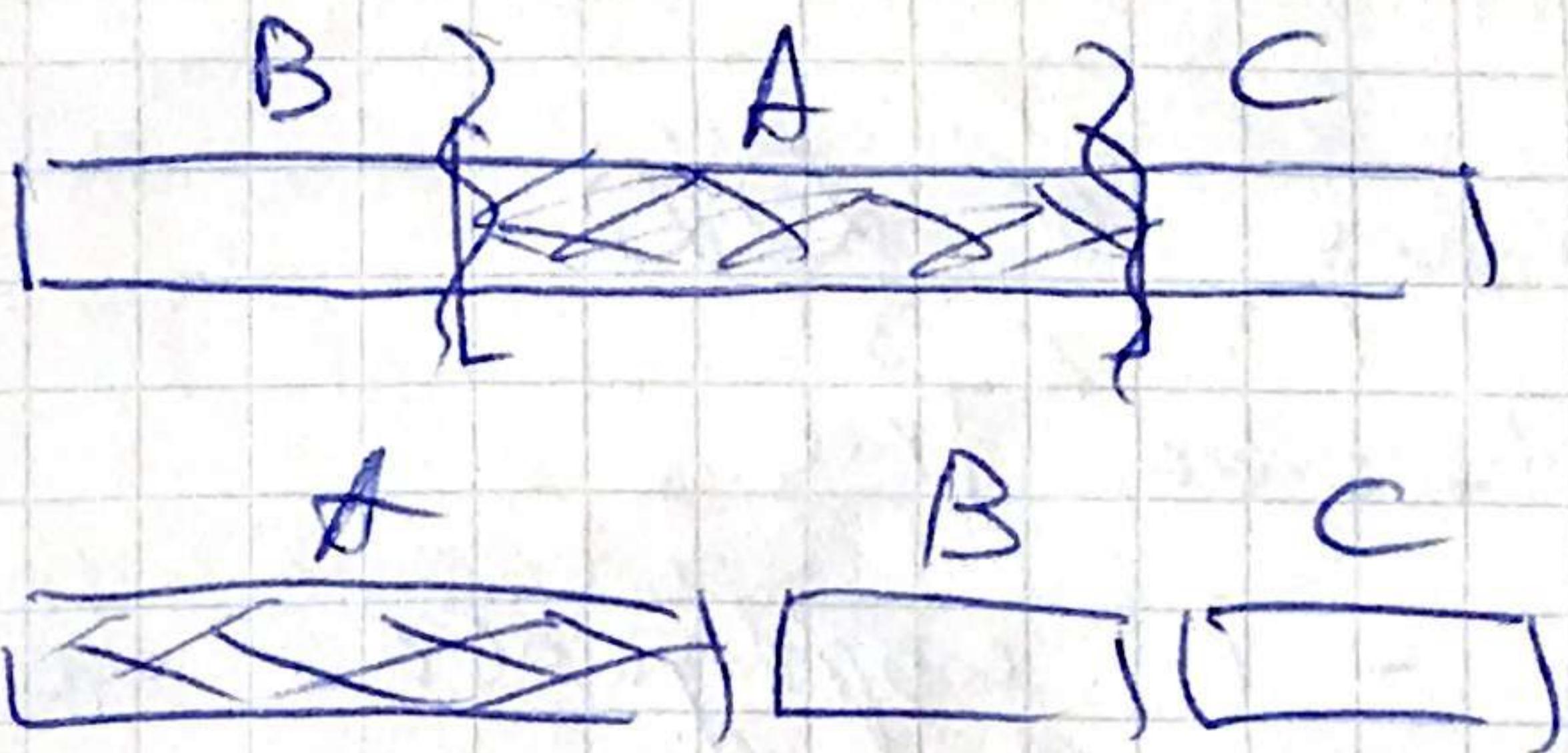


Find (idx, T)

```

if (idx < size(left))
    return T
else if (idx < size(left))
    return Find(left, idx)
else //idx > size(left)
    return Find(right, idx - 1 - size(left))
  
```

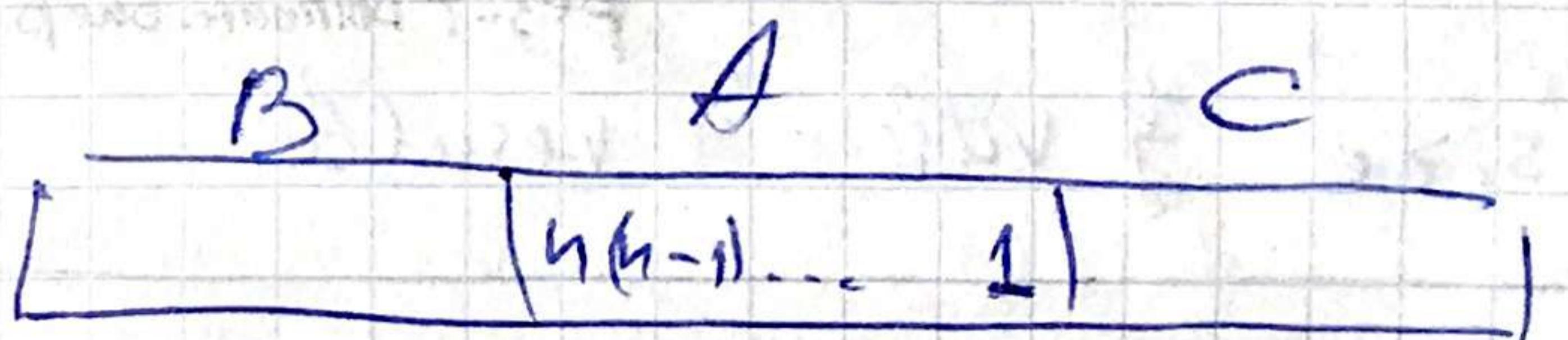
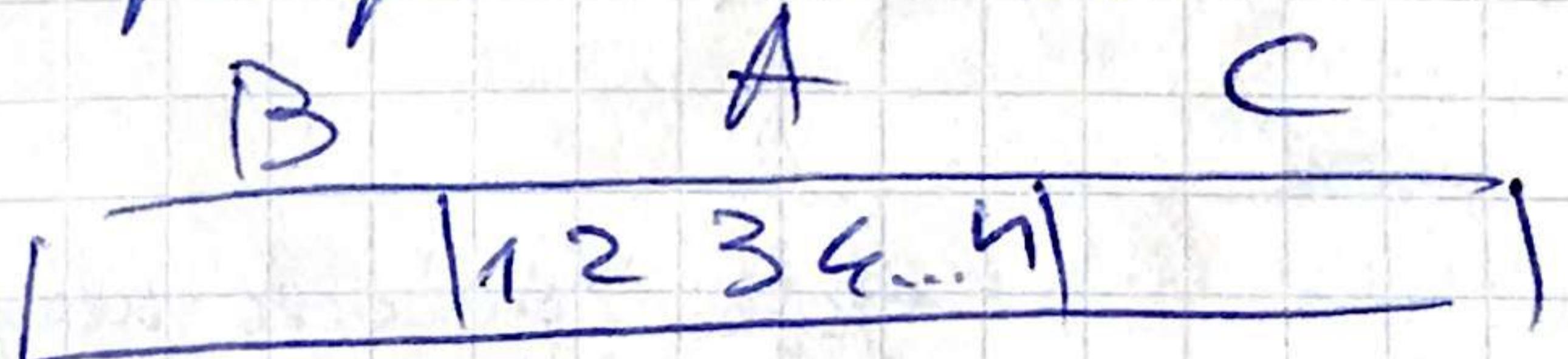
1)



std::rotate - O(n)

Merge (A, merge (B, C))

2) Рекурсия на массиве



reverse = 0 until 1

right <= 0      right > 0

Push(node)

8.12.20

## 1) Upgärdeggus &amp; DD

T - cartesian Tree

 $T[i] = X_{(i)} - i^2$  upgärdeggus expeccchaX - Ne - i addeler  
G upgärdeggus expeccchaToder expecccha neop  
pe3-T BIRRAR. one p. na uogggp.

Node { ... size ... &amp; value &amp; result }

Size (node):

referon node == nullptr?  $\emptyset$ : node  $\rightarrow$  size.// Size (node) == 1 + Size (Left (node))  
+ Size (Right (node))

UpdateNode (node)

if node != null:

node  $\rightarrow$  size = 1 + Size (node  $\rightarrow$  left)  
+ Size (node  $\rightarrow$  right)node  $\rightarrow$  result =operation (value  
 $\leftarrow$  node  $\rightarrow$  value,  
Reselt (node  $\rightarrow$  left),  
Reselt (node  $\rightarrow$  right))

Get (T, idx):

if T == null:

return  $\emptyset$ if idx  $\geq$  Size (T  $\rightarrow$  ref):return T  $\rightarrow$  x;else if Size (T  $\rightarrow$  ref)  $<$  idxreturn Get (T  $\rightarrow$  right, idx - Size (T  $\rightarrow$  ref) - 1)else: // Size (T  $\rightarrow$  ref)  $>$  idxreturn Get (T  $\rightarrow$  left, idx)

O(log n)

2) RMQ / RSQ

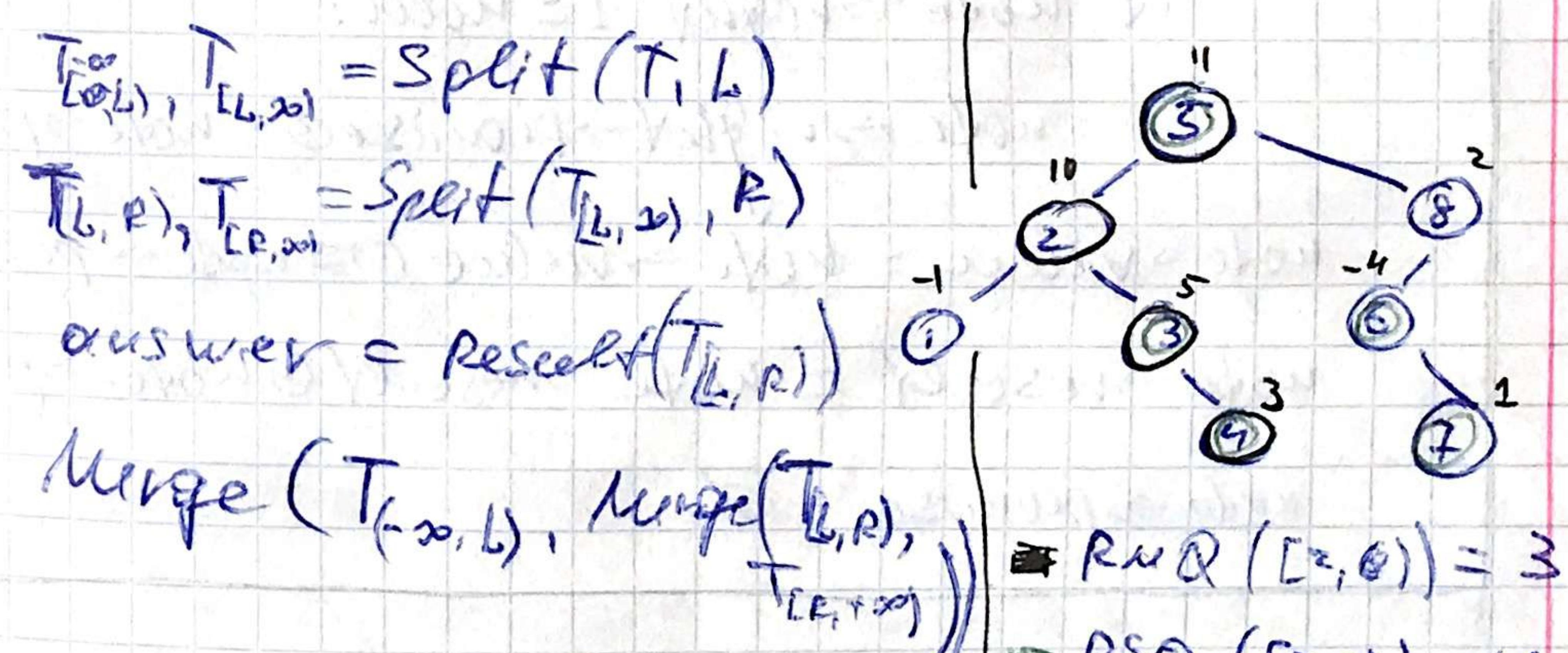
RMQ (L, R):

RMQ (L, R) - 3anuoc uakernym

rex 3ne-x:  $L \leq x \leq R$  $T_{[0,L]}, T_{[L,R]}$  = Split (T, L) $T_{[L,R]}, T_{[R,\infty)}$  = Split ( $T_{[L,R]}$ , R)answer = Reselt ( $T_{[L,R]}$ )Merge ( $T_{[-\infty, L]}$ , Merge ( $T_{[L,R]}$ ,  
 $T_{[R,\infty)}$ ))

RMQ ([2, 6]) = 3

RSQ ([3, 8]) = 16



### 3) Trynnobree oddebnen

update ( $[L, R]$ ,  $A$ )

$\xrightarrow{\text{new value}}$

$x: L \leq x \leq R$

(azerterece  $x, a$  value)

new value  
geesqayt gibra odzayay

Dobareen none "promise" & qzel gepeba  
→ TO Odebnen & ocepoys oddebnene.

### Propoekubaree (Beweknesece edegateem)

Push (node)

if node != null:

if node → left != null:

node → left → promise  $\oplus=$  node → promise

if node → right != null:

node → right → promise  $\oplus=$  node → promise

node → value = node → value  $\oplus=$  node → promise

node → result = node → result  $\oplus=$  node → promise

node → promise =  $\emptyset$

Update ( $T, [L, R], A$ ):

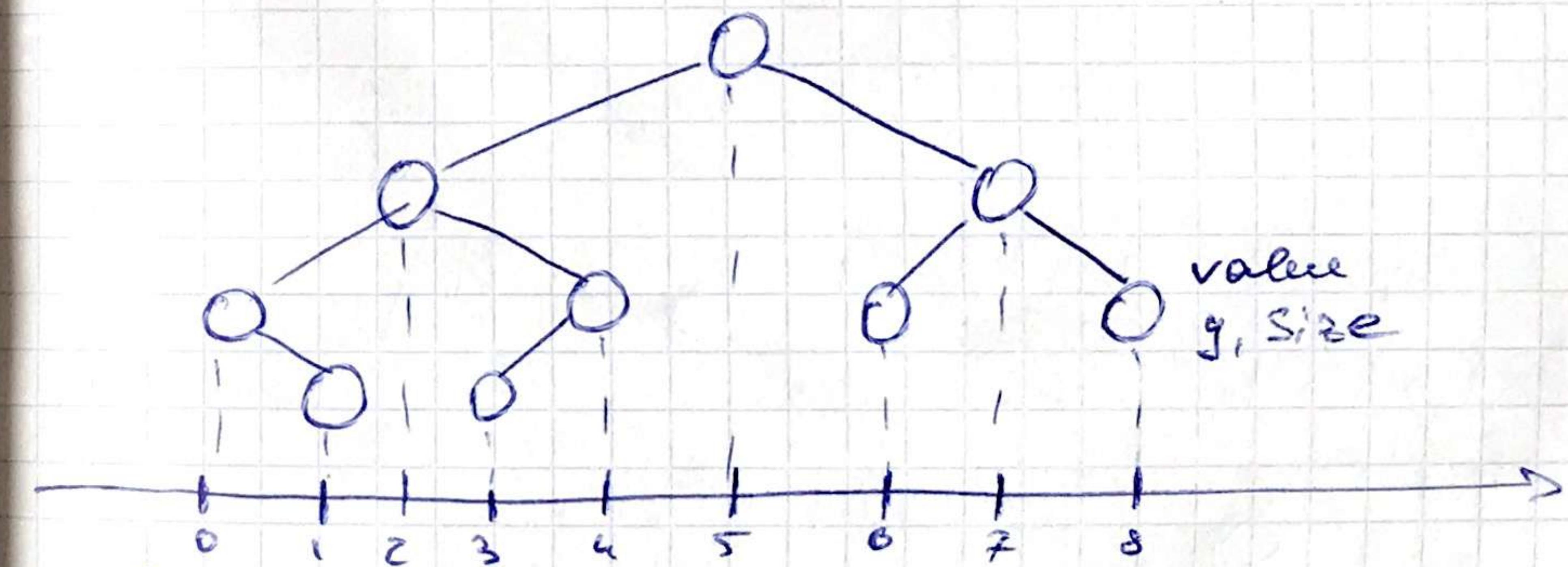
$T_{[-\infty, b]}, T_{[L, \infty)} = \text{Split}(T, L)$

$T_{[L, R]}, T_{[R, \infty)} = \text{Split}(T_{[L, \infty)}, R)$

$T_{[L, R]} \rightarrow \text{promise } \oplus= A$

merge...

4) DD no teesborey khor. ( $X = \text{size}(left)$ )



Dobareen yonnum khore az gepeba

Split ( $T, K$ )

if  $\text{size}(T \rightarrow \text{left}) < K$ :

$T_1, T_2 = \text{Split}(T \rightarrow \text{right}, K - \text{size}(T \rightarrow \text{left}) - 1)$

## "Дискретный массив"

- 1) Вставка в конец массива за  $O(\log n)$
- 2) Удаление  $O(\log n)$
- 3) Индексирование  $O(\log n)$
- 4) RMQ/LSQ  $O(\log n)$
- 5) Update  $O(\log n)$
- 6) Shift (уничтожение  $k$  первых элементов)  
 $O(\log n)$

10.12.2020 Поиск подстроки в строке

P - паттерн, S - строка начиная с i, j:  $S[i..j] = P$

### 1) Brute-Force (Привидимый)

for  $i=0$  to  $S.size - P.size$ :  $\leftarrow O(|S| - |P|)$

if ( $S[i..i+P.size-1] == P$ ):  $\leftarrow O(|P|)$

FoundMatch()

...

Итог:  $O(|P| \cdot (|S| - |P|))$  - Time

$$|P| \approx \frac{|S|}{2}$$

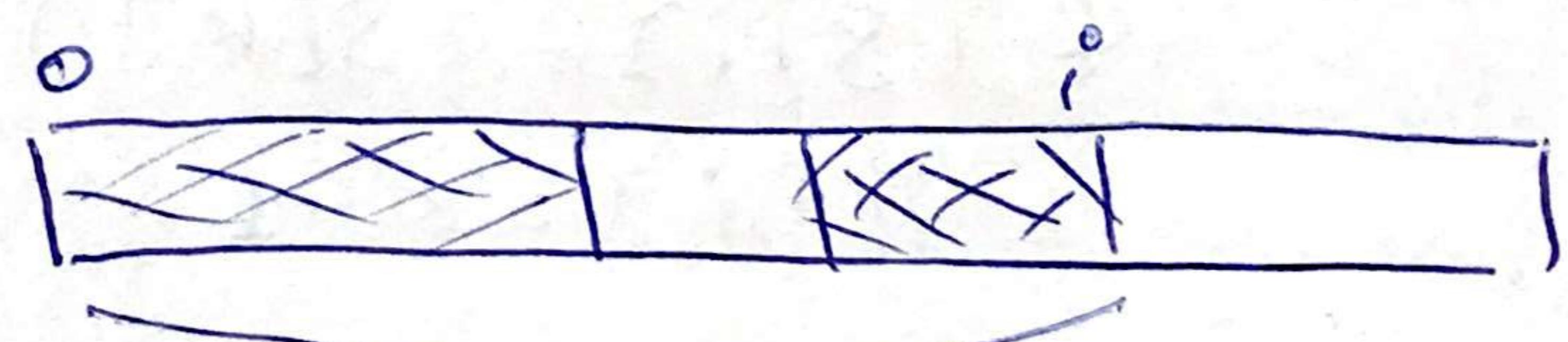
$$O\left(\frac{|S|}{2} \cdot \frac{|S|}{2}\right) = O(|S|^2)$$
 - Time

$$O(1)$$
 - Memory

### 2) Префикс-функция - функция от строки S и позиции i ( $< n$ ) такая что: $\pi(S, i)$ = длина наивысшего

составленного префикса, ~~от~~ который совпадает с суффиксом префикса строки S длины  $i + 1$ .

$S[0..i]$  - префикс



$S[i..len-1]$  - суффикс

$$\pi(S, i) = \max_k \{ k : S[0..k-1] = S[i-k+1..i] \}$$

Пример: абвабвабвабв

$$\pi: 0 \ 0 \ 0 \ 1 \ 2 \ 1 \ 2 \ 3 \ 4$$

Префикс-функция от 0 равна 0

Это приведенный алг.  $O(n^3)$   $n = |S|$

S

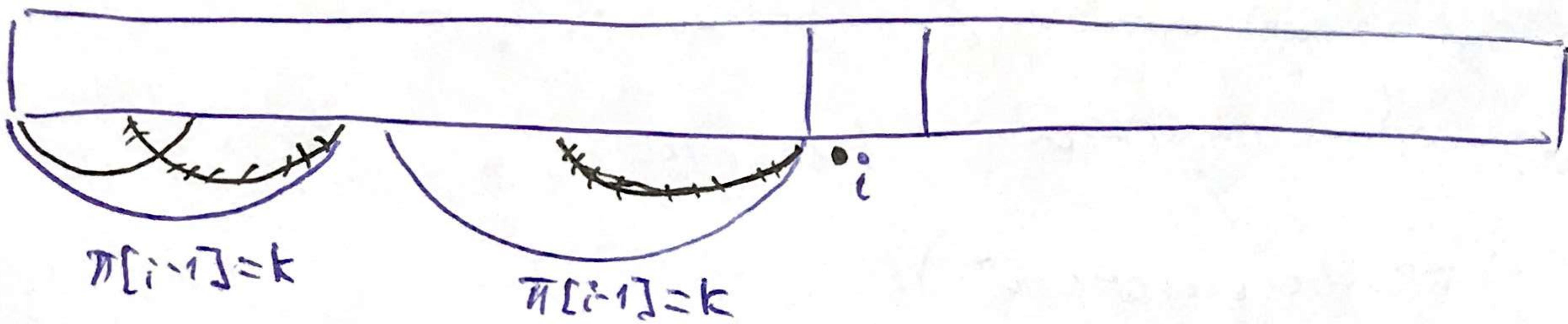
\_\_\_\_\_

$\pi$

0 ~~абв~~ ~~абв~~ ~~абв~~ ?

$$i-1 \quad i$$
  
$$\pi(i-1) = k$$

$$K \quad K \quad \pi(i) \leq k+1$$



if ( $s[i] == s[k+1]$ ):

$\pi[i] = k+1$

else:

$k = \pi[k+1]$

Prefix ( $s$ ):

$\pi = [0\dots 0] \quad // \text{size} = |s|$

for  $i$  in  $[1\dots |s|-1]$ :

$k = \pi[i-1]$

while ( $k > 0$  and  $s[i] != s[k]$ ):

$k = \pi[k-1]$

if ( $s[i] == s[k]$ ):

$\pi[i] = k+1$

return  $\pi$ ;

Время работы:

на каждого символа  
авторитет  $k$  надо  
увеличиваться на 1 (1 pos),  
и это уменьшается  
бесконечно мал

$\sum n_-$  - здесь зерно увеличения  $k$

$\sum n_+$  - ||| увеличение  $k$

$$T = O(\sum n_- + \sum n_+) = O(|s|)$$

$$\sum n_+ < |s|$$

$$\sum n_- < \sum n_+ < S$$

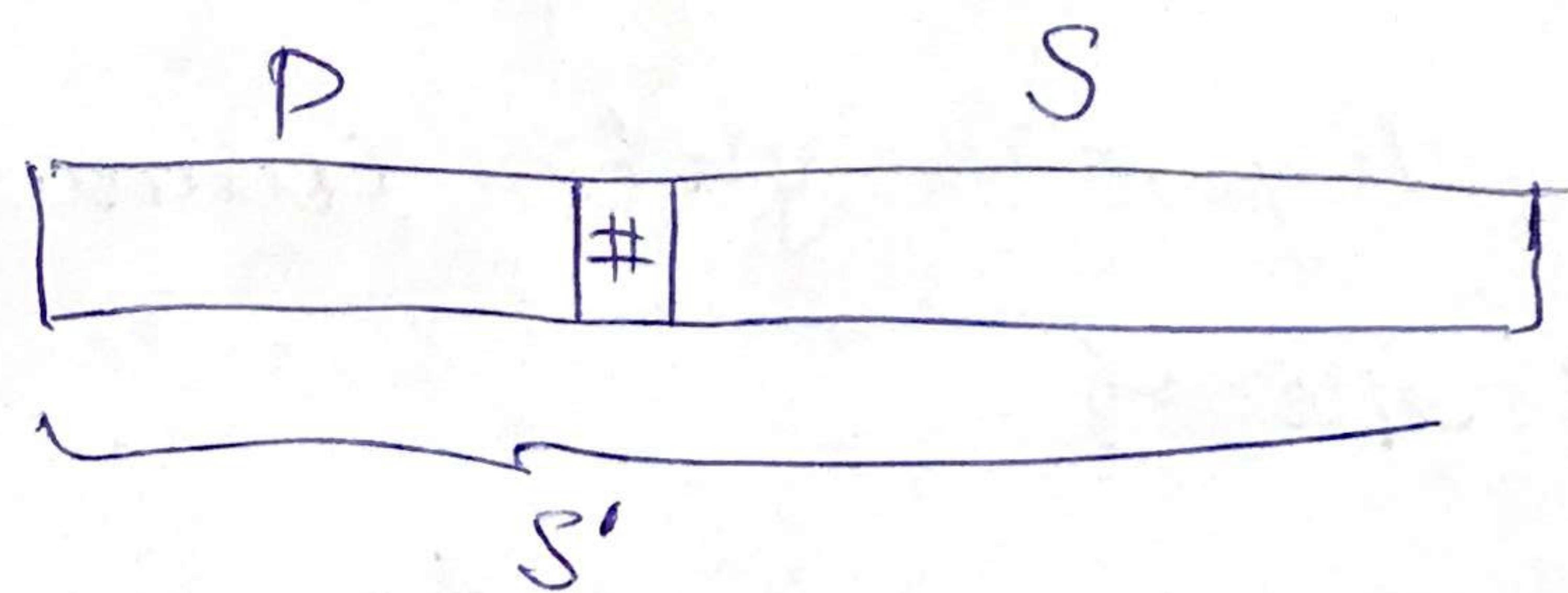
### 3) Алгоритм Кнудса - Морриса - Пратта

P-шаблон

S - строка текста вхождение P в S

$$\textcircled{1} \quad S' = P + " \# " + S$$

разделитель  
(ан-т, который  
не встречается  
ни в P ни в S)



\textcircled{2} Рассмотрим префикс ф-ю от  $S'$   
(или Z-функцию)

\textcircled{3} Находим все позиции, в которых  $\pi[\Sigma^i] = |P|$

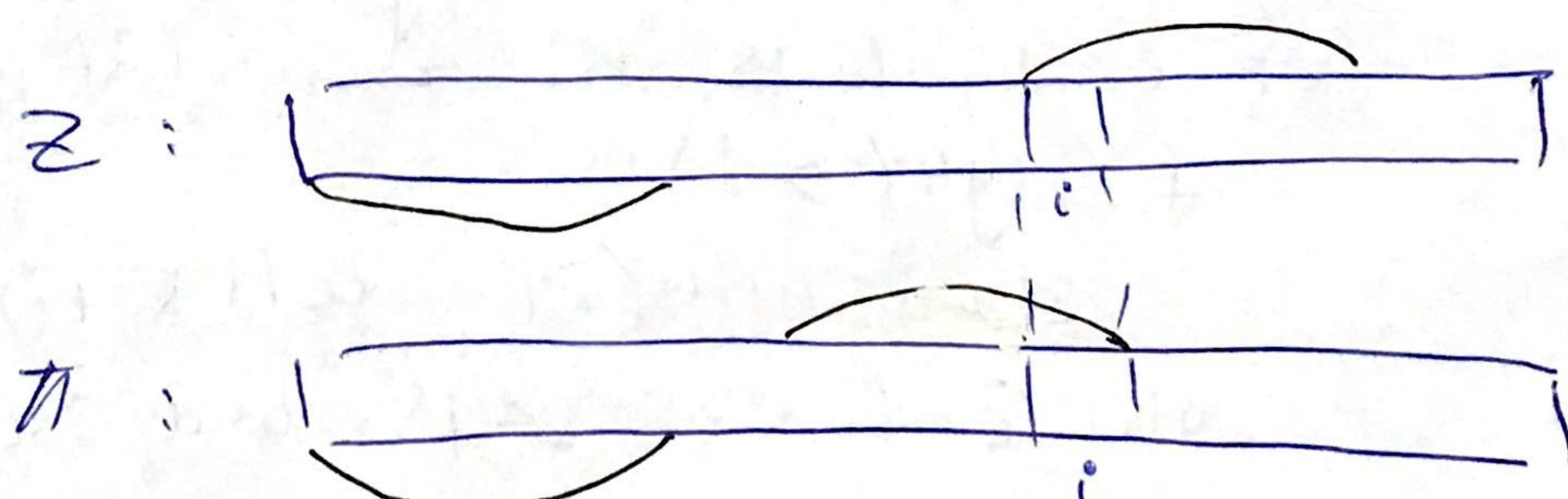
Время работы:  $O(|P| + |S|) = O(|S|)$   
 $|P| < |S|$

4) Z-функция — функция от строки S, называемая

такая что:  $Z(S, i) = \max k : S[i \dots i+k-1] = S[0 \dots k-1]$   
сущность  $S[i \dots]$ , который является  
префиксом S

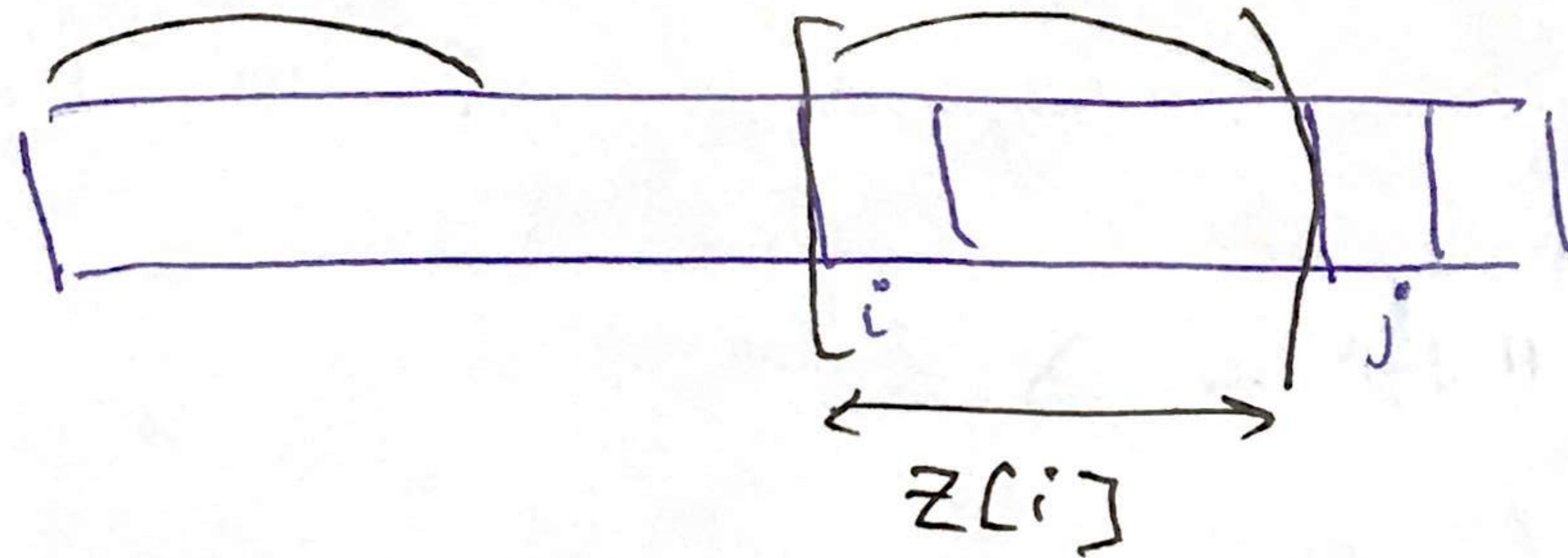
$$Z(S, i) = \max k : S[i \dots i+k-1] = S[0 \dots k-1]$$

$$Z(S, 0) = 0$$



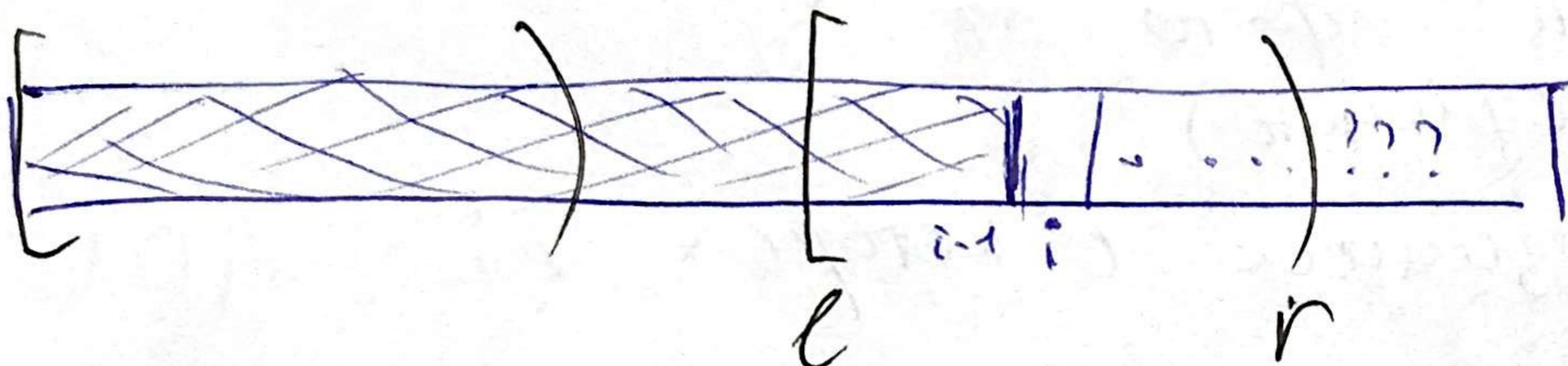
Тривиальный алгоритм  $O(|S|^2)$

Z-shok - napa ( $i, j$ ):  $Z[i] = j - i$



Rycie right - canas upobor spaleczaor bcez  
Z-shokow

left - ... debas yatacza



right > i

Unguanuzasza:  $Z[i] = \min(Z[i - \text{left}], \text{right} - i)$

while ( $i + Z[i] < |S|$  and  $S[Z[i]] = S[i + Z[i]]$ )  
 $\quad \quad \quad ++Z[i]$

$Z(S)$ :

$Z = [0 \dots 0]$

$\text{left} = 0$

$\text{right} = 0$

for  $i = 1$  to  $|S| - 1$ :

if ( $\text{right} > i$ ):

$Z[i] = \min(Z[i - \text{left}], \text{right} - i)$   $O(1)$

while ( $i + Z[i] < |S|$  and  $S[Z[i]] = S[i + Z[i]]$ )  
 $\quad \quad \quad ++Z[i]$

if ( $i + Z[i] > \text{right}$ )

$\text{left} = i$

$\text{right} = i + Z[i]$

return  $Z$

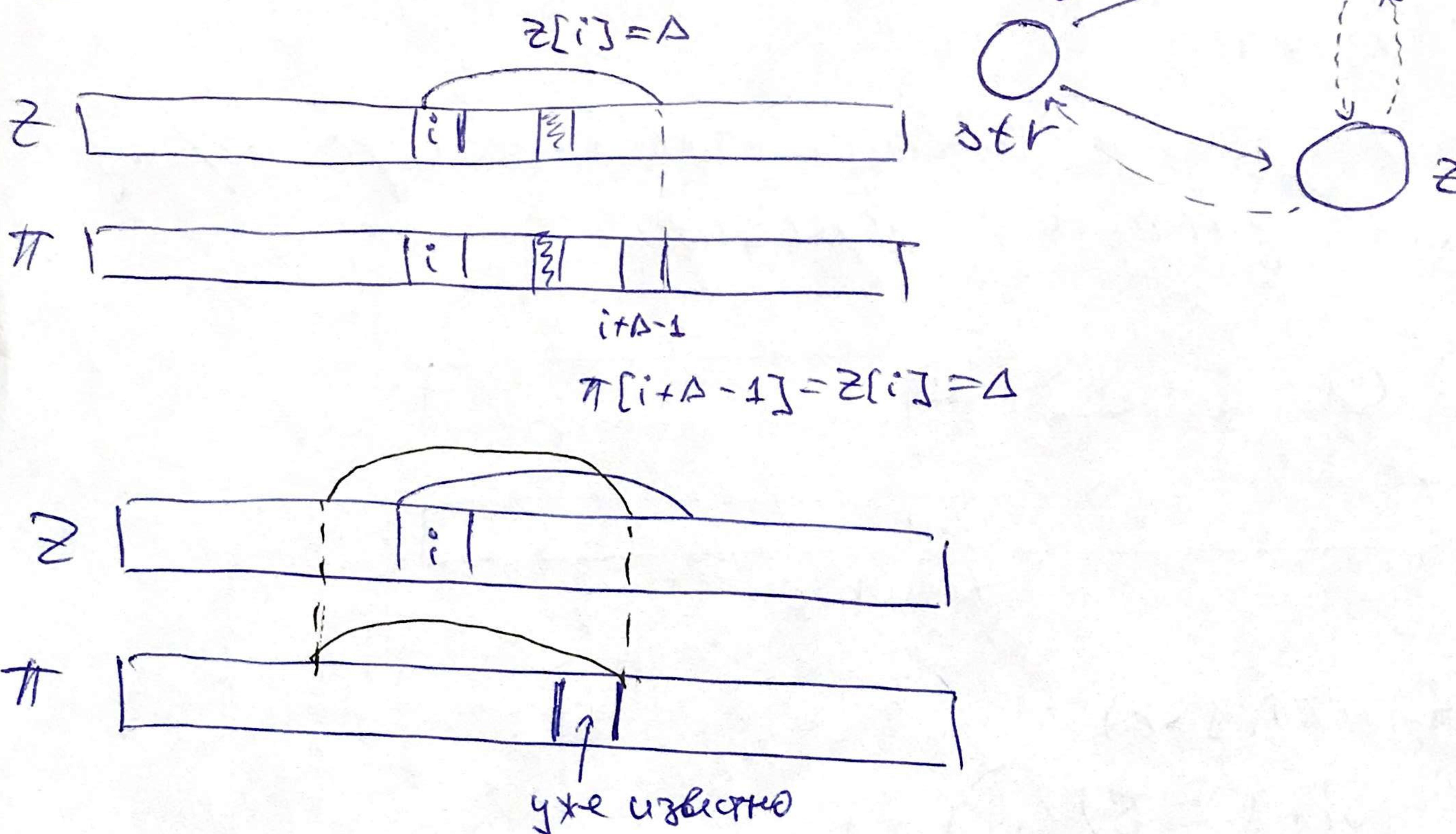
Breme podora:  $O(|S|)$

aceotura right.

Na kaxgou urepazun uod0  
reaxoguu OTCET za  $O(1)$ ,  
uod0 cyburulen right, HO  
right re uoxer epelkey/6co dneue  
zaen  $|S|$  nay

$$\sum_{i=1}^n (1+a_i) \leq S+S = O(S)$$

5)  $Z \rightarrow \pi$



$Z \& \pi(z)$ :

$$\pi = [0 \dots 0]$$

for  $i = 0$  to  $|Z|-1$ :

    for  $\Delta = Z[i]-1$  to  $\Delta \geq 0$ :

        if ( $\pi[i+\Delta] > 0$ )

            break;

$\pi[i+\Delta] = \Delta + 1$

    return  $\pi$

Break poldra.

$O(|S|)$

a b c a b a b c a

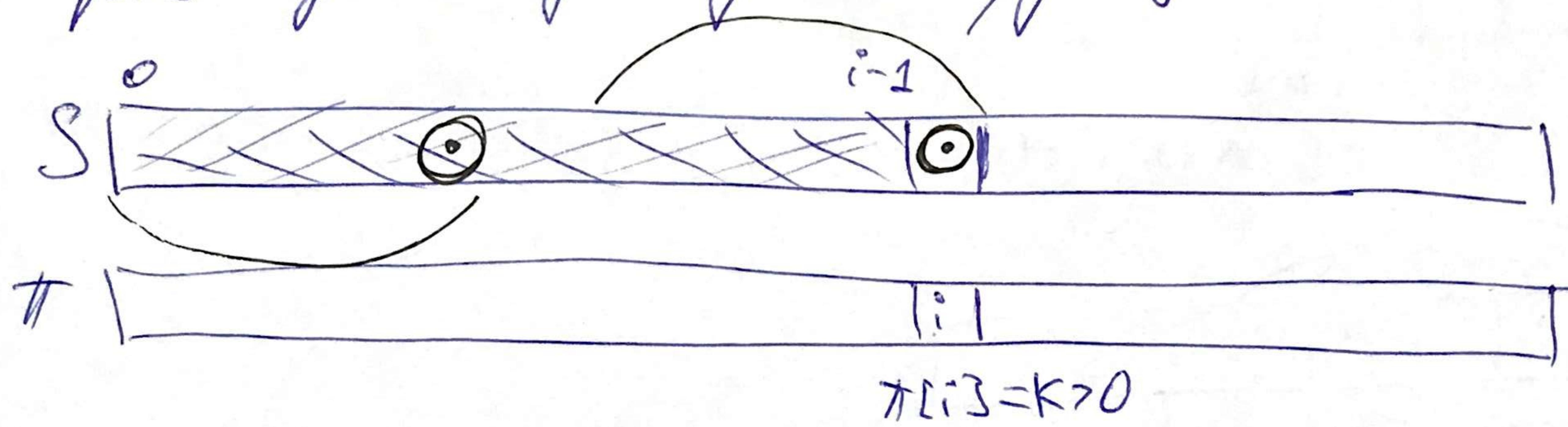
$$Z = \overbrace{0 \ 0}^1 \otimes \overbrace{2 \ 0}^2 \ 4 \ 0 \ 0 \ 1$$

$$\pi = 0 \ 0 \ 0 \ 1 \ 2 \ 1 \ 2 \ 3 \ 4$$

## 6) $\pi \rightarrow \text{str}$

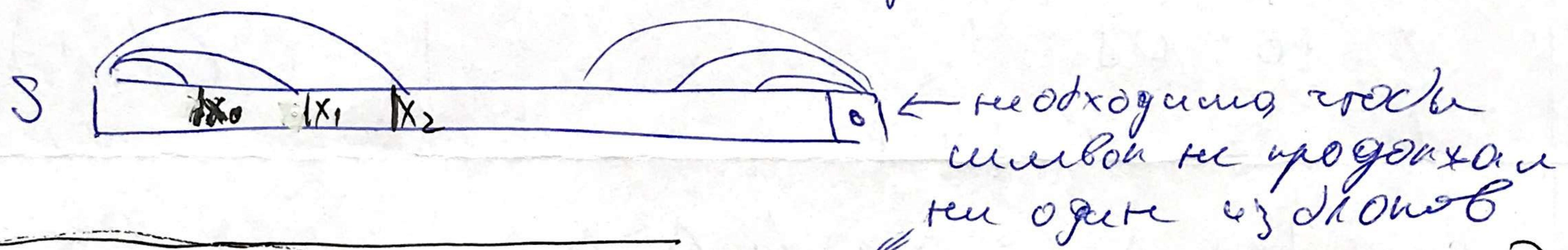
Задача:  $\pi$  - паттерн

Найти самое длинное палиндромное подстроку, которая есть в заданном  $\pi$ -паттерне.



1. if ( $\pi[i] > 0$ ):  
 $S[i] := S[\bar{\pi}[i]-1]$

2. if ( $\pi[i] = 0$ ): // делает простой поиск -  
- подбирает новый символ



$\pi \rightarrow \text{str}(\pi)$ :

sfr = "";

sfr[0] = 'a';

for  $i=1$  to  $|\pi|-1$ :

if  $\pi[i] > 0$ :

$S[i] := S[\bar{\pi}[i]-1]$

else:

если не встречалась  
символ, ее добавляется  
к sfr

$O(|\pi|)$

used = {} // хэш-массив символов, которые есть в  $\pi$

$k = \pi[i-1]$

while ( $k > 0$ ):

used.add( $S[k]$ )

$k = \pi[k-1]$

3@ практика

анализ  $O(11)$

refine str

for symbol = 'a' to 'z':

if (symbol not in used)

$S[i] = \text{symbol}$

break