

## Алгоритм Форда-Фалкерсона

th поток  $f$  - максимальный  $\Rightarrow$

$\Leftrightarrow \emptyset G_f$  нет пути из  $s$  в  $t$

D-ко.  $\Rightarrow$  Доказано  $\emptyset G_f \Rightarrow$  нет пути из  $s$  в  $t$

$\Rightarrow$  поток  $f' \in G_f$



$\Rightarrow f + f'$ -поток  $\in G_f$ ,  
таким

поскольку  $|f + f'| > |f|$  ( $f$  не достиг максимума)

$\Leftarrow$  Рассмотрим максимальный поток, а  $f$ -поток, который ведет к текущему моменту

$\Delta f' = f_{\max} - f \stackrel{12}{=} \text{поток } \in G_f$ ,

то т.к.  $\emptyset G_f$  нет пути из  $s$  в  $t$   $\Rightarrow$

$\Rightarrow |f'| \leq 0 \rightarrow |f| \geq |f_{\max}|$ , т.к.  $|f| = |f_{\max}|$

1) С помощью DFS находят либо из  $s$  в  $t$  иначе не текущий максимальный поток

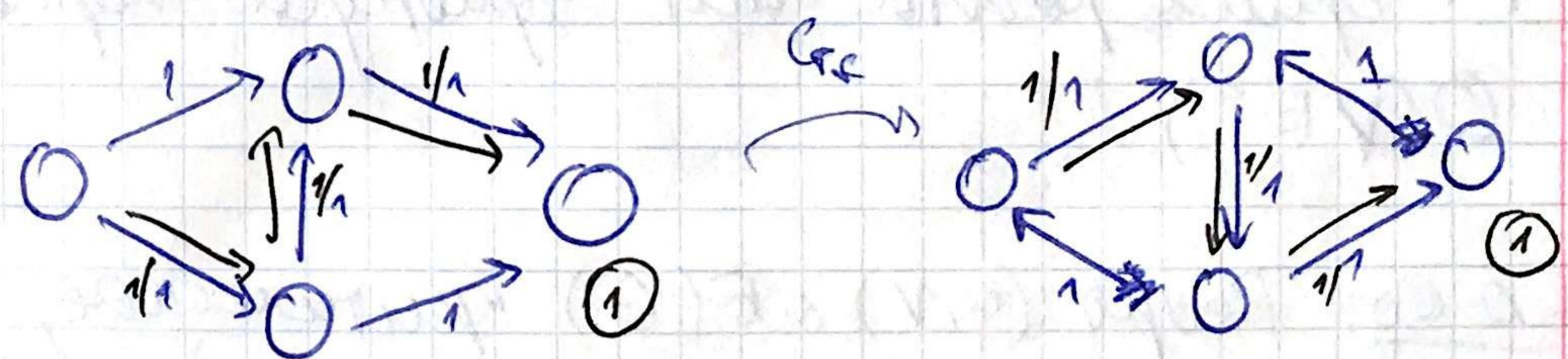


2) С помощью остаточного сета  $G_f$   $C_f = C - f$

3) С помощью DFS находят либо из  $s$  в  $t$  иначе

из  $s$  в  $t$  и ...

2-3) Повтор пока есть пути из  $s$  в  $t$

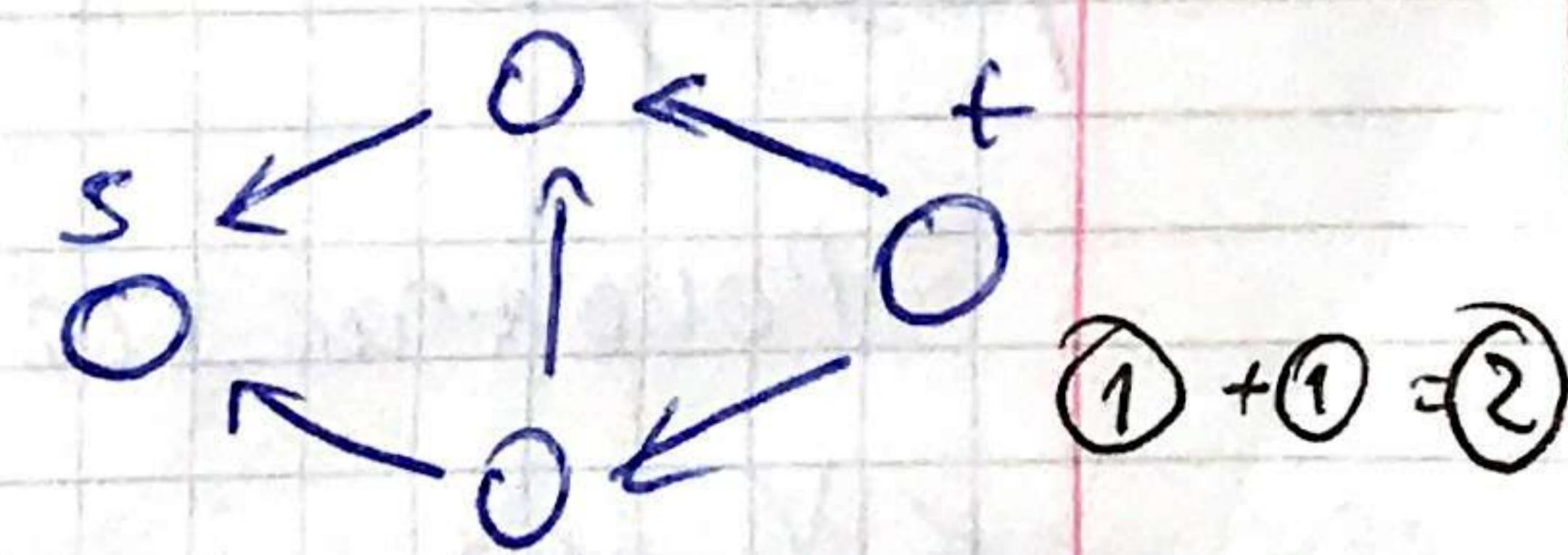


## Время работы

DFS нескольких раз

$$O(E) \cdot O(|f|) = O(E \cdot |f|)$$

$(O(V+E))$



поток  
максимальный

## Алгоритм Эдмондса-Карпа

Dfs  $\rightarrow$  Bfs решает все проблемы  
(искажающее кратчайшее пути)

Время работы:  $O(VE^2)$   $E \approx V^2$   
 $O(V^3)$

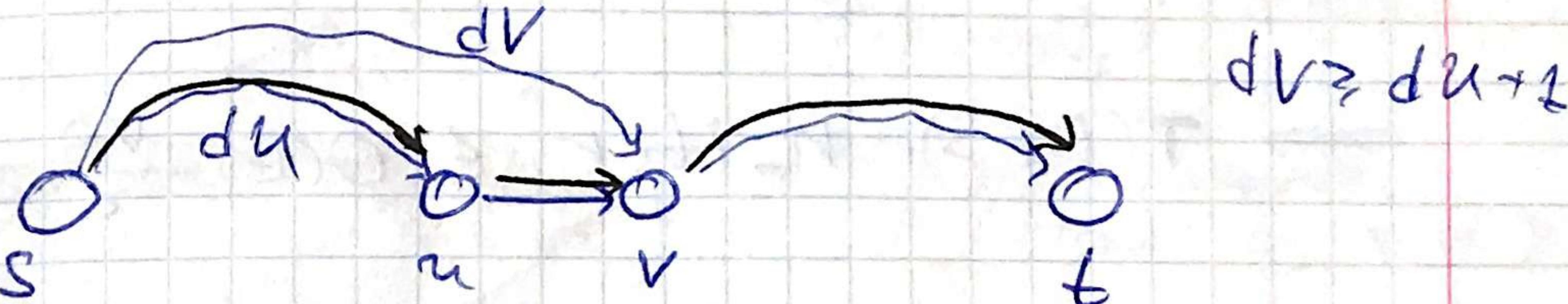
Корректность выходит из th Рордса-Ранкенсона

th Время работы алг. Эдмондса-Карпа  
 $O(VE^2)$

Доказательство: Ребро  $(u, v) \in E(G)$  критическое,  
если на пути из  $u$  в  $v$  нет никаких ненулевых  
размеров

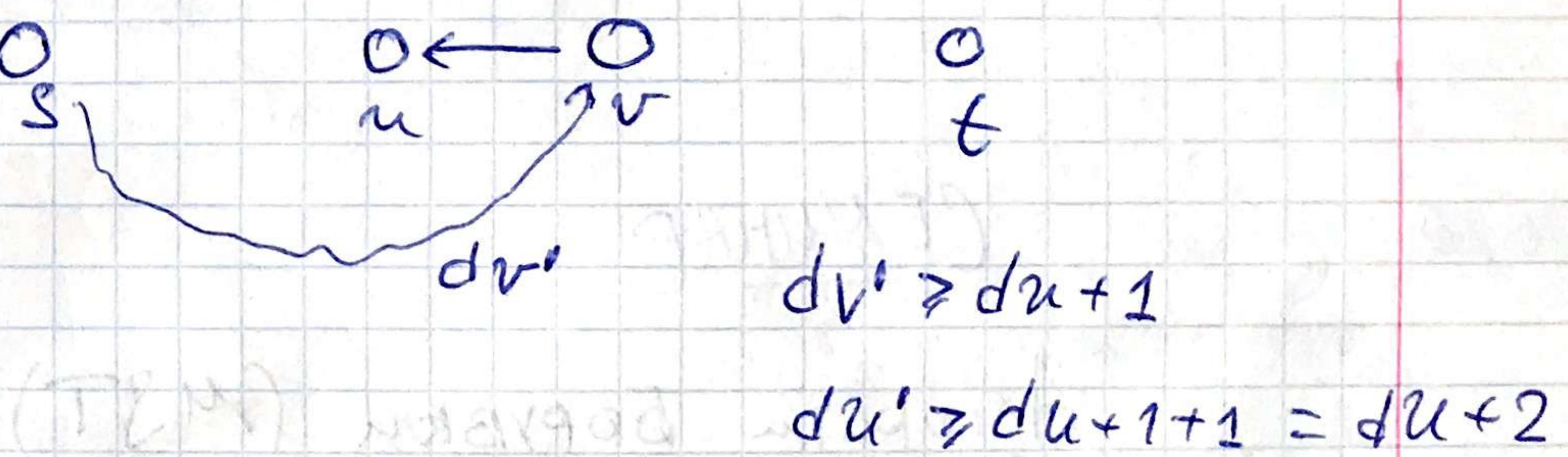
Покажем, что ребро может быть критическим  
 $\Leftrightarrow V \text{ раз}$

1) Ребро  $(u, v)$  блокирует статус критического



$(u, v)$  критическое покрытие в  $G_f$

2)  $\Leftrightarrow$   $2^{d_u}$  номера, когда  $(u, v)$  станет  
критическим



$$du' = dv' + 1 > du + 1 + 1 - du + 2 \Rightarrow$$

$\Rightarrow$  критическое для  $u$  несет увеличение  
на единицу  $\frac{V}{2}$  раз  $\Rightarrow (u, v) \leq \frac{V}{2}$  раз критическое

Всего ребер в графе  $E$ , и находят на  $\frac{V}{2}$   
разах из ребер становятся критическими  $\Rightarrow$

$\Rightarrow \text{длительность} \leq \frac{V \cdot E}{\sum}$  времени

$\Rightarrow T(\text{BFS}) \cdot n_{\text{итер}} \leq O(E) \cdot \frac{V \cdot E}{\sum} = O(V \cdot E^2)$

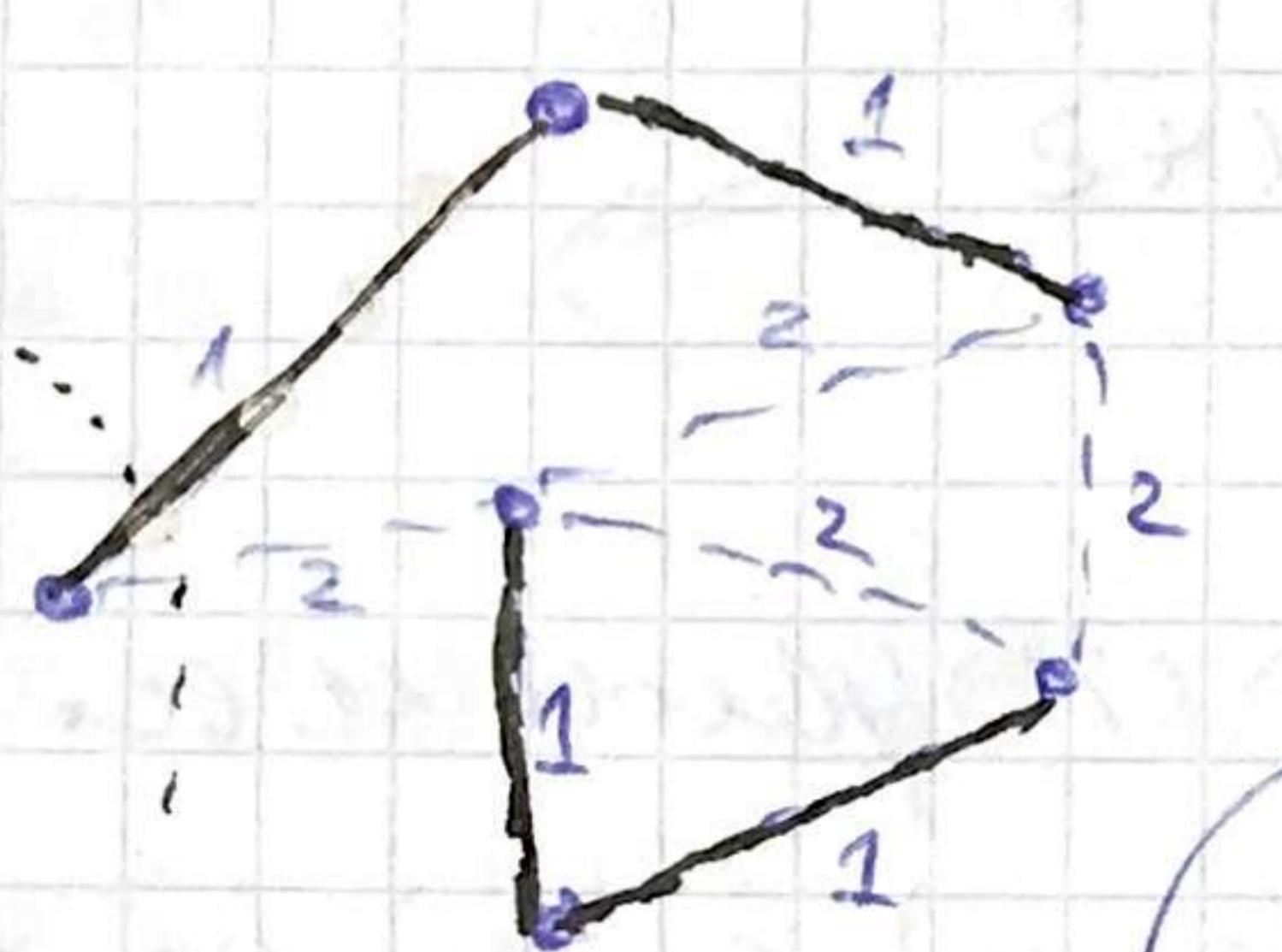
Алгоритм Дейкстры  $O(V^2 E)$

Задержка  $O(V(V+E))$

24.10.20

## СЕМИНАР

### Алгоритм Борюбкин (MST)



Лемма о диагональном подграфе

$e(\text{from}, \text{to}, \text{weight})$

Коды на C++:  $O(E)$

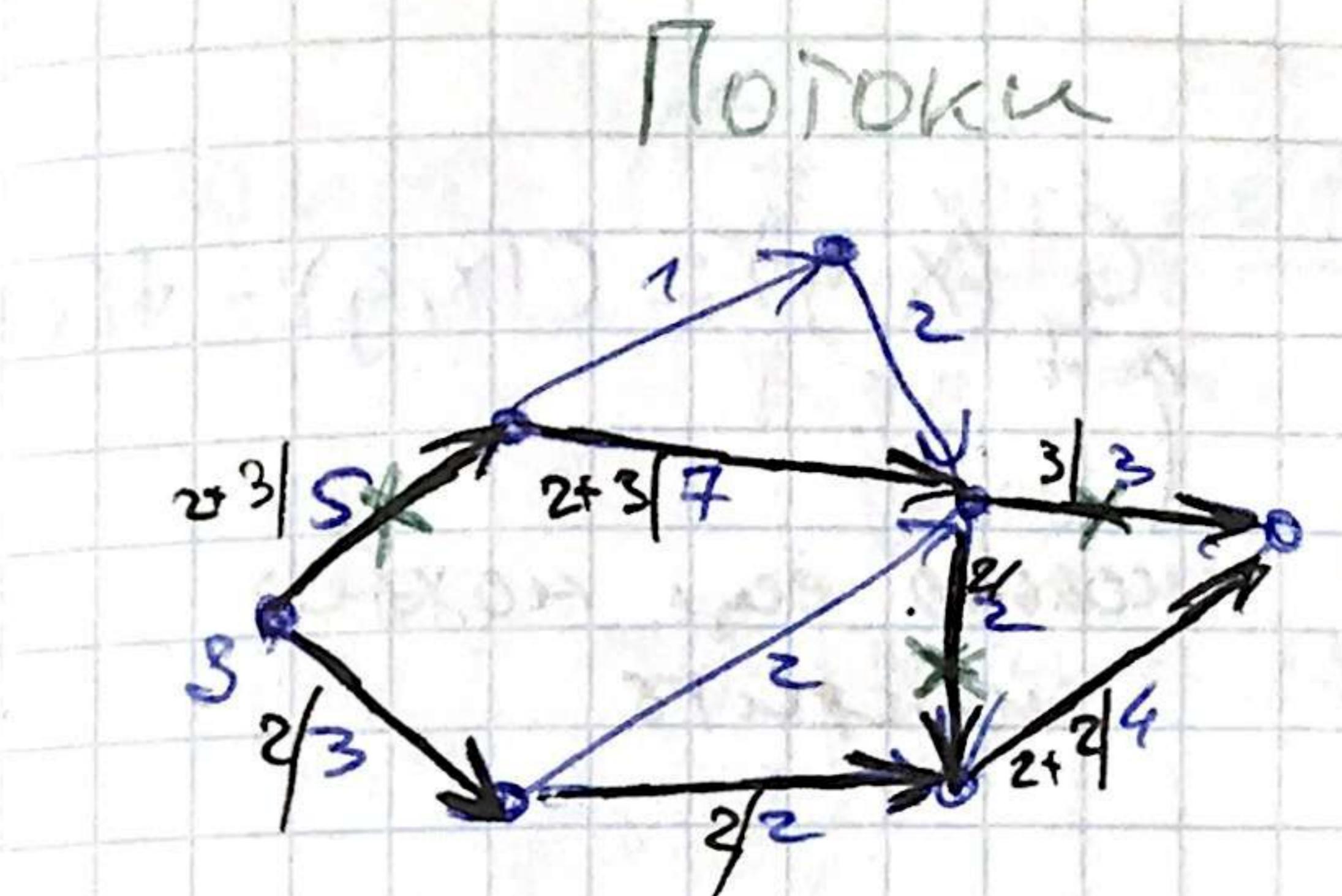
Без кода:  $O(\log V)$

ИТОГ:  $O(E \log V)$

CMP  
compare weight(a, b)

if ( $a.\text{weight} < b.\text{weight}$ )  
return True  
if  $>$  return False

True & if  $a.\text{number} < b.\text{number}$

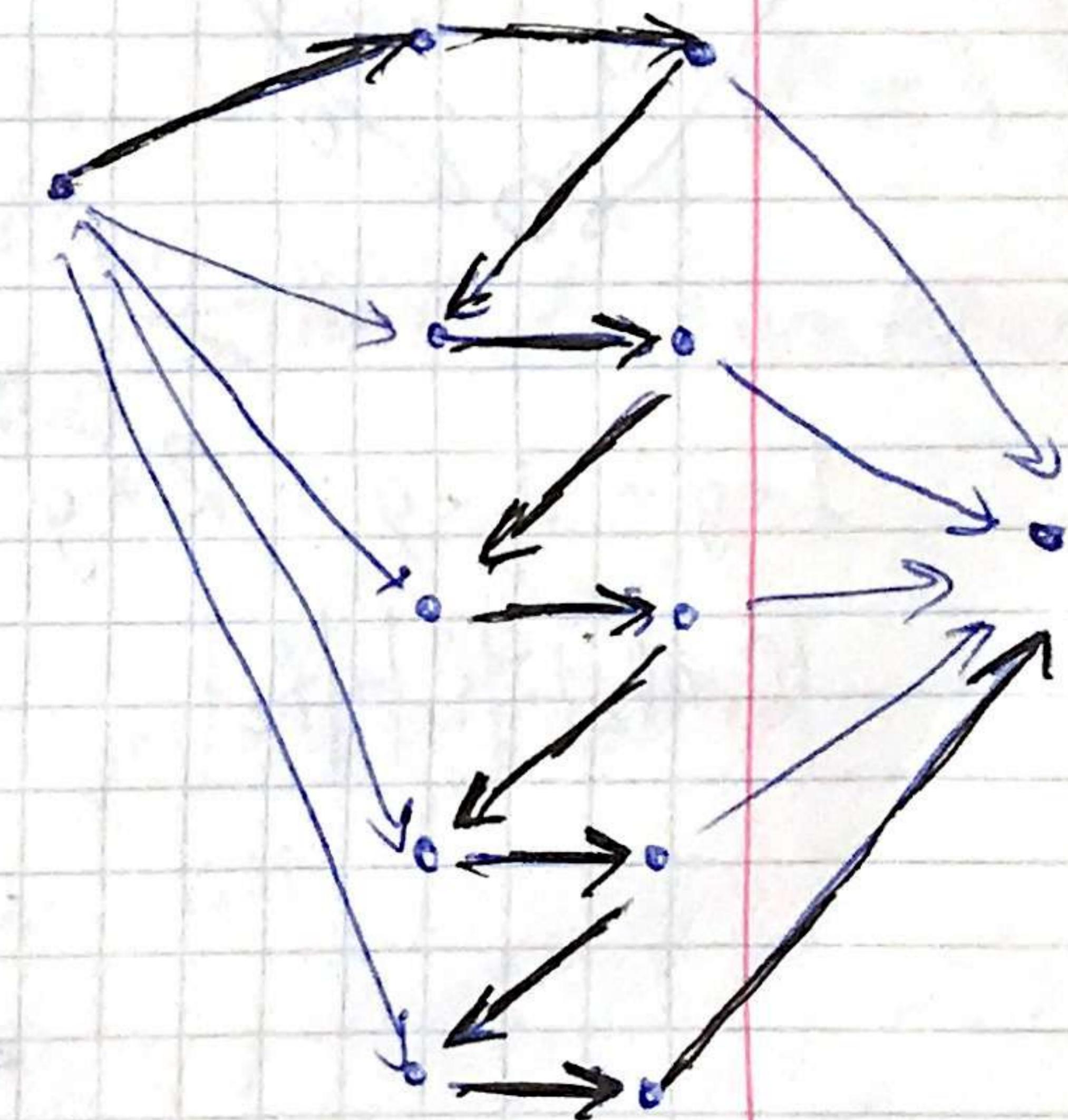
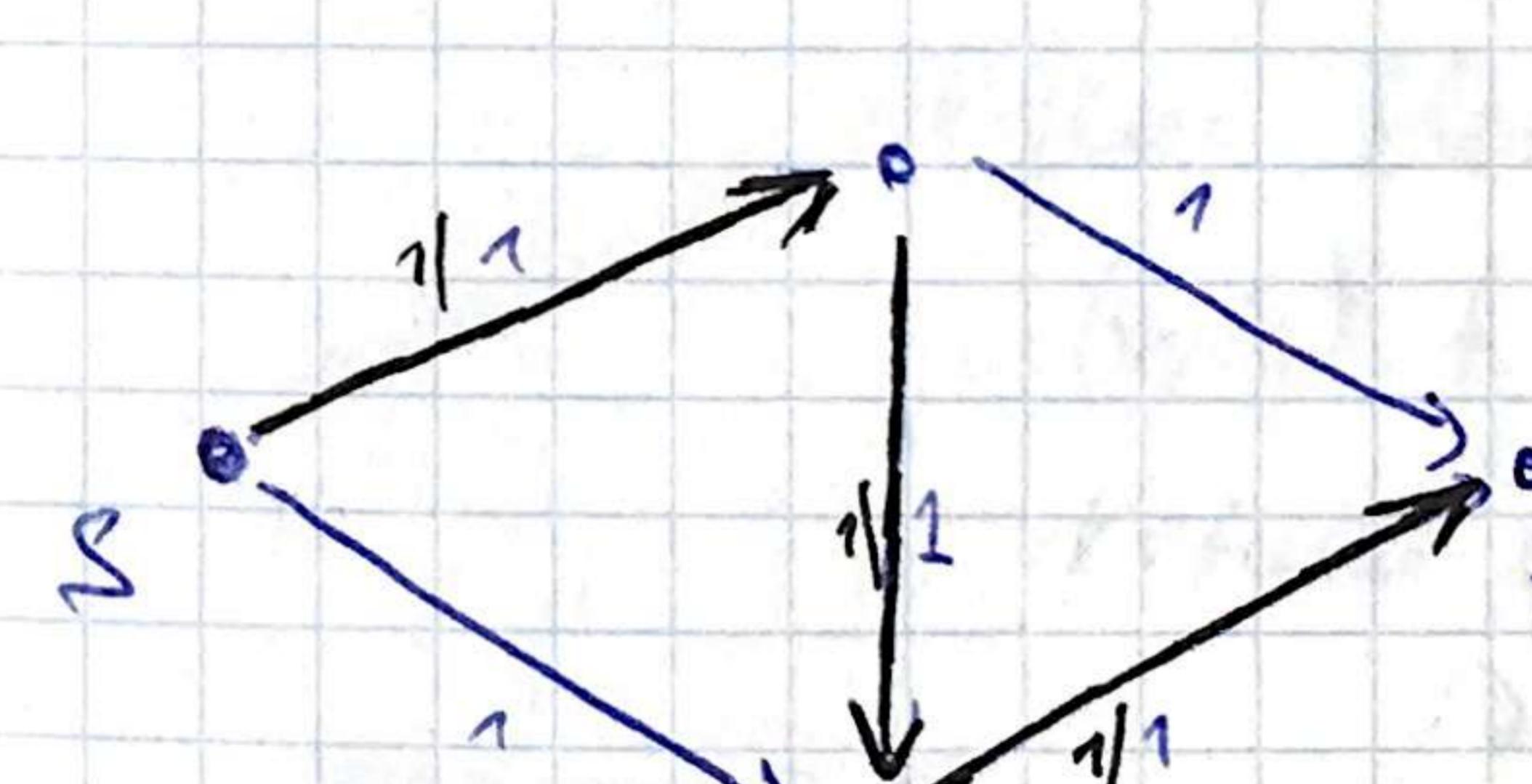


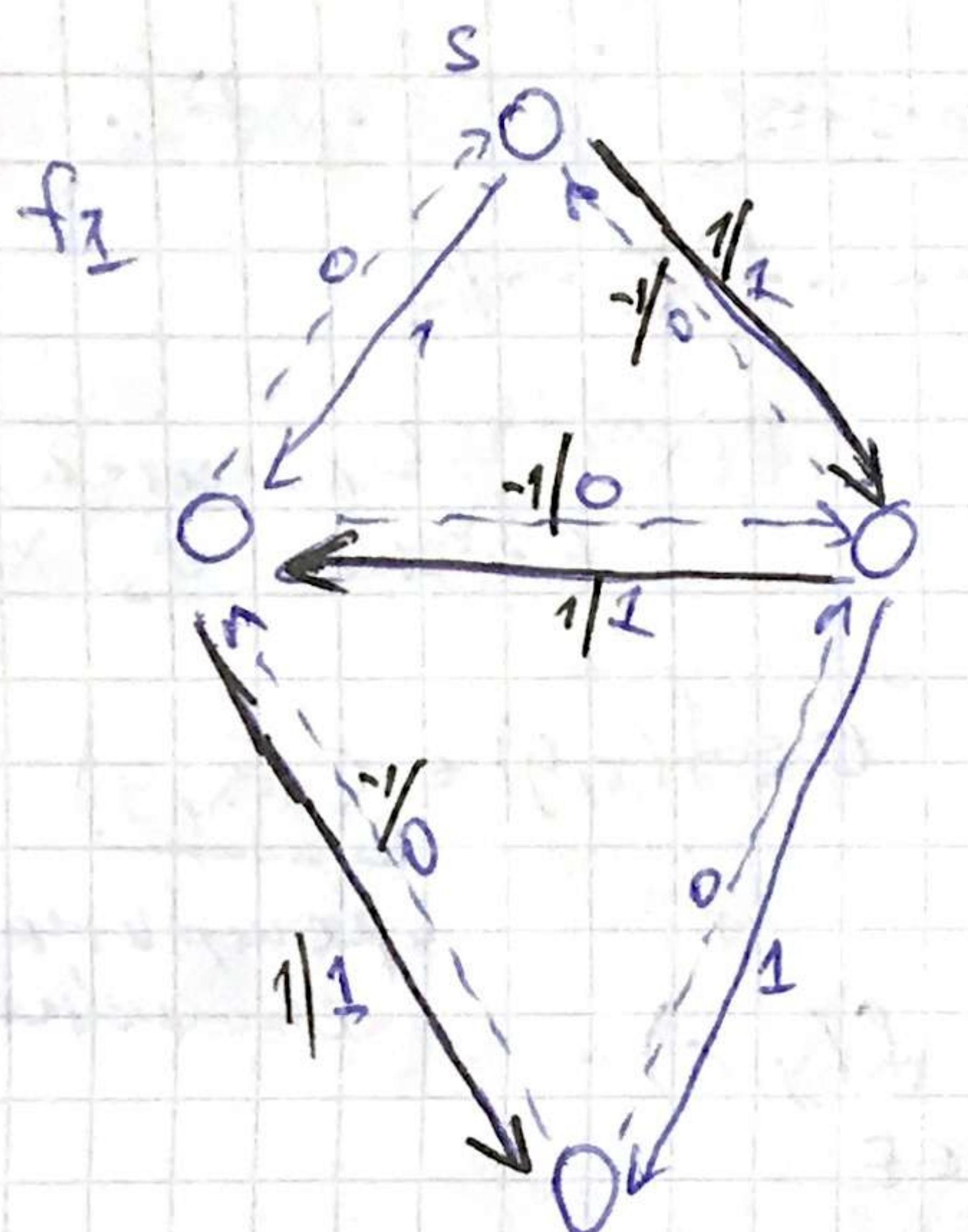
$$\forall x \neq s, t \quad \sum_{(x,y) \in E} f(x,y) = \sum_{(y,x) \in E} f(y,x)$$

$$|f| = \sum_{(s,y) \in E} f(s,y) = \sum_{(y,t) \in E} f(y,t)$$

$f: E \rightarrow \mathbb{R}_+$   
 $f(x,y)$  - стоимость  
маршрута  $x \rightarrow y$

$0 \leq f(x,y) \leq C(x,y)$   
упомянутое  
свойство

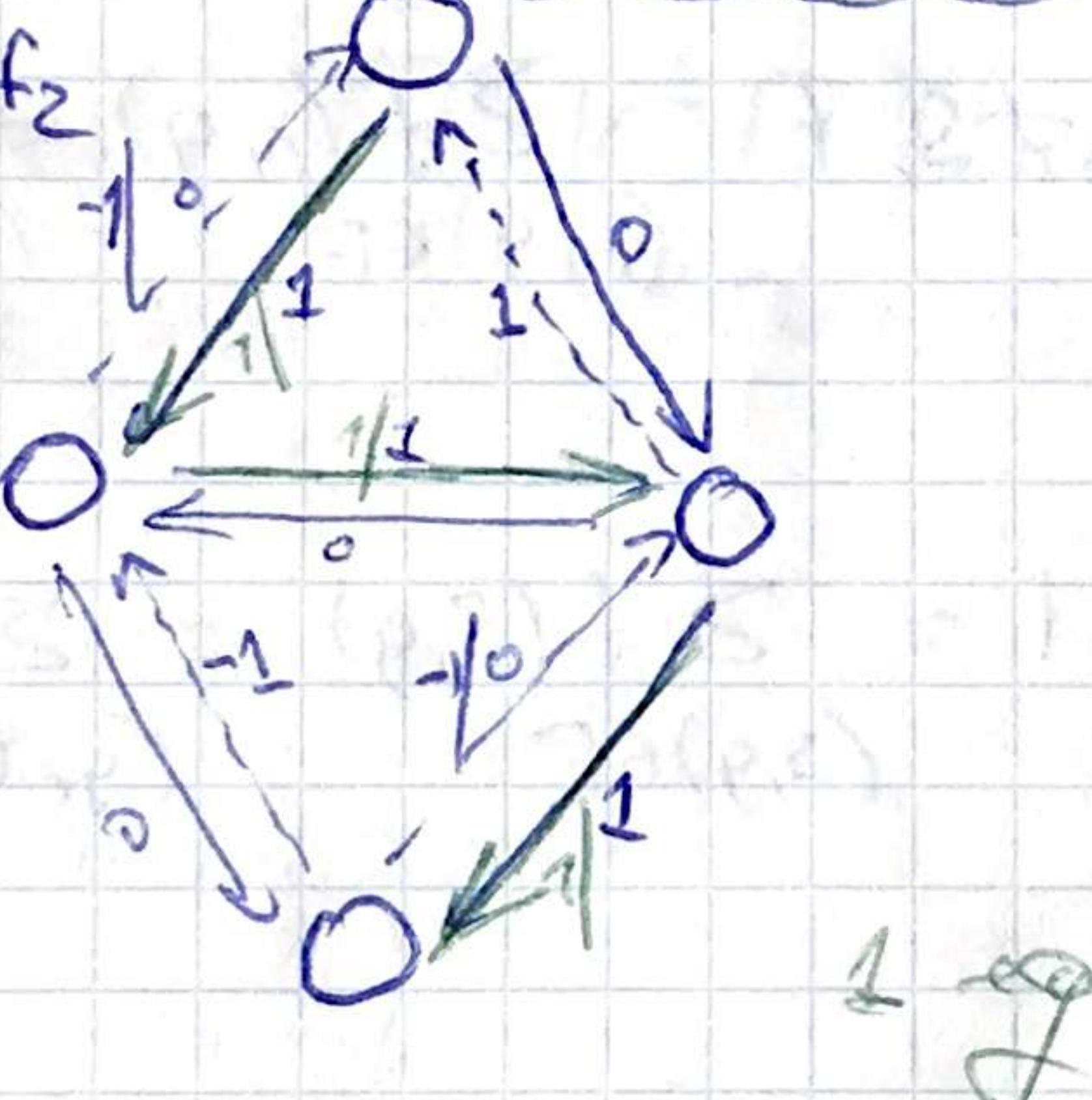




$$G_f(x, y) = (f(x, y)) - f(x, y)$$

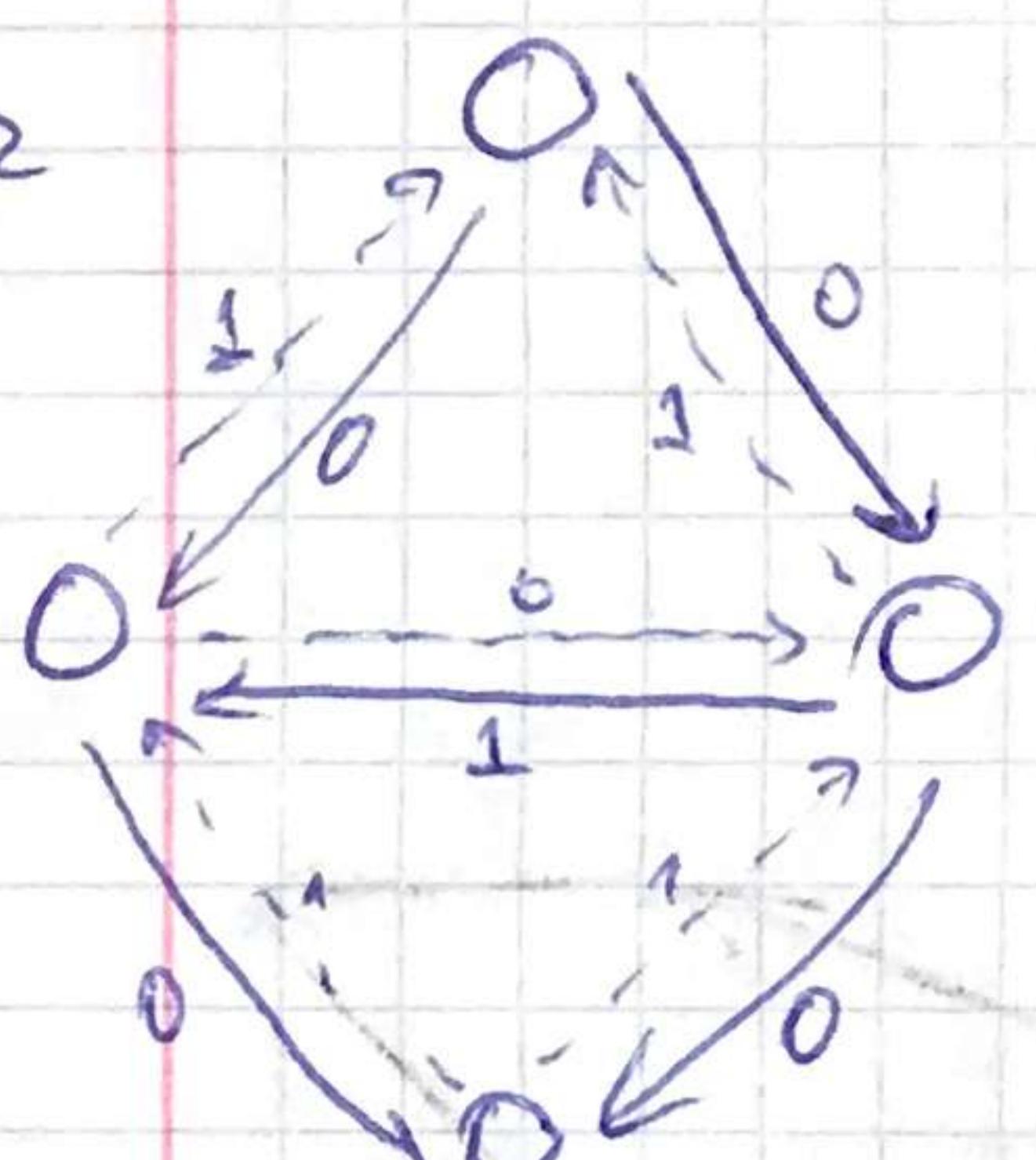
когдa edge maxed  
уничтож

OCTAТОRНАЯ СЕТЬ:



1-eg.

$f_1 + f_2$



1-eg + 1-eg = 2-eg. уничтож

$$|f_1 + f_2| = |f_1| + |f_2|$$

$$\begin{matrix} & \downarrow & \\ + & \xrightarrow{\quad} & = \end{matrix}$$

while ( $\exists G_f.\text{path}(s, t)$ ):

$f' = \text{FindFlow}(G_f, s, t, \infty) \leftarrow O(V+E)$

↑ (DFS)

редо min беека таңызу

$$f+ = f'$$

$O(|f|)$  үтепозгүй

FindFlow( $G, s, t$ , cur\_flow):

colors[v] = gray  $\leftarrow$  if ( $v = t$ ) return cur\_flow  
for ( $(v, u) \in G.E(v)$ ):

if (colors[u] == white  $\& f(v, u).c(v, u) > 0$ ):

cur\_flow = FindFlow( $G, u, t, \min(f(v, u).c(v, u), t, cur_flow)$ )

if (cur\_flow > 0):

$$(v, u).f+ = cur_flow$$

return cur\_flow

$(u, v).f - = cur_flow$

return 0

struct Edge {

int from;

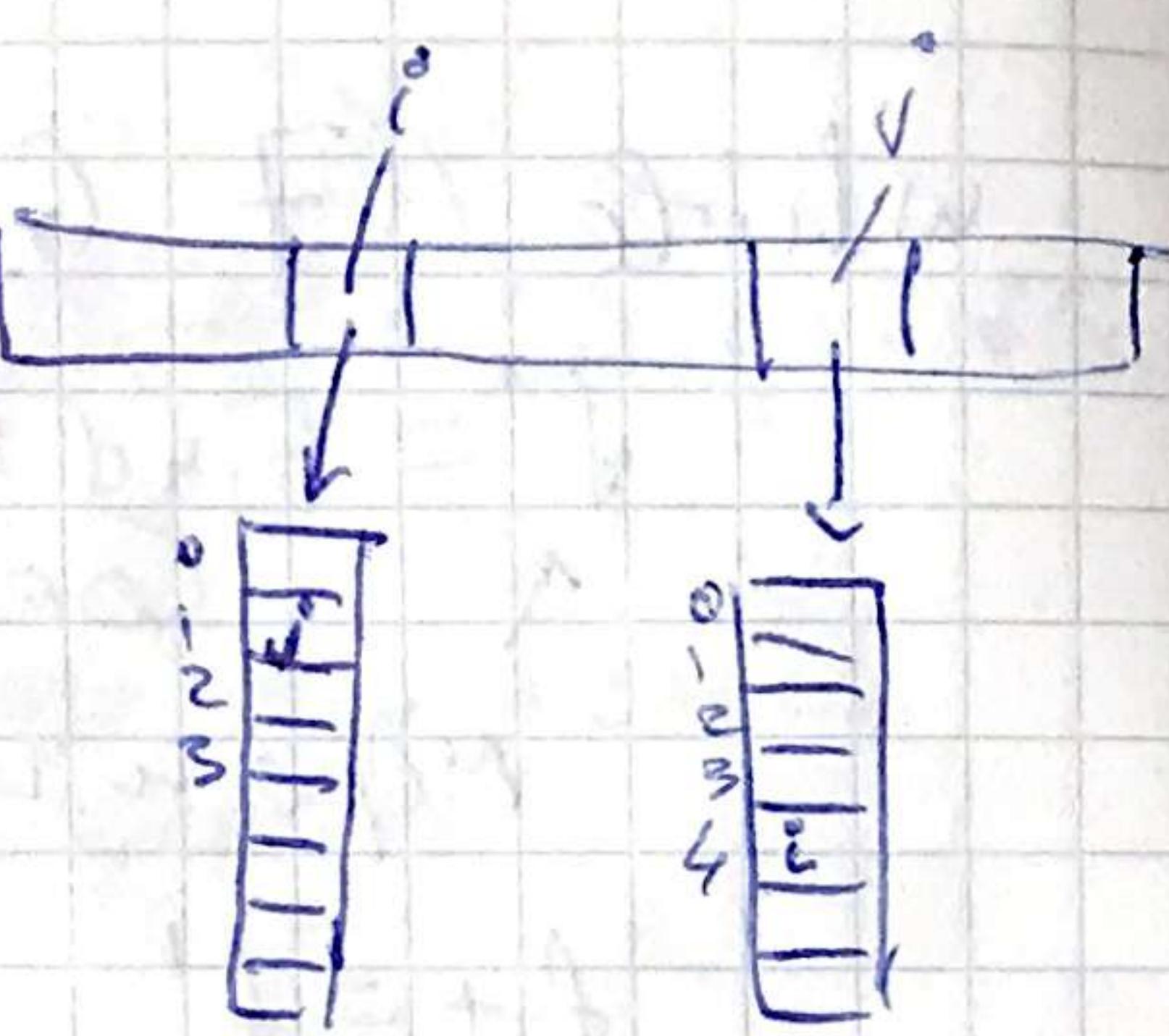
int to;

weight\_t flow;

weight\_t capacity;

int back;

}



(i,j).back == 2

(j,i).back == 1

adj\_list[j][i].back

(j,i)

weight\_t FindFlow (const Graph &graph)

vertex\_t start, vertex\_t finish

size\_t

: gotha za 24.10.20 ~ 19<sup>02</sup>

(u,v) - здесе одознай. може бити від u до  
єдине підключення від u до v.

## Алгоритм Эдмунда-Карпа

DFS → BFS

even шаг кратний

$O(VE^2)$

$O(V^5)$

$O(E)$

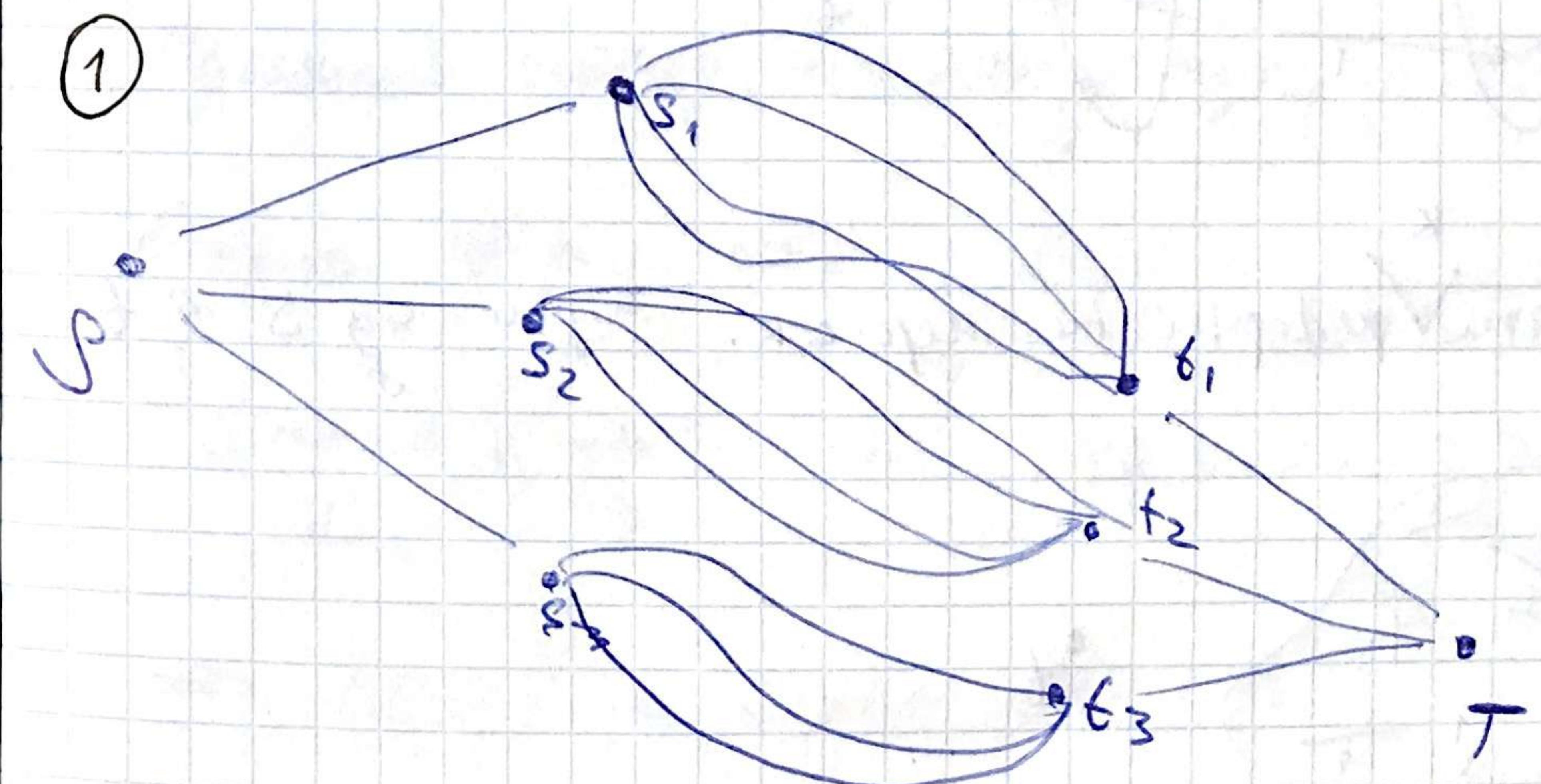
$O(VE)$

BFS

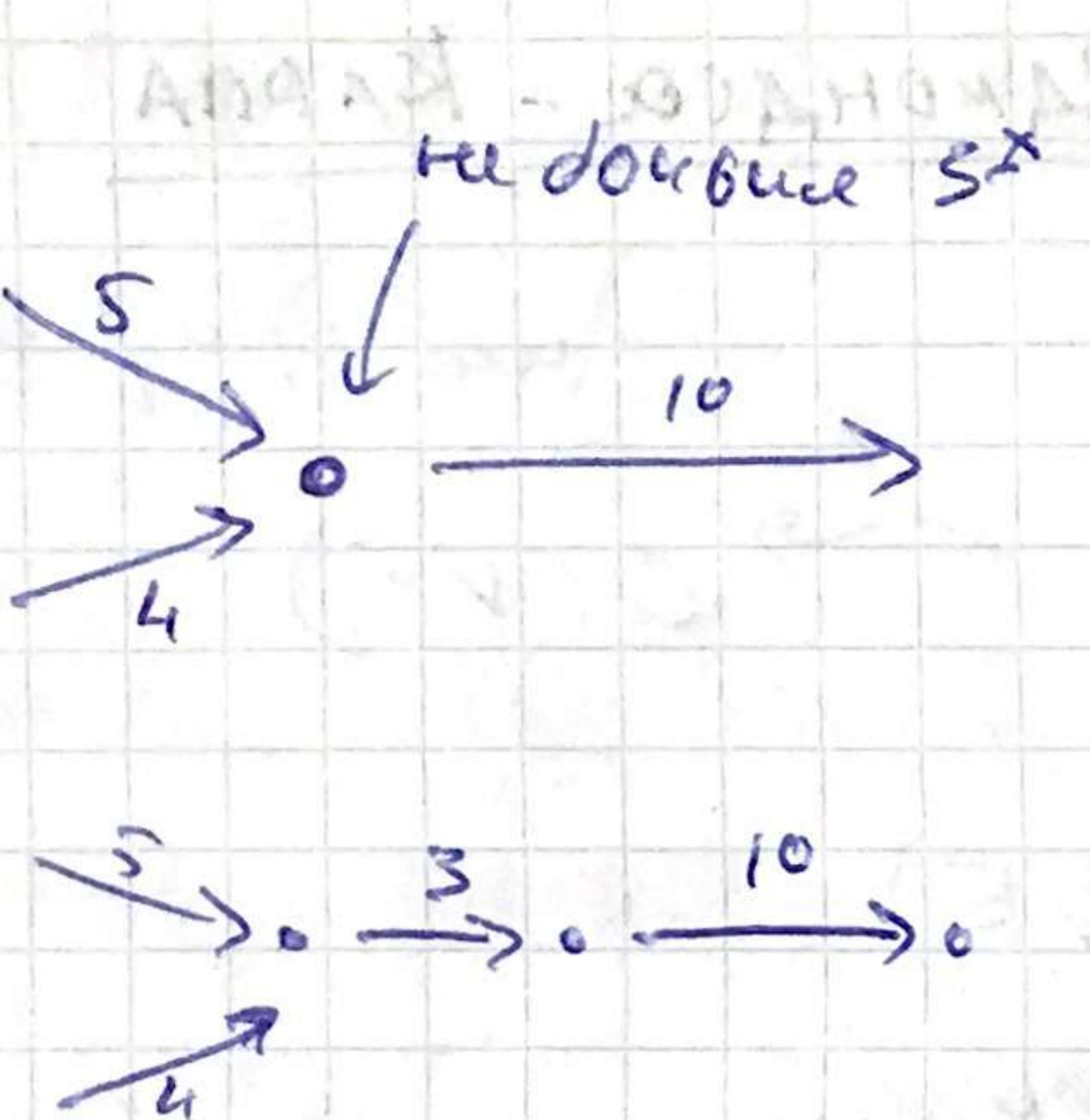
непрервно

Алгоритм  $O(EV^2) \sim O(V^4)$

$O((V+E)V) \sim O(V^3)$

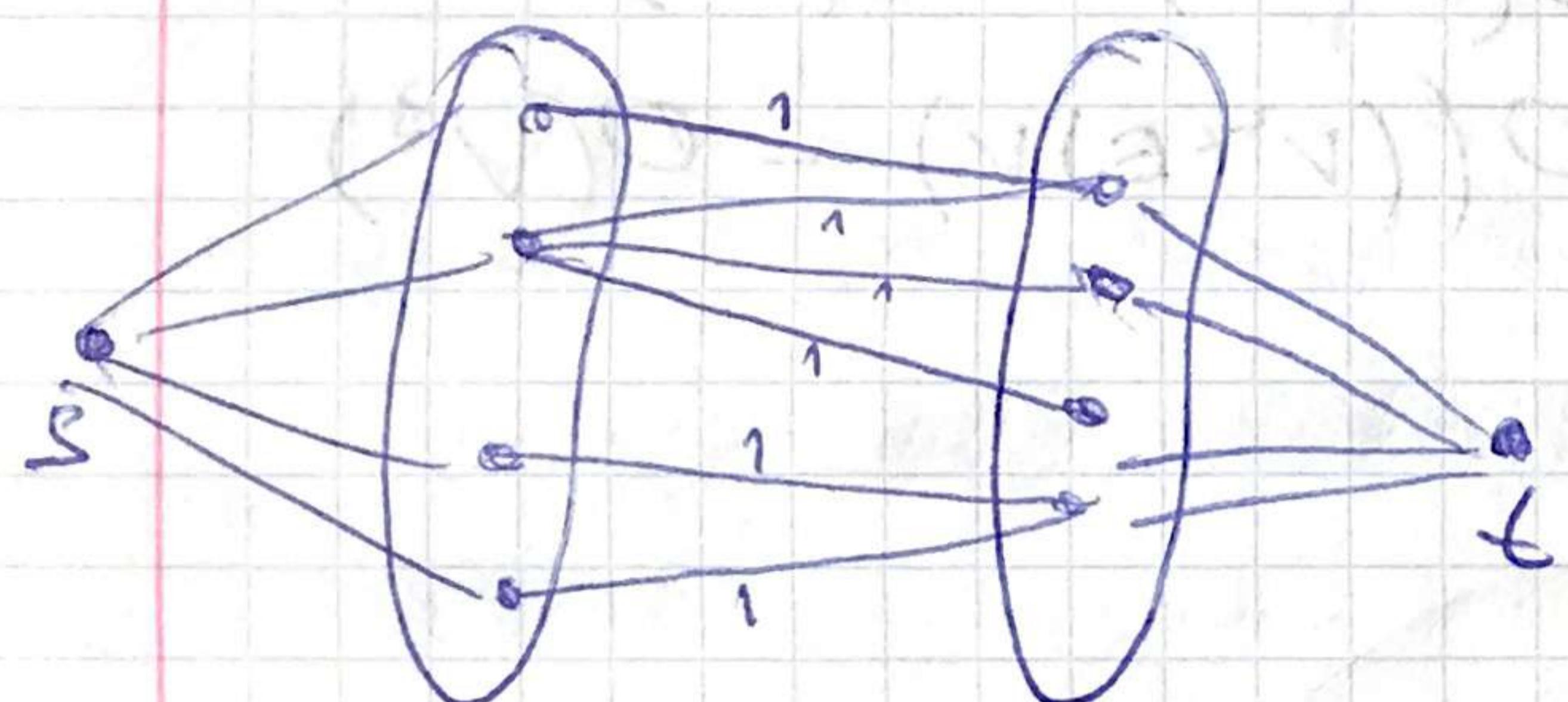


(2)



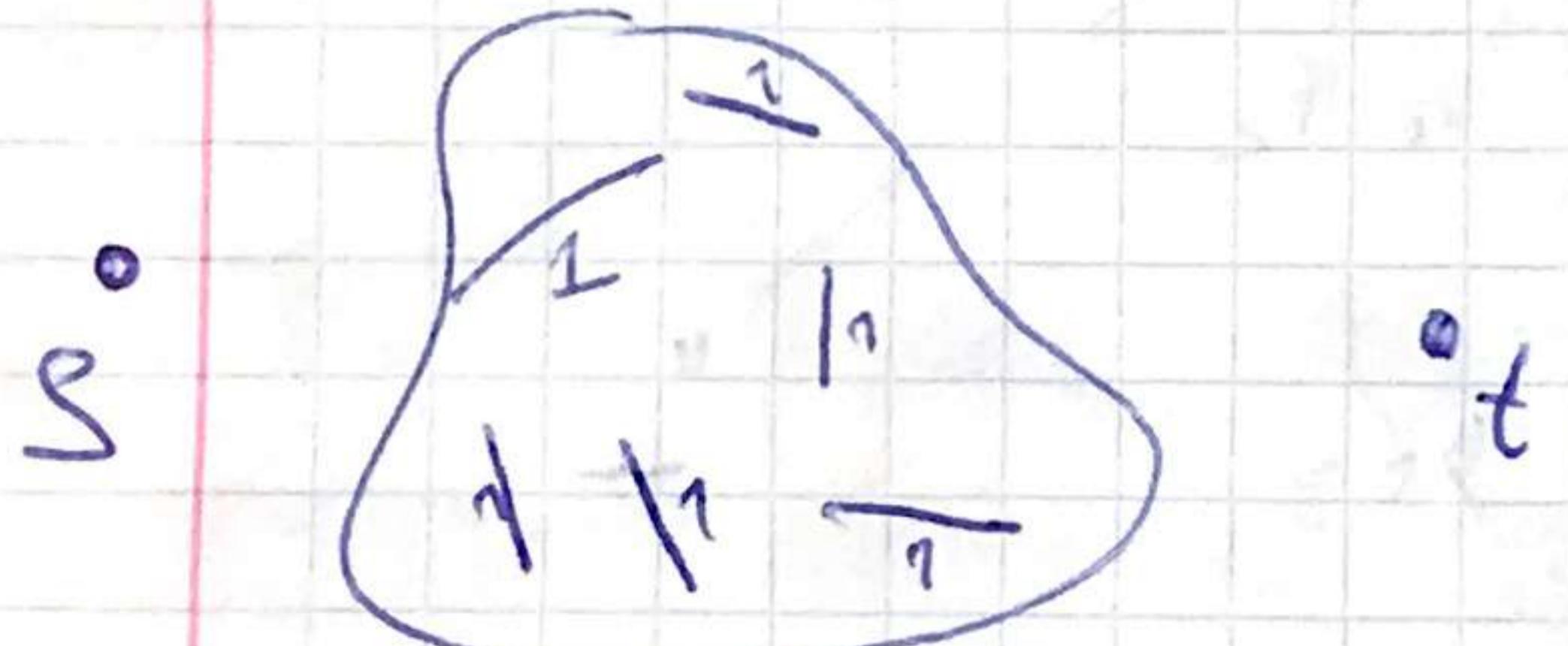
(3)

МАКСИМАЛЬНОЕ ПОРОСОГРАДИУС



(4)

Предположим, что существует путь из S в t

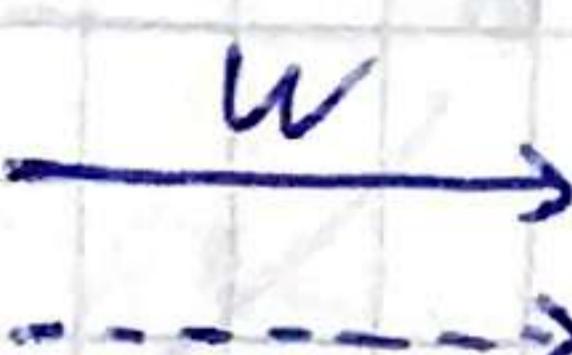
Если в подграфе  $K_{1,2}$  есть путь, то

Эйлеров - кард

Теорема: Время работы  $O(VE)$ 

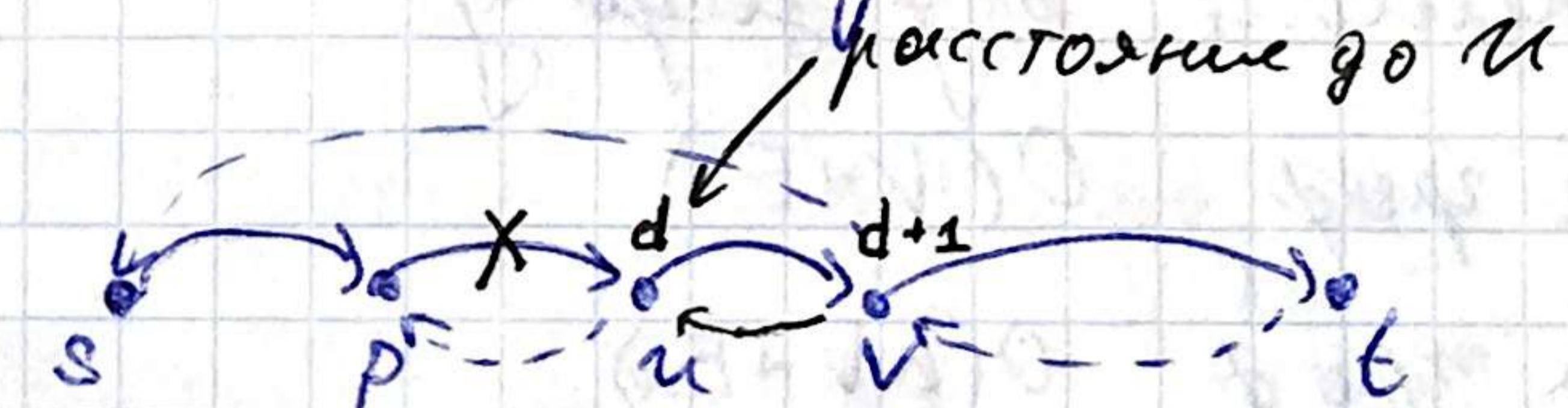
Факт: граф не имеет циклических путей

с тремя различными вершинами

Лемма: На каждой цепочке есть вер.  $\exists k$ 

K1 из S и S из K1 друг от друга не соединены

D-60:



Рассмотрим путь по этому пути

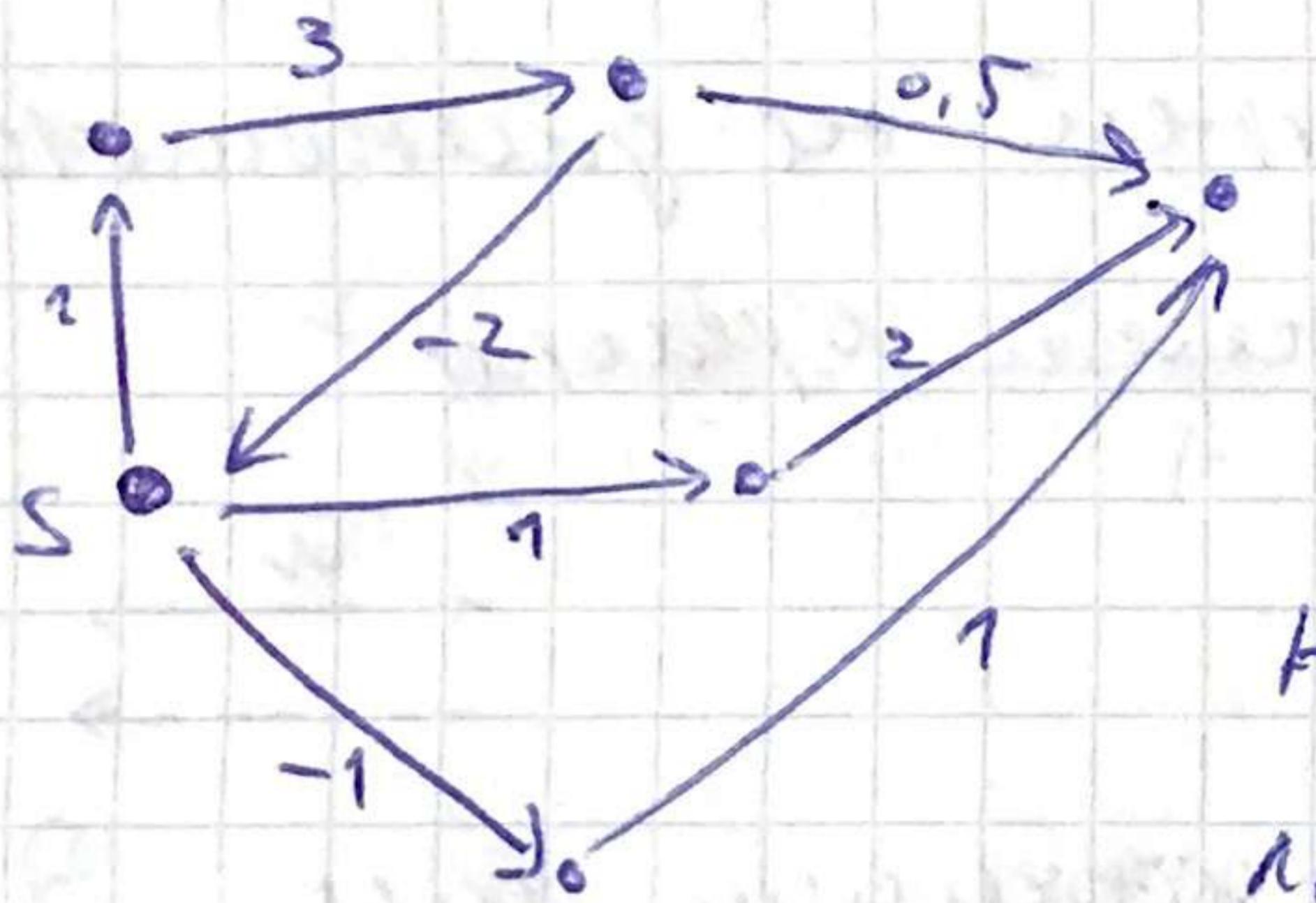
1) Доказать  $V \rightarrow u \in K1 \text{ из } S \text{ из } u$ 

$$S \rightsquigarrow V \rightarrow u \quad | \Rightarrow \beta(S, u) = d+2 > d \Rightarrow$$

 $(V, u)$  не могут быть K1

2) При удалении ребра K1 не удаляется

## Кратчайшие пути в графах



$w: E \rightarrow \mathbb{R}$  - вес  
ребра

Задача (SS)

Найти КП от  $S$  до  
любой другой вершины

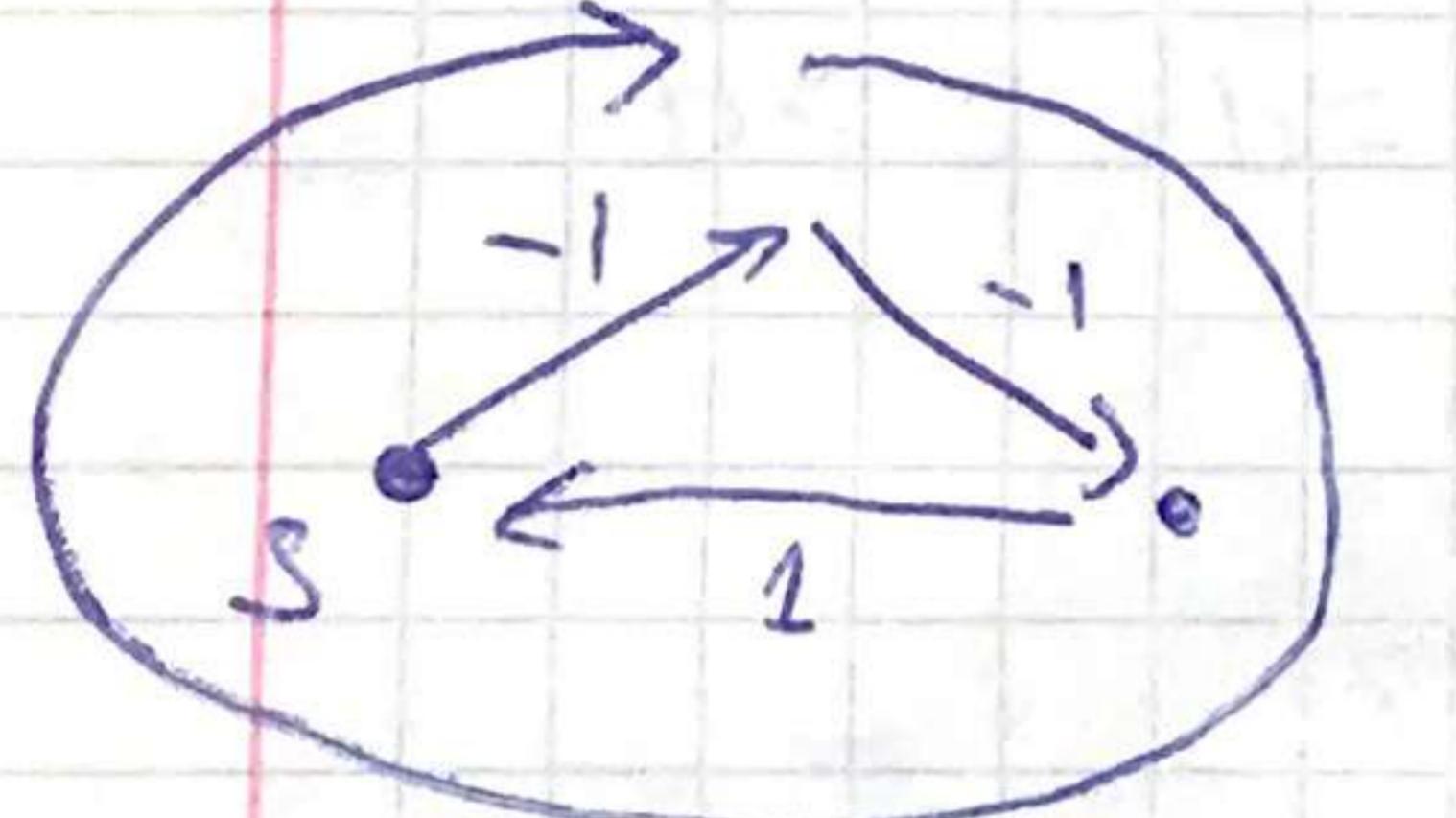
- BFS решает эту задачу

0-1 граф  $O(V+E)$

0-к граф  $O(KV+E)$

только с целочисленными весами

Lemma: (корректность итеративной задачи)  
 $O(KP)$



Задача нахождения КП корректна

В задаче есть отриц. весы.

0-60:  $\Rightarrow$  В задаче КП нет циклов

1) Если для дара отриц. весы  $\Rightarrow$

$\Rightarrow$  невозможно найти путь между двумя вершинами

$\Rightarrow$  исходная задача не имеет решения.

2) Если для дара некоторый цикл  $\Rightarrow$

$\Rightarrow$  это means узел из которых и  
которые старт для цикла



$\Rightarrow$  Донесение нет отриц. весов  $\Rightarrow$

$\Rightarrow$  любой КП из  $S$  в  $t$  не содержит

циклов [т.к. отриц. нет, а никак. не может

иметься]

$\Rightarrow$  КП однозначно простой

Безо простых путей из  $S$  в  $t$  корректен

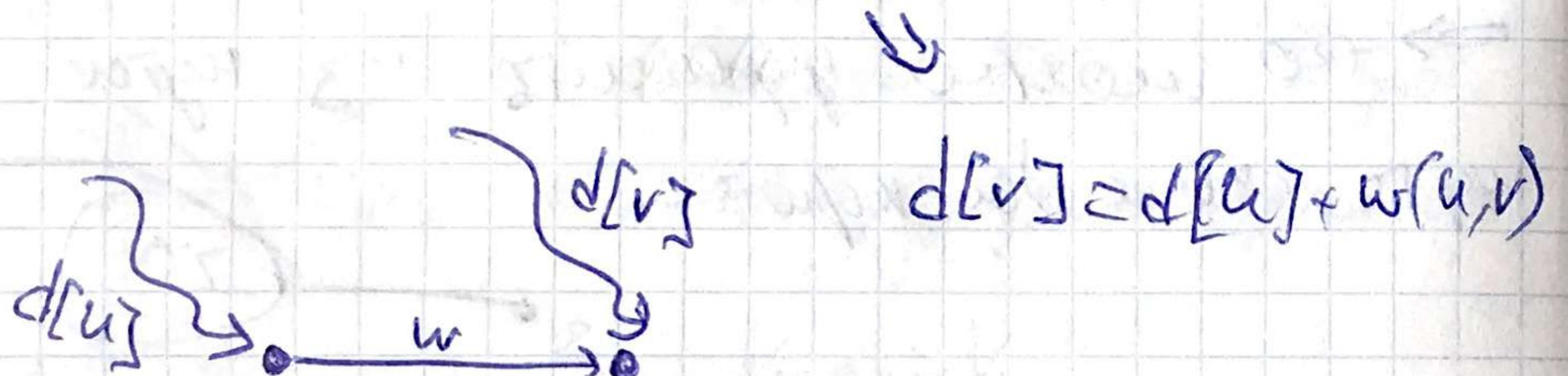
знач. В корректной задаче всегда есть мин.

$\nexists$  КП из  $S$  в  $t$

## Алгоритм Форда-Белмана

Рассматриваем пары  $(u, v)$ : Рассчитываем  $d[u]$ ,  $d[v]$  — расстояние от вершины  $u$  до вершины  $v$  по ребру.

$$\text{Если } d[u] + w(u, v) < d[v]$$



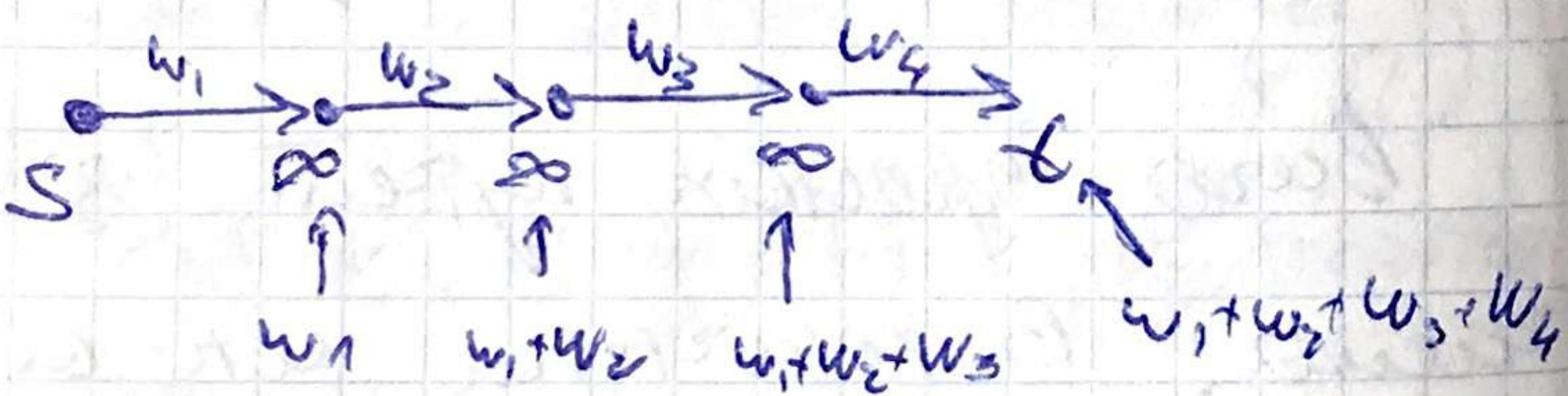
$$d[0..n-1] = \infty, d[s] = 0$$

```
for i in [0..n-1]:
    for e in E:
        Relax(e)
```

$$O(VE) \sim O(V^2)$$

$$\sim O(V^3)$$

Корректировка:



Простой цикл имеет  $\leq V-1$  ребра  $\Rightarrow$   
 $\Rightarrow$  простой цикл имеет  $\leq V-1$  ребра

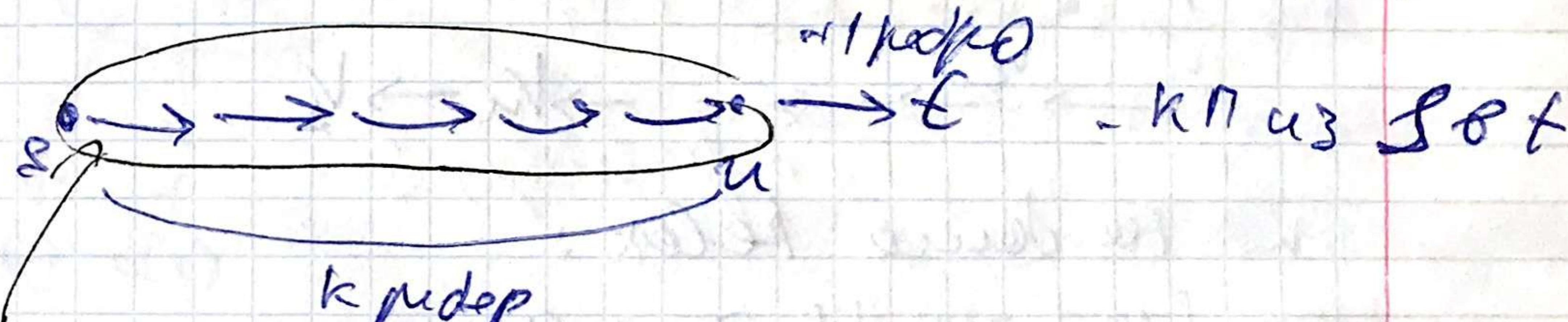
D-60  $f_k$  (Алгоритм Форда-Белмана корректирует  
находяется  $k$  за  $V-1$  шаг)

D-60 Утверждение № (доказательство  $\leq SBF$ )  
(ко-60 предп.)

Доказ.  $d[S] = 0 \rightarrow d[S] = 0$

Рассчитывается все  $KR$  в  $\leq k$  предп.

После  $\geq k$  шагов рассчитывается  $K+1$  предп.



корректирует на  $\leq K$

Так как  $d[e]$  уменьшается на  $\leq 1$  при каждом шаге  
если предп.  $e$  верно, то  $d[e] \leq K$   $\Rightarrow$  корректирует  $KR$ .

for i in [0..n-1]:

for e in E:

Relax(e)

for e in E:

if (Relax(e)):

free negative cycle!

th(Кратчайшийopp. гранич)

B G есть гранич opp. Гречес



т.к.  $V - \bar{v}$  не является границей Relax.

D.G.  $\boxed{\exists i}$  существует  $v_i$  в  $\partial V$  кратчайшее

$\boxed{\exists i}$  Доказано feet Relax.

& opp. гречес

$$v \rightarrow v_2 \rightarrow \dots \rightarrow v_n \rightarrow v_1$$

т.к. все доказано Relax:

$$d[v_2] \leq d[v_1] + w(v_1, v_2)$$

$$d[v_3] \leq d[v_2] + w(v_2, v_3)$$

+

$$d[v_1] \leq d[v_n] + w(v_n, v_1)$$

$$\sum_{i=1}^n d[v_i] \leq \sum_{i=1}^n d[v_i] + \sum_{i=1}^n w(v_i, v_{i+1})$$

$$0 \leq \sum_{i=1}^n w(v_i, v_{i+1})$$

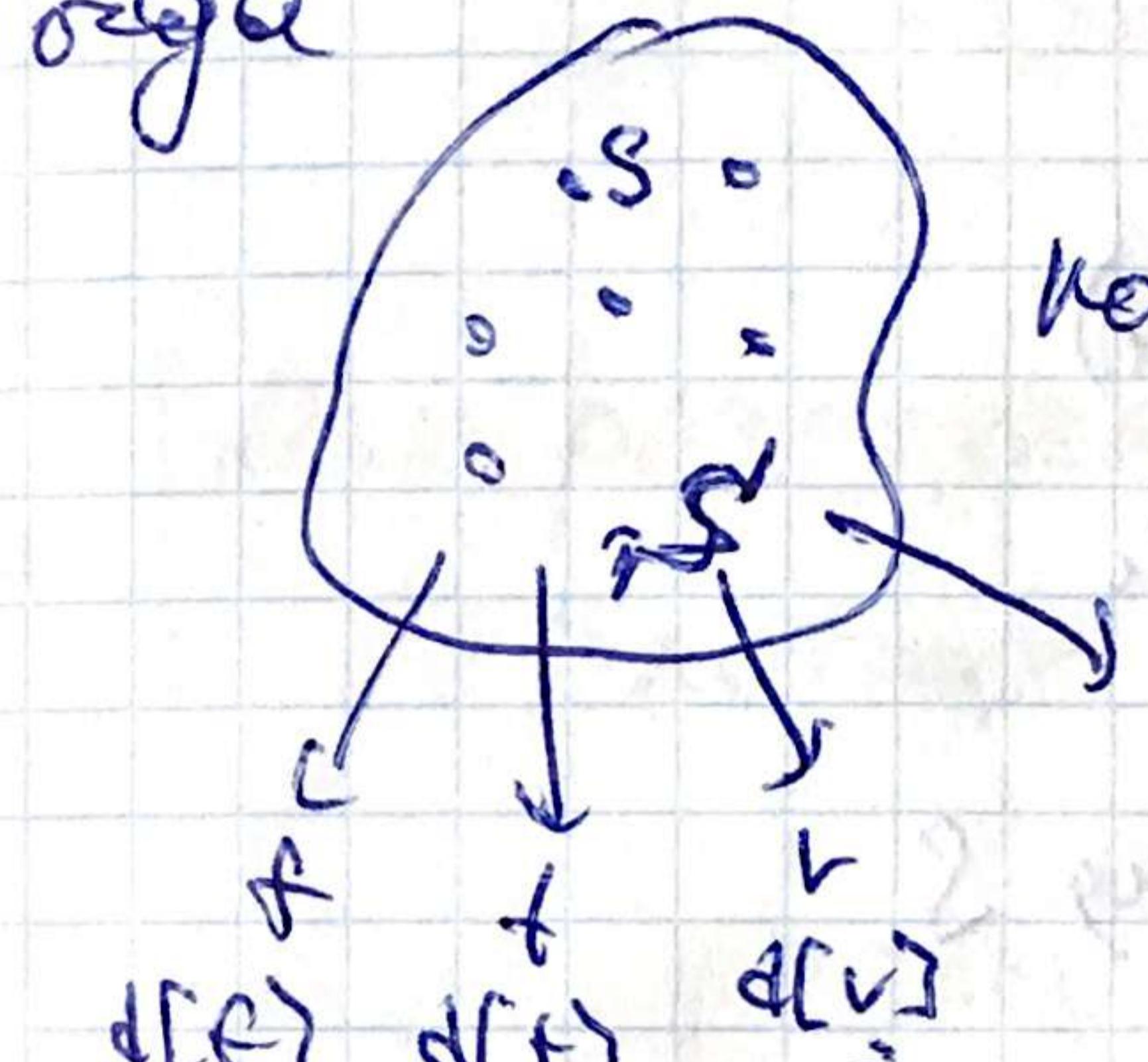
граница  
опис.  
запись

O(VES) находит  $KV$  между opp. Гречес

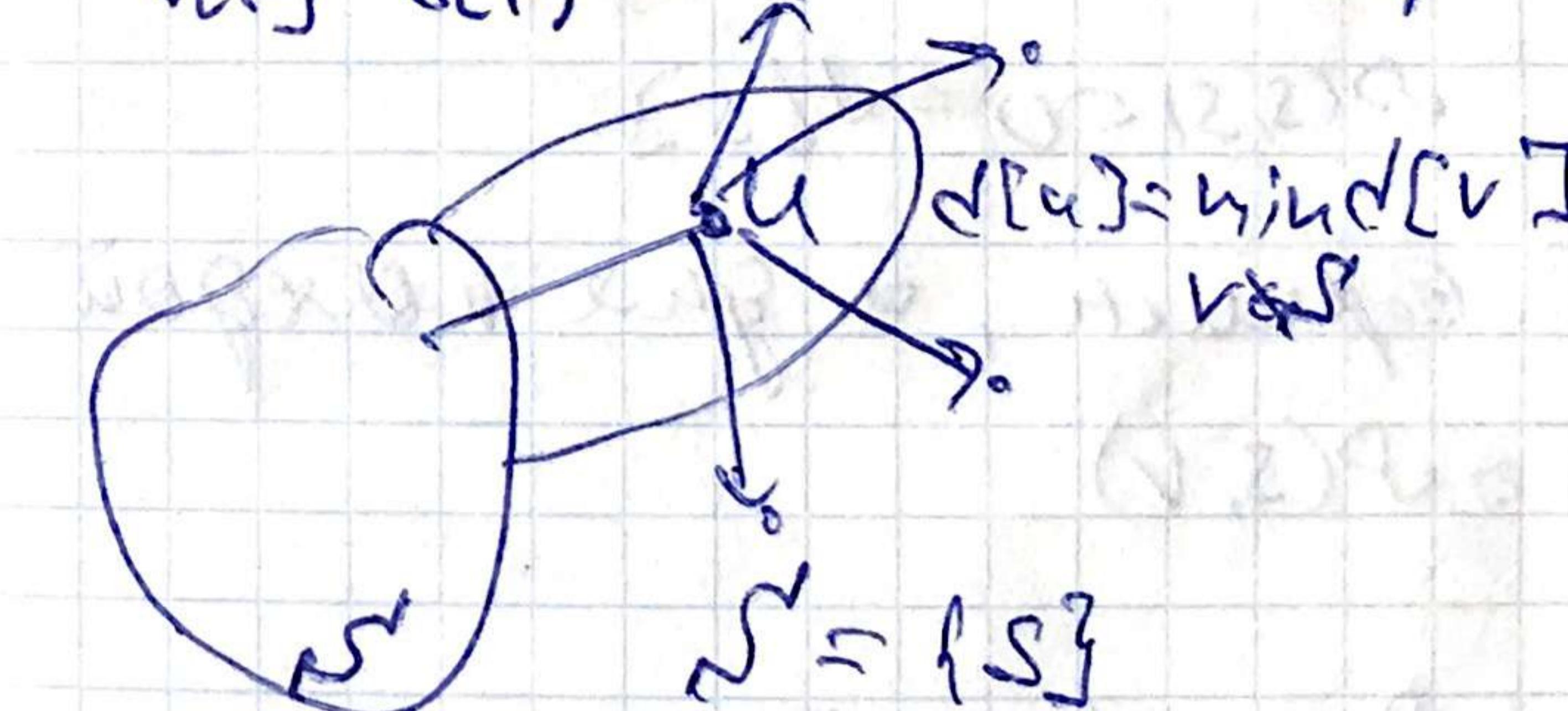
### Алгоритм Динкинга

Рядо Гречес предп  $\geq 0$ .

Tогда



$S = \emptyset$  инициализация,  $g_{\infty}$  и  
коробка избраных  $KV$  от  $S$ .



$d[u] = \min(d[v])$   
 $v \notin S$   
ищется  $v$  с  $d[v]$

если  $v \in S$   
занести

$O(E \log E)$

$O(E \log E)$

$O(E \log E)$

for  $i \in 0 \dots V$ :

ищется  $v \in S$

$O(V)$

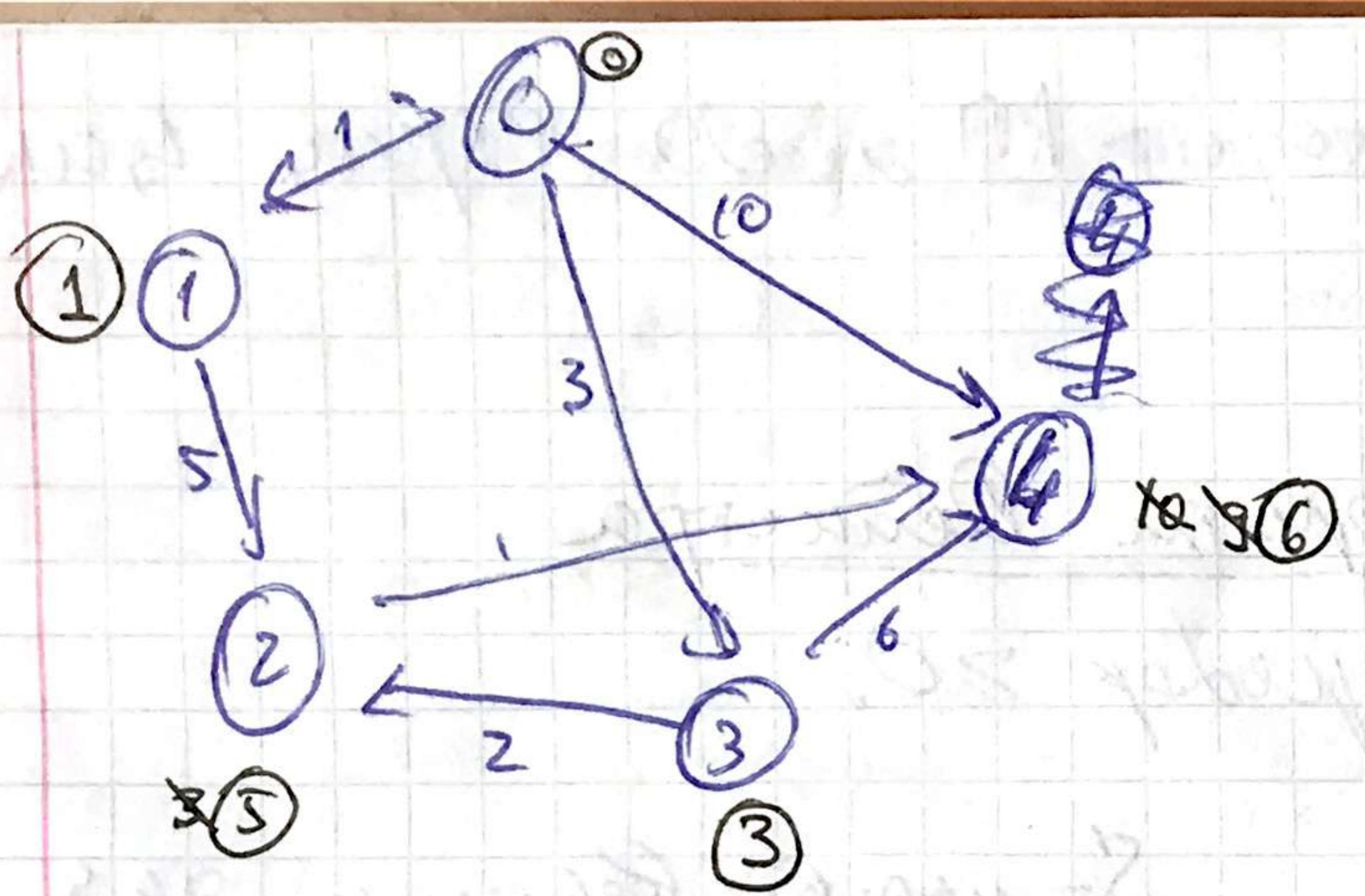
• Выбрать  $v \in S$ :  $d[v] \min$

• Выбрать  $v \in S$

• for  $(v, u) \in E$ :

$O(F)$

Relax( $v, u$ )



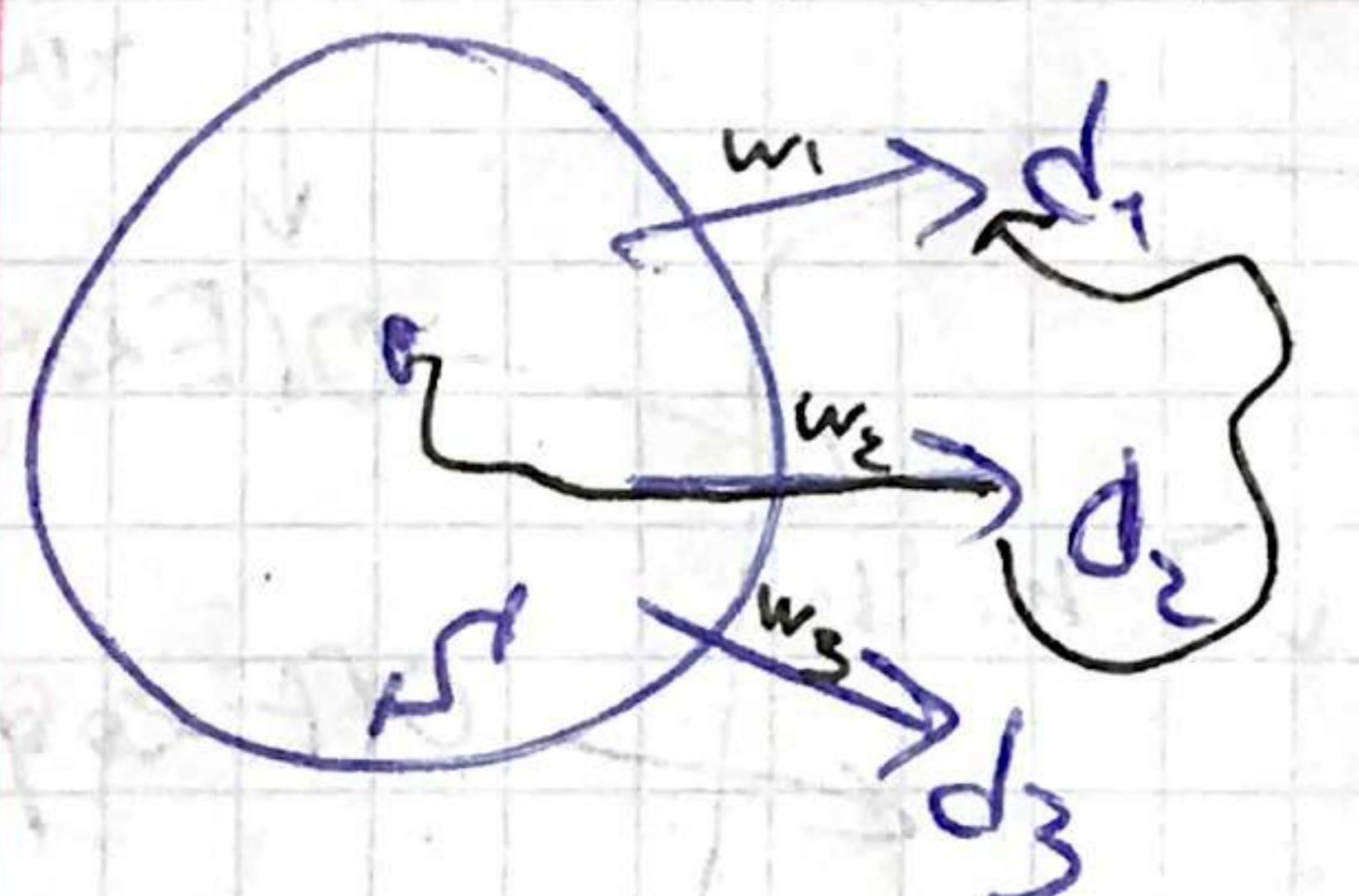
$$\text{th: } \forall u \quad d[u] = \mathcal{P}(S, u)$$

наше алг. делает

Доказ.: ищем  $v \in S$

$$\text{тогда: } S \cup \{v\} \quad \mathcal{P}(S, S \cup \{v\}) = d[S]$$

Найдем  $v \in S$  к которому, а где находят  
указав  $d[v] = \mathcal{P}(S, v)$



Найдем  $d_i - min$   
Получим, что это  
нужно копировать из  $d_1$

$$\text{Значит, } d[u] < d[v]$$

Так как предо  $w_2 \in k\pi$  от  $S \ni u \Rightarrow$   
 $\Rightarrow d[v] - k\pi$  от  $S \ni v$

$$\mathcal{P}(S, u) = dV + p(v, u) \geq dV \cancel{\geq} \text{ н.о.}$$

$\geq 0$

количество разности  
меньшее

$$d[u] = \mathcal{P}(S, u)$$

$$dV > du$$

ищем первое  
н.о., которое  
принадлежит.

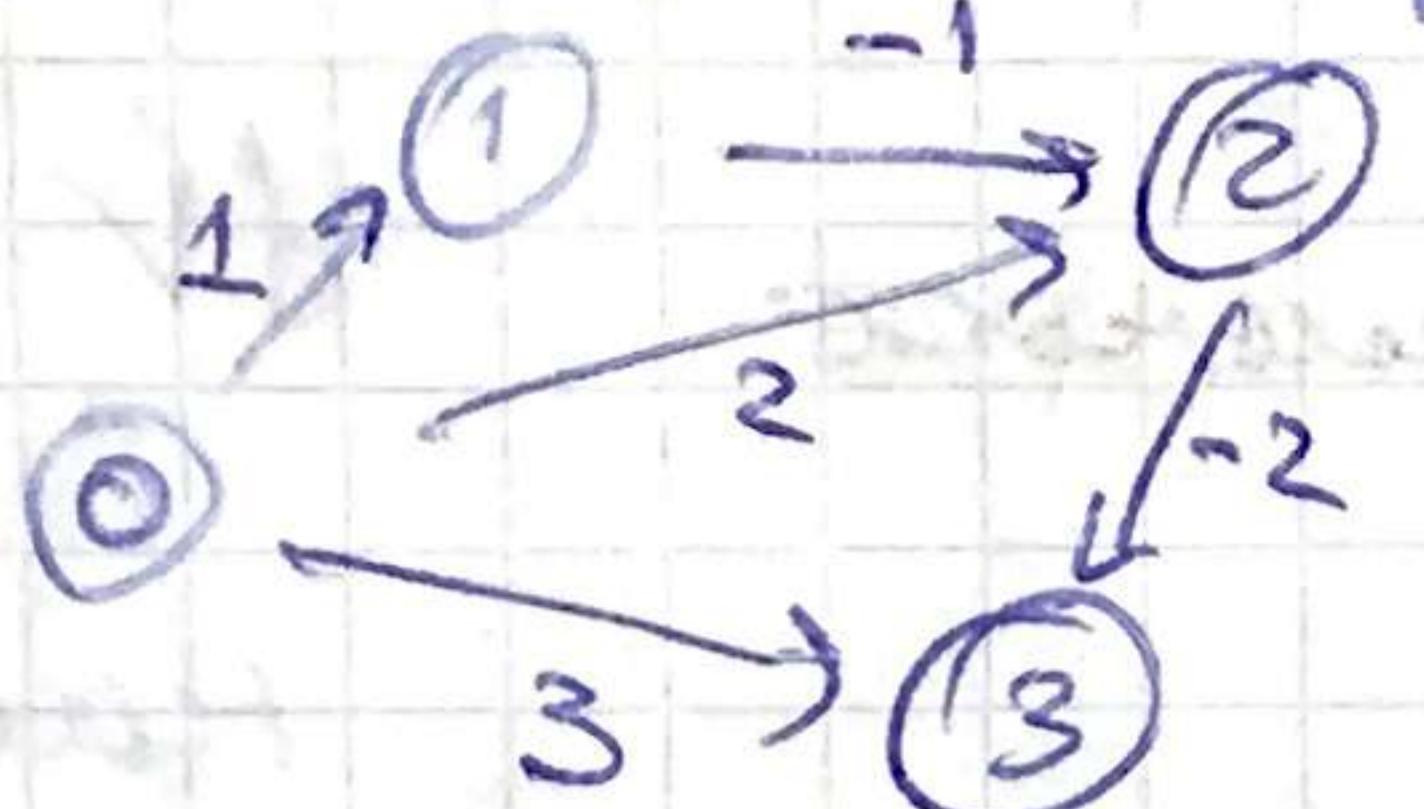
когда	Для разрешимых задач $\Theta(V \log V)$ Плохих
-------	---

если	Для разрешимых задач $O(V^2)$ для $V$ Плохих
------	---

$$O(V^2) \text{ для } V^2$$

31.10.20

## СЕМИНАР (Single Source)

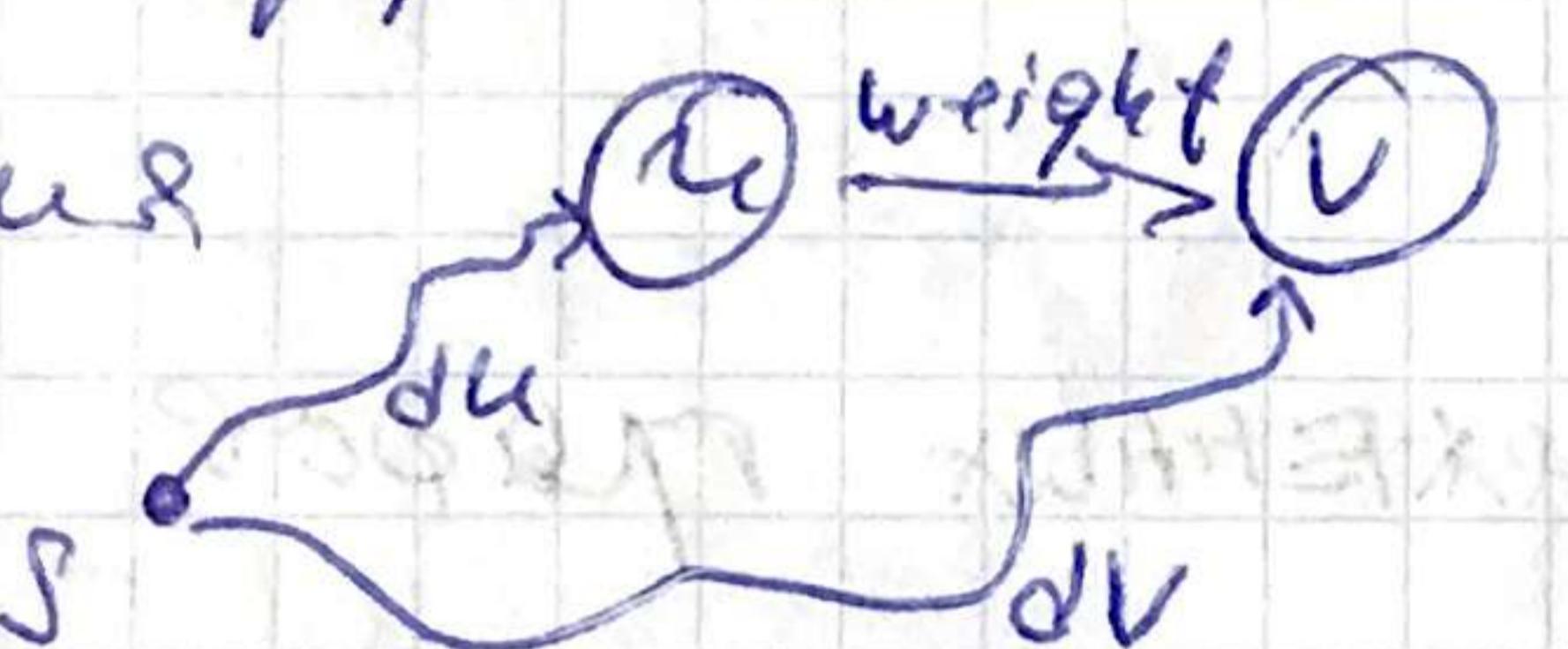


$$w: E \rightarrow \mathbb{R}$$

Начинаем с [самого] вершины  
самого

### Алгоритм Поппа - Денкмана

Решение



$$d_V = \min(d_U, d_W + w)$$

$$d[0..n-1] = \infty$$

$d[S] = 0$   
for ( $i \in \{0\} \cup \{n-1\}$ )  
for  $e \in E$ :

Relax( $e$ )

Relax( $e$ ):

if  $d[e.to] \geq d[e.from] + e.w$   
 $d[e.to] = d[e.from] + w$   
 $p[e.to] = e.from$

если огрызая

for  $e \in E$ :

if (Relax( $e$ )):  $\rightarrow$  если огрыз. узел в K & S

return  $\exists e \in E$  огрыз. узел

return

нет огрыз. узел

std::vector<vertex\_t> GetSP(const Graph &g, vertex\_t s)

const int inf = 1000000000;

vector<int> min\_distance(g.s, zero);

vector<vertex\_t> prev(g.s, zero);

min\_distance[s] = 0

for (~~int i = 0~~; i < g.s.size(); ++i) {

for (const auto &neighbor : g) {

for (auto vertex : neighbor) {

Relax( $v$ , min\_dist, prev)

g

return prev

const Graph &g,

void Relax(~~edge~~ &edge, ~~vertex~~ vertex, vector<int> &min\_dist,

min\_dist[edge.from] != inf vector<vertex\_t> prev) {

if  $\& min\_dist[\text{vertex}] > min\_dist[\text{edge}.from] +$

+ edge.weight {

min\_dist[edge.to] = min\_dist[edge.from] + edge.weight

prev[edge.to] = edge.from

упорядочен



$O(VE)$

$w \geq 0$

## Авторем Дейкстры (Решение)

$S' = \{ \text{текущий узел} \} - \text{однодоминантные узлы}$

$d[u, v-1] = \infty$

$d[S] = 0$

for i from 0 to V-1:

$O(\log E) \rightarrow \text{FindMin } d[u] : u \in S'$

$S'.add(u)$

[for  $(u, v) \in E$ :

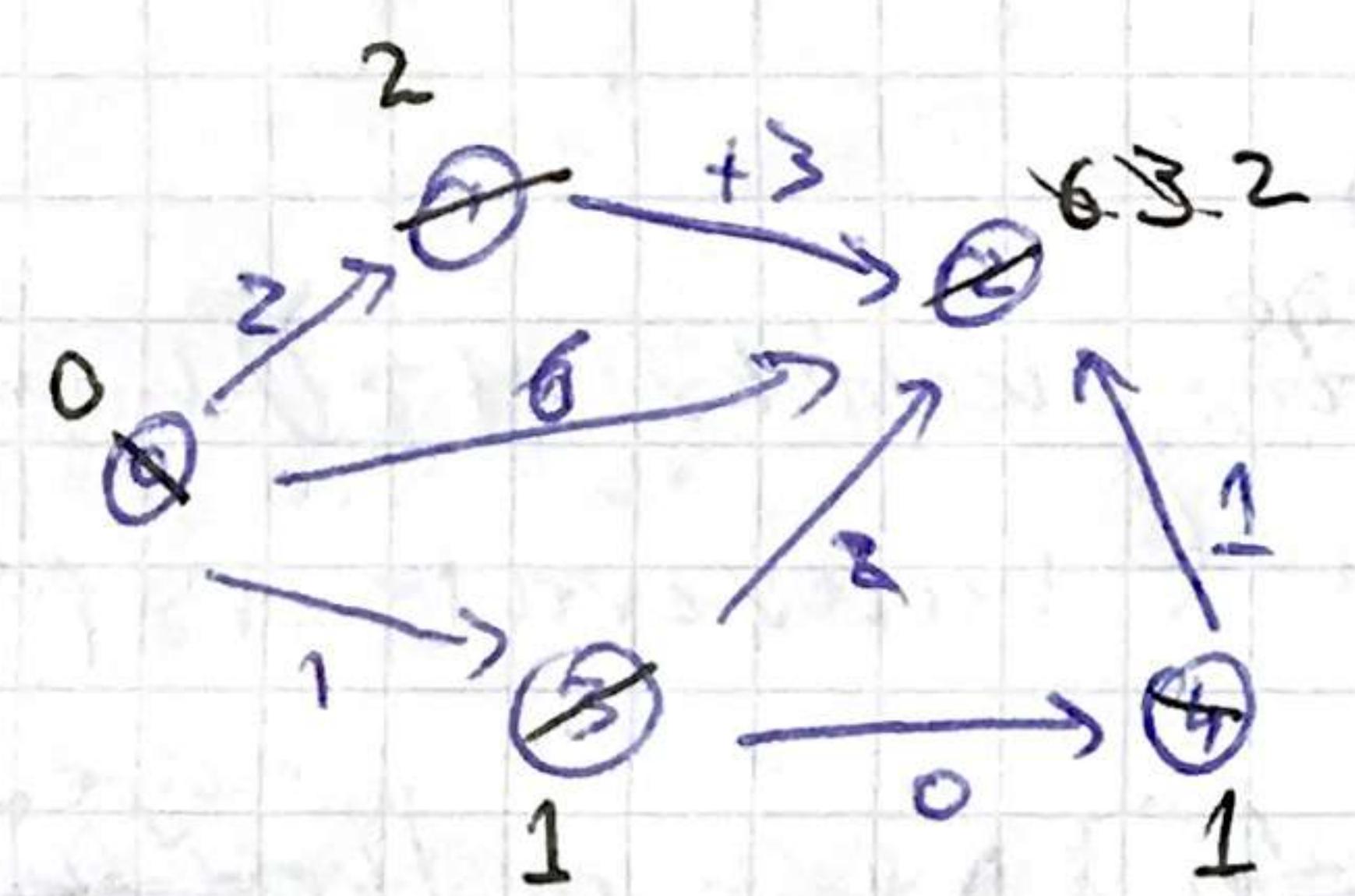
Relax( $u, v$ )

$\rightarrow \text{Relax}(e)$ :

if  $d[e, v_0] > \dots$

...  
heap.push( $d[e_0]$ )

$O(\log E \cdot \deg u)$



$O(V \log E + E \log E) =$

$= O((V \cdot E) \log E)$

КУ4А

Для параллельной работы:  $V \log V - 10/10$

Для квадрата

небольшой

Для параллельной

работы

:  $V^2$

:  $V^2$

:  $V^2$

## СЕМИНАР

E~V | E~V | 7.11.20

Popg - Баннер

Дейкстра

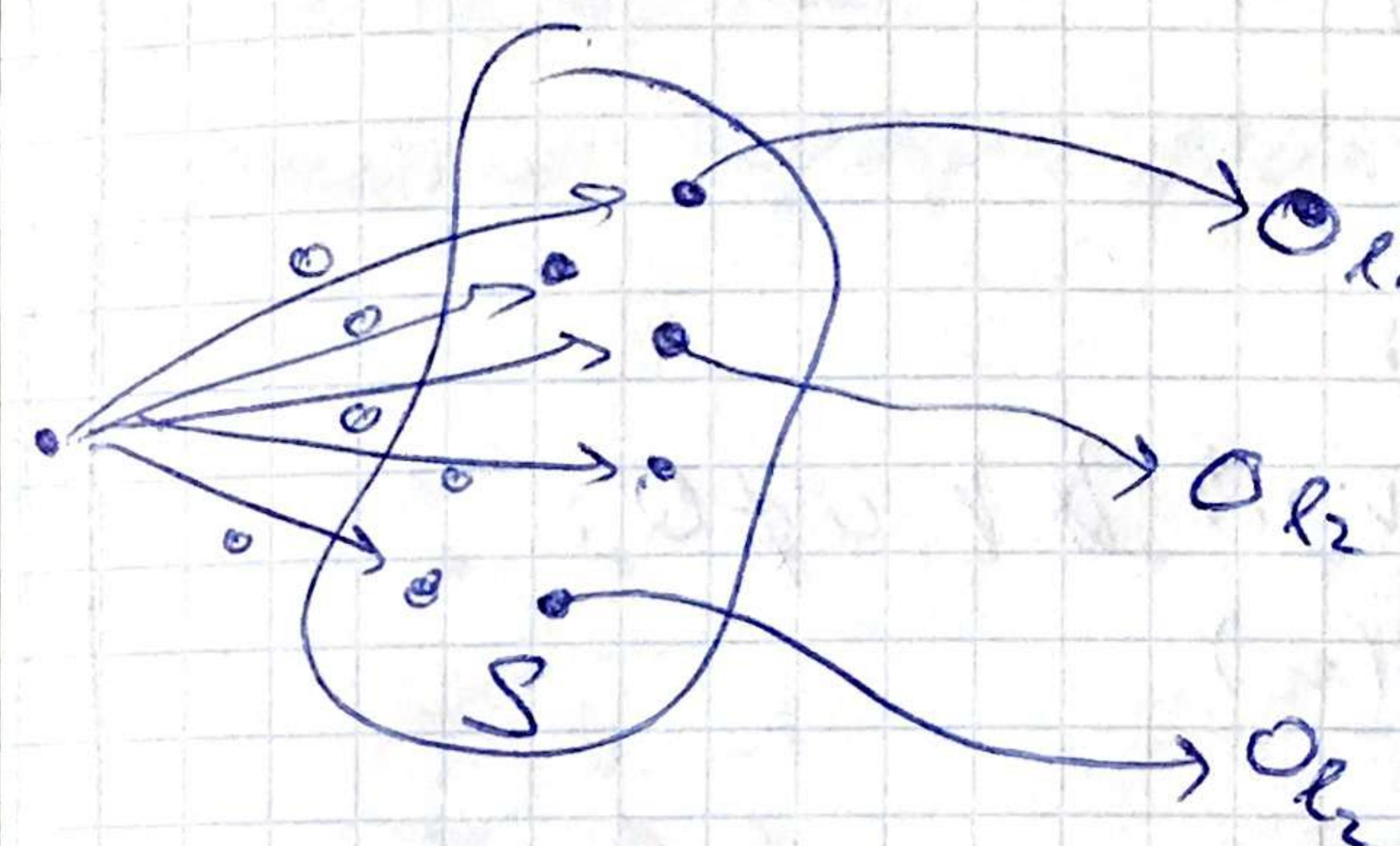
$\rightarrow \text{Heap}(E \log V) | V \cdot \log V | V^2 \log V$

$\rightarrow \text{Dinic}(V+E) | V^2 | V^2$

$d[u, v-1] = \infty$

$d[S] = 0$

$\rho(S, l_i) \rightarrow \min$



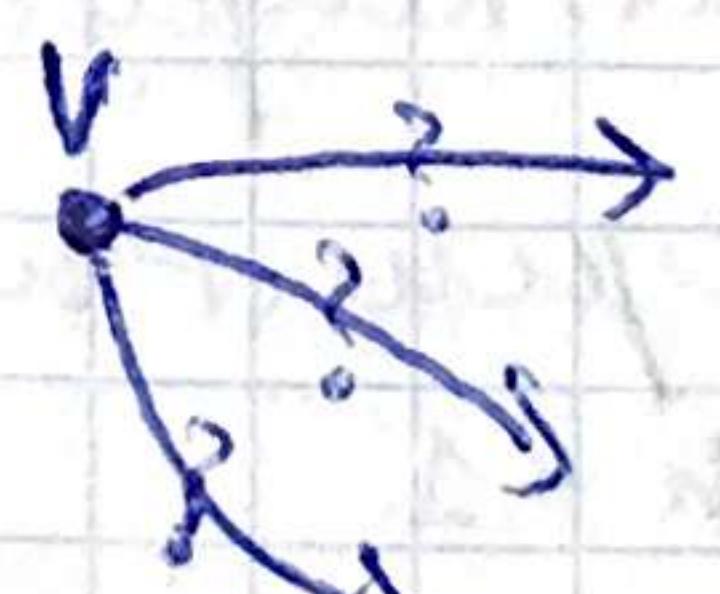
## Гарантии Popg Баннера

for i from 0 to V-1:

for  $e \in E$ :

relax( $e$ )

- 1) Если на узле было не огорожено Relax, то Break
- 2) SPF (shortest path faster algorithm)

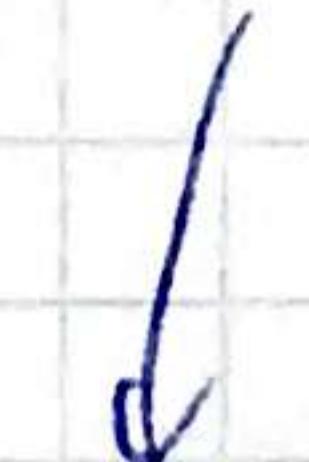


если  $\rho_{S0} V$  не изменилось, то на

$d[u, v-1] = \infty$

$d[S] = 0$

$Q = \{S\}$



$$d[0..n-1] = \infty$$

$$d[S] = 0$$

$$Q = \{S\}$$

while (!Q.empty())

$$v = Q.pop()$$

for (v, u) ∈ E:

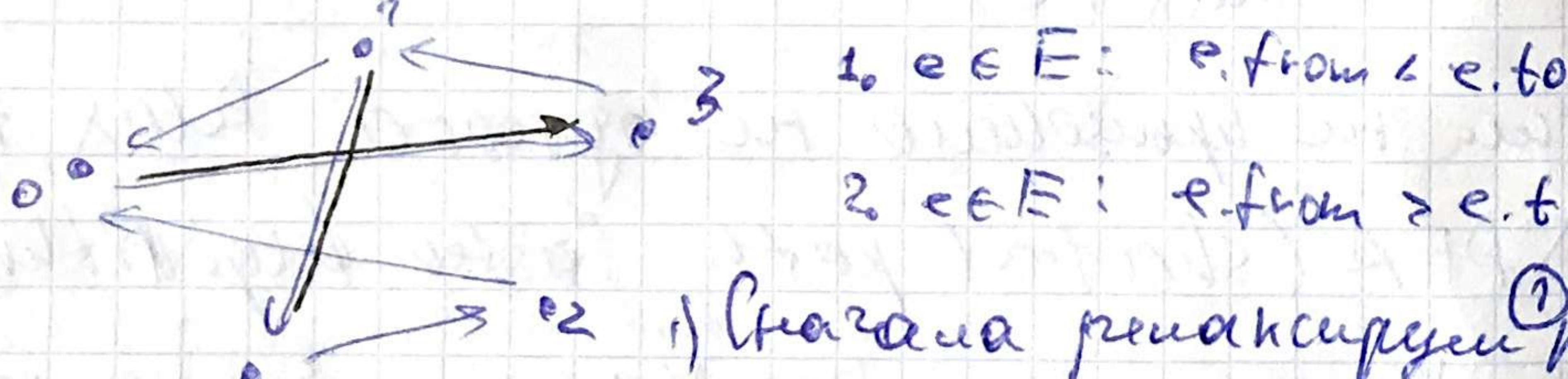
if relax(v, u) & u ∉ Q:

Q.push(u)

$O(VE)$  - в худшем случае

$O(E)$  - в среднем по неотрицательной графу

3) Yen's improvement // ускорение в 2 раза

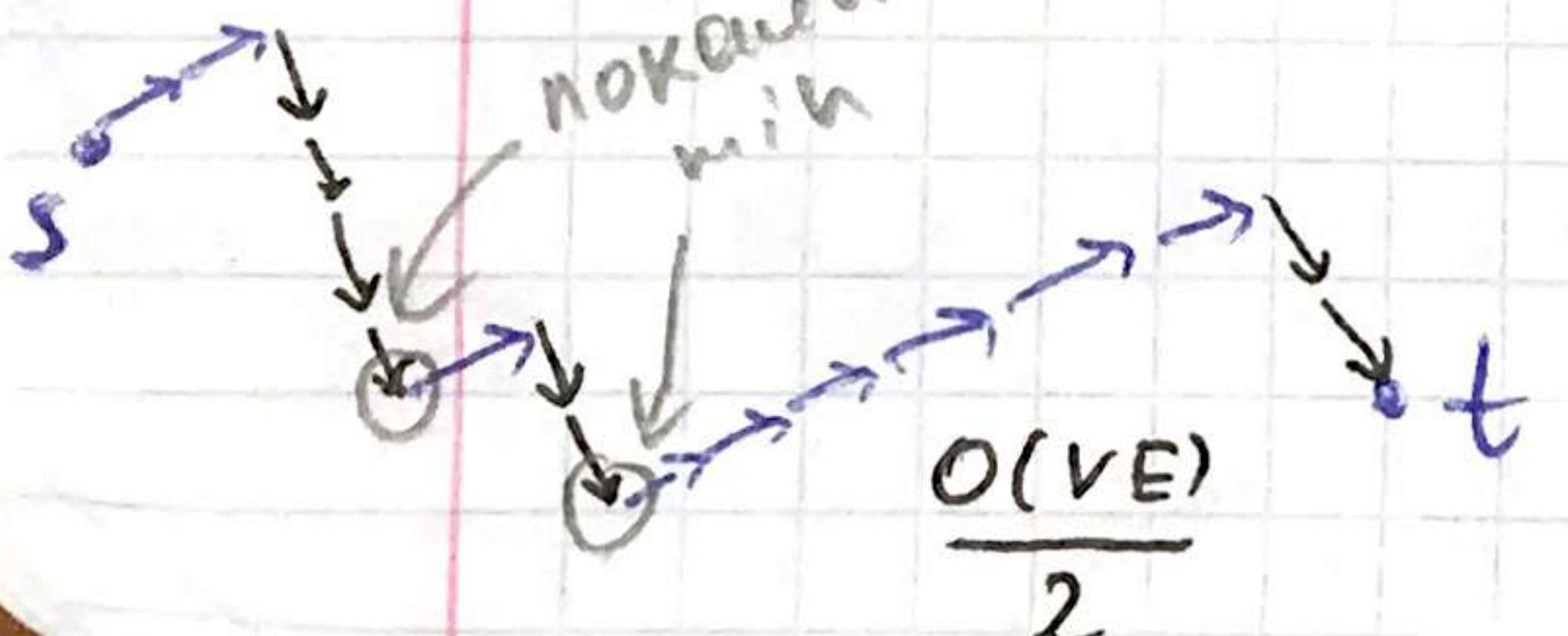


1.  $e \in E$ :  $e$ .from <  $e$ .to

2.  $e \in E$ :  $e$ .from ≥  $e$ .to

- 1) Сначала решаем задачу ①
- 2) порядок от вершины 0 до вершины  $n-1$

2) ② задача от ~~0~~  $n-1$  до 0

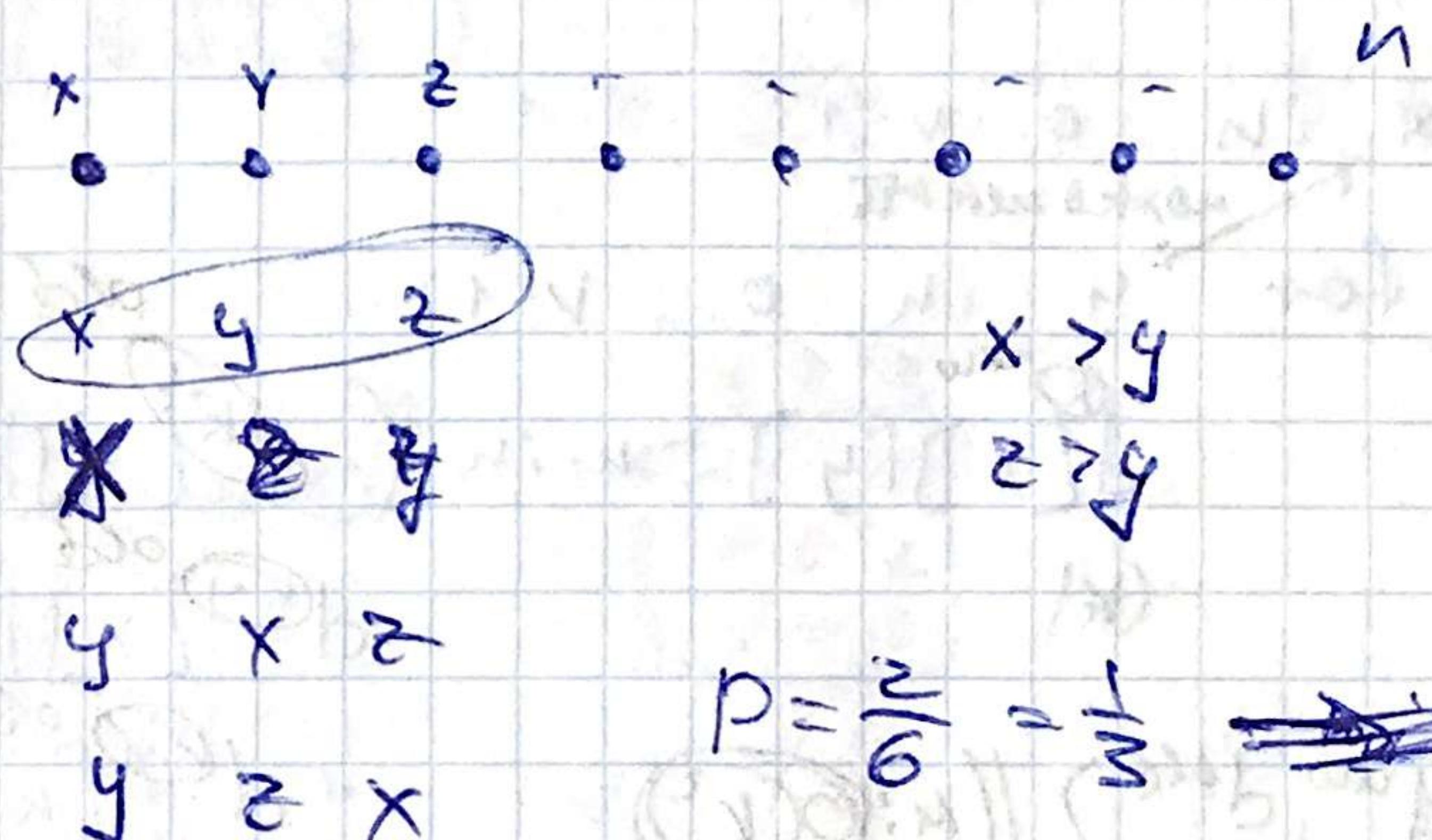


$$\frac{O(VE)}{2}$$

4) Cormenster-Eppstein

на основе алгоритма сегментации и рекурсии  
Brelinski + Yen's improvement

под  $O(VE)$ , это even & odd вершины для подграфов  
сегментации порядок редк., то мы ускорим в 2 раза  
вывод



$$P = \frac{n}{6} = \frac{1}{3} \Rightarrow \frac{4}{3} \text{ или } \frac{4}{3} \text{ нок. мин}$$

В среднем находим  $\lceil \frac{V}{3} \rceil$  сегментов

$$\frac{O(VE)}{3}$$

## All Pairs

- 1) Using union of max & min Heuristic  
min.  $\Theta(\text{V}E \log E)$ ;  $\Theta(V(V^2+E))$  - Dijkstra  
 $\Theta(V^2E)$  - Bellman-Ford

2) All-pairs Shortest T:  $\Theta(V^3)$  M:  $\Theta(V^3)$

for  $k$  in  $0..V-1$ :

    for  $x$  in  $0..V-1$ :

        for  $y$  in  $0..V-1$ : no extra memory

$d[x][y] = \min(d[x][y]^{\text{old}}$

$(*)$   $d[x][y]^{\text{new}} = \min(d[x][y]^{\text{old}},$

$d[x][k]^{\text{old}} +$

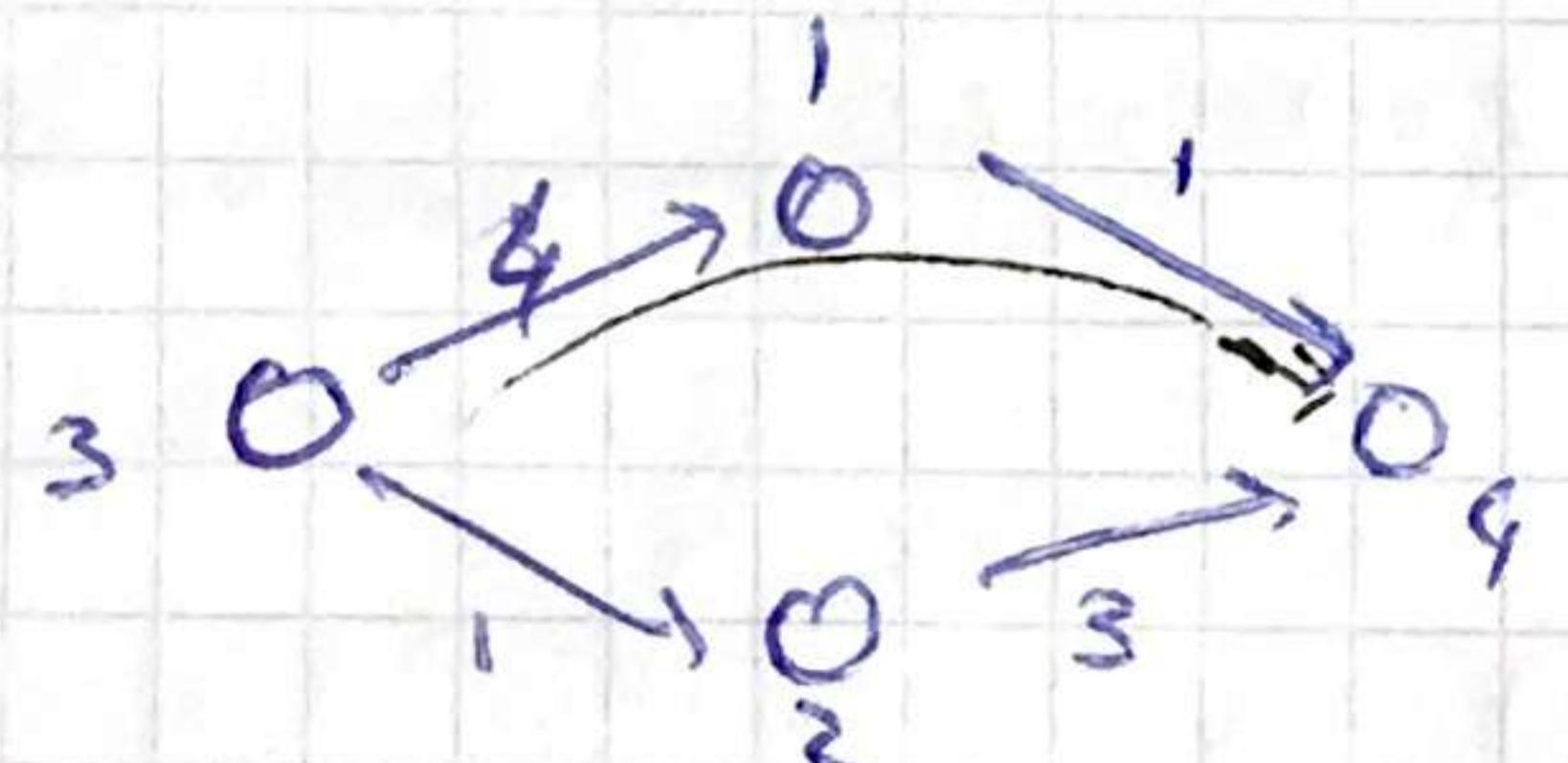
$+ d[k][y]^{\text{old}}$ )

            swap( $d^{\text{new}}, d^{\text{old}}$ ) //  $\Theta(V^2)$

$d[k][x][y] =$

$O(1)$

$k \in \{x, y\}$  or  $x \neq y$ : on shortest paths to  
    length  $\leq k$ .



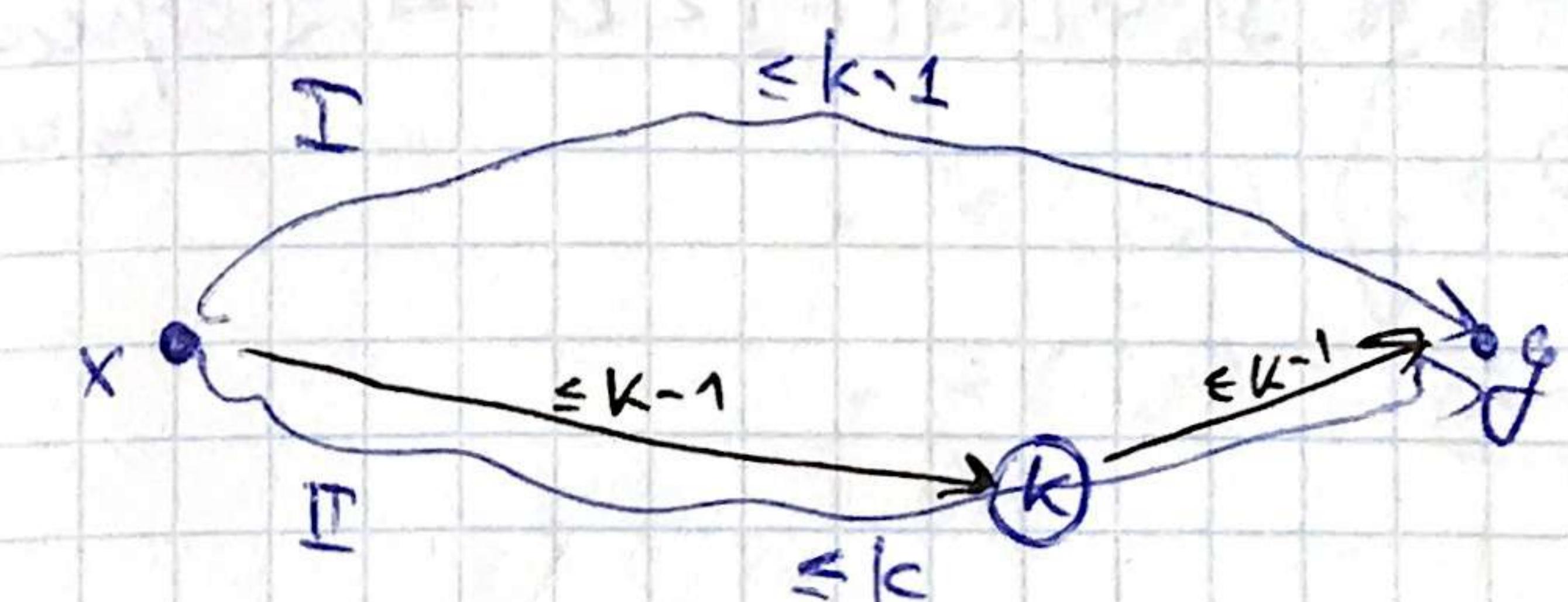
$$d^1[3][4] = 5$$

$$d^0[2][4] = 3$$

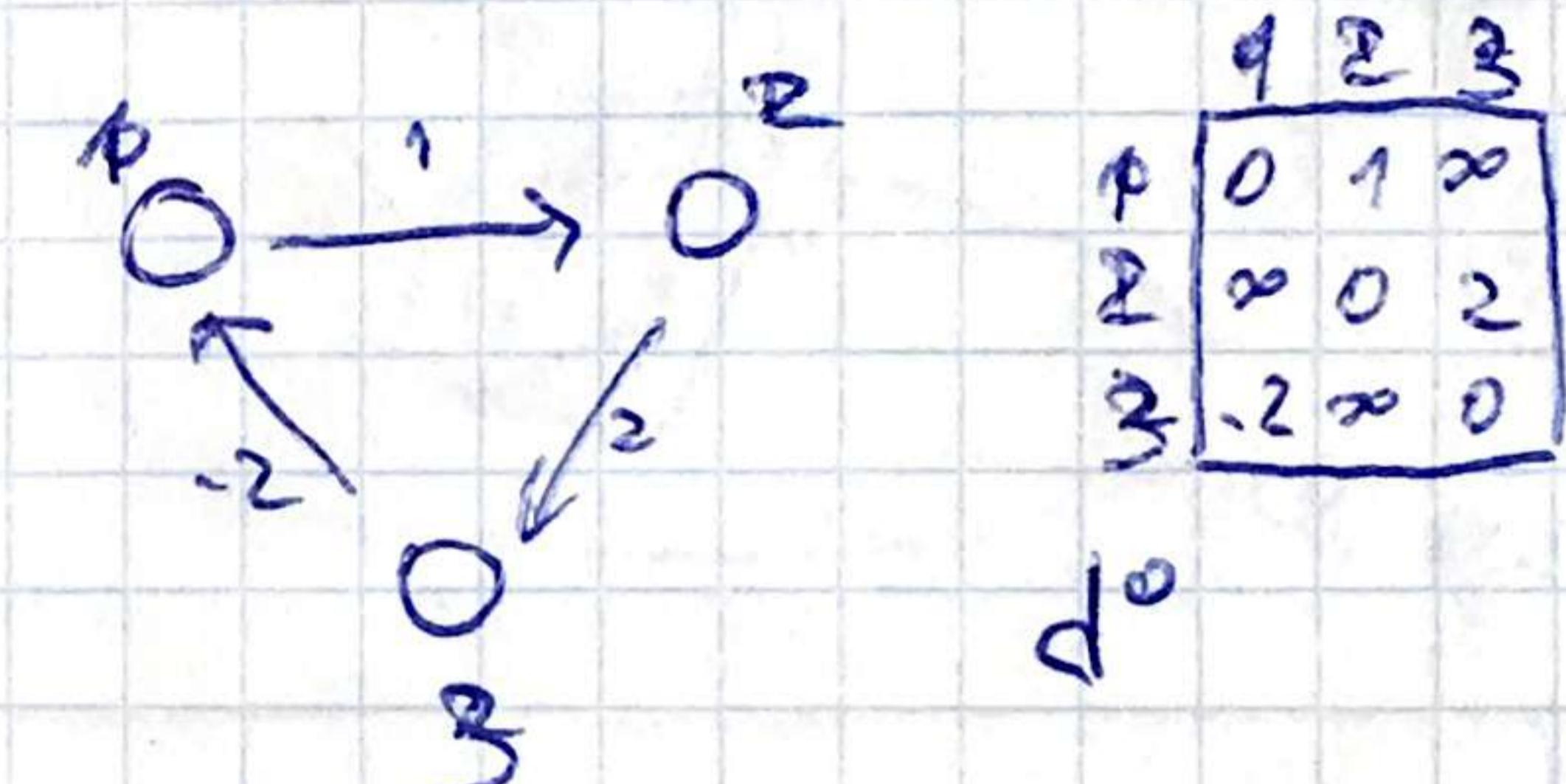
$$d^2[3][4] = 4$$

$$d^3[3][4] = \infty$$

$d^0 = W$  - max-pair between nodes



$$d^k[x][y] = \min \begin{cases} \text{I: } d^{k-1}[x][y] \\ \text{II: } d^{k-1}[x][k] + d^{k-1}[k][y] \end{cases}$$

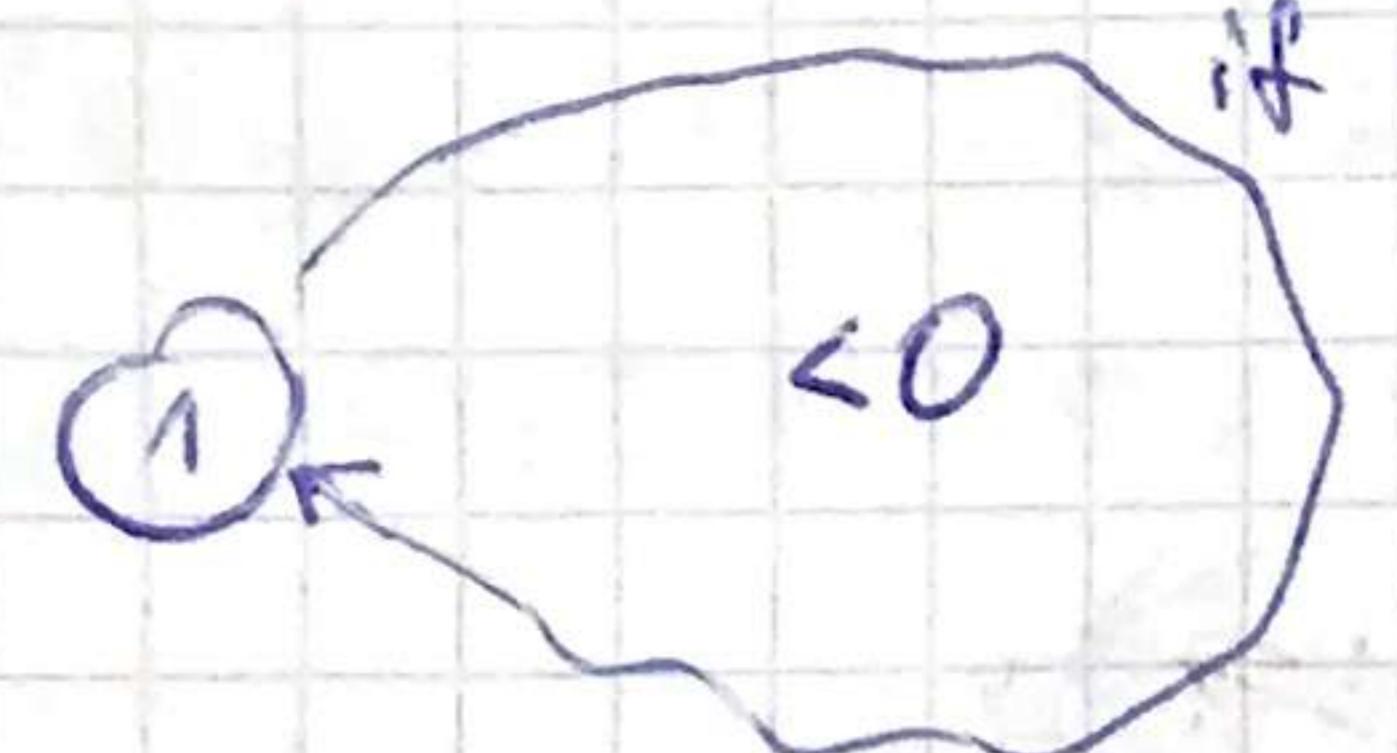


$$d^0 = \begin{matrix} \infty & 0 & 0 \\ 0 & \infty & 2 \\ 0 & 2 & \infty \end{matrix}$$

$$d^1 = \begin{matrix} \infty & 0 & 3 \\ 0 & \infty & 2 \\ 0 & 2 & \infty \end{matrix}$$

$$d^2 = \begin{matrix} \infty & 0 & 3 \\ 0 & \infty & 2 \\ 0 & 2 & \infty \end{matrix}$$

Ограничение на веса:



if  $\exists i: d[i][i] < 0 \Rightarrow \text{Forbes. value}$

Блокчейнские алгоритмы

$P[s][v] = \text{предок } V \text{ на узле } s \text{ и } v$

$P[1][3]$

$P[1][P[1][3]]$

(\*) if Relax:

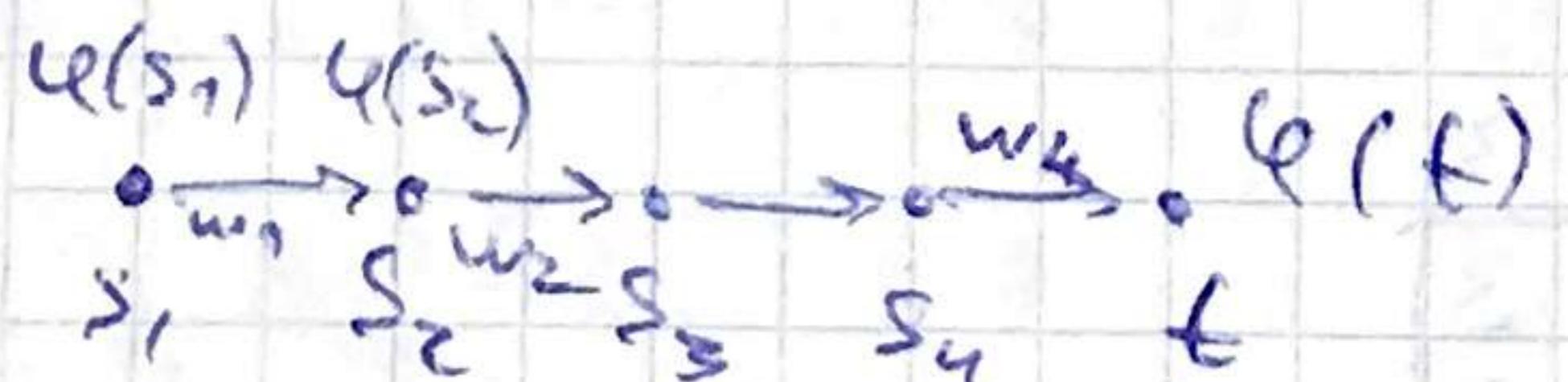
$$P[x][y] = P[k][y]$$

$V^2 \log V$  Динксп. разрх.

находит only с текущим предком

3) Алгоритм Охотника:

(разрекурсивен, и есть предок  $< 0$ )

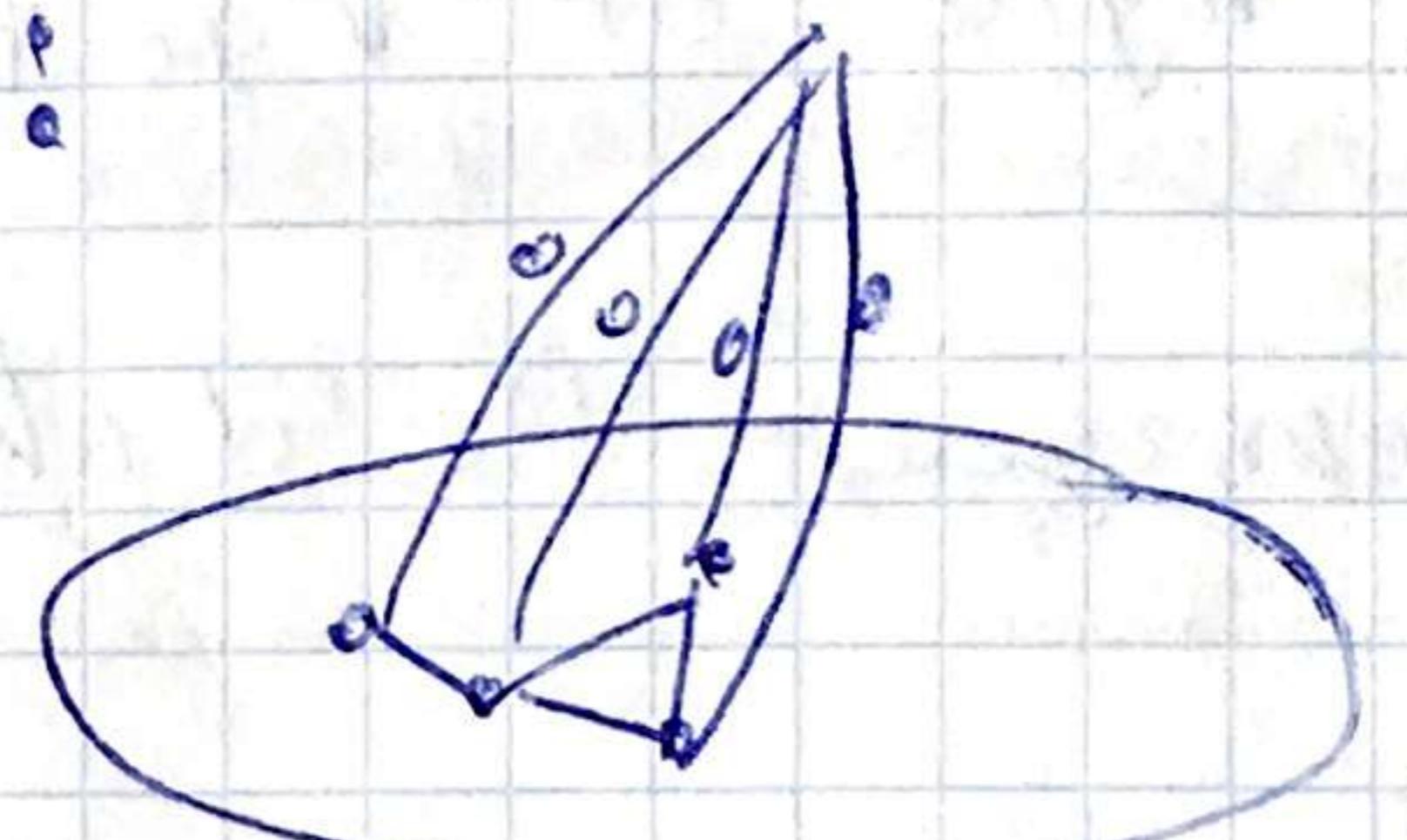


$$l: V \rightarrow \mathbb{R} \quad W'_{uv} = W_{uv} + \ell(u) - \ell(v)$$

$$\begin{aligned} w_1 + w_2 + w_3 + w_4 &+ \ell(s_1) - \ell(s_2) \\ &+ \ell(s_2) - \ell(s_3) \\ &+ \ell(s_3) - \ell(s_4) \\ &+ \ell(s_4) - \ell(t) \end{aligned}$$

All Pair  $V^2 \log V$

l:



$$\ell(v) := P(s^*, v)$$

$O(V^2) - \text{алгоритм}$

$\ell$  генерирует рекордное значение (если неограничен)

Алгоритм Охотника

1) Запускаем ФБ для каждого  $e O(VE)$

//  $\ell$  генерирует предок рекорд  $W'_{uv} = W_{uv} + \ell(u) - \ell(v)$   $O(E)$

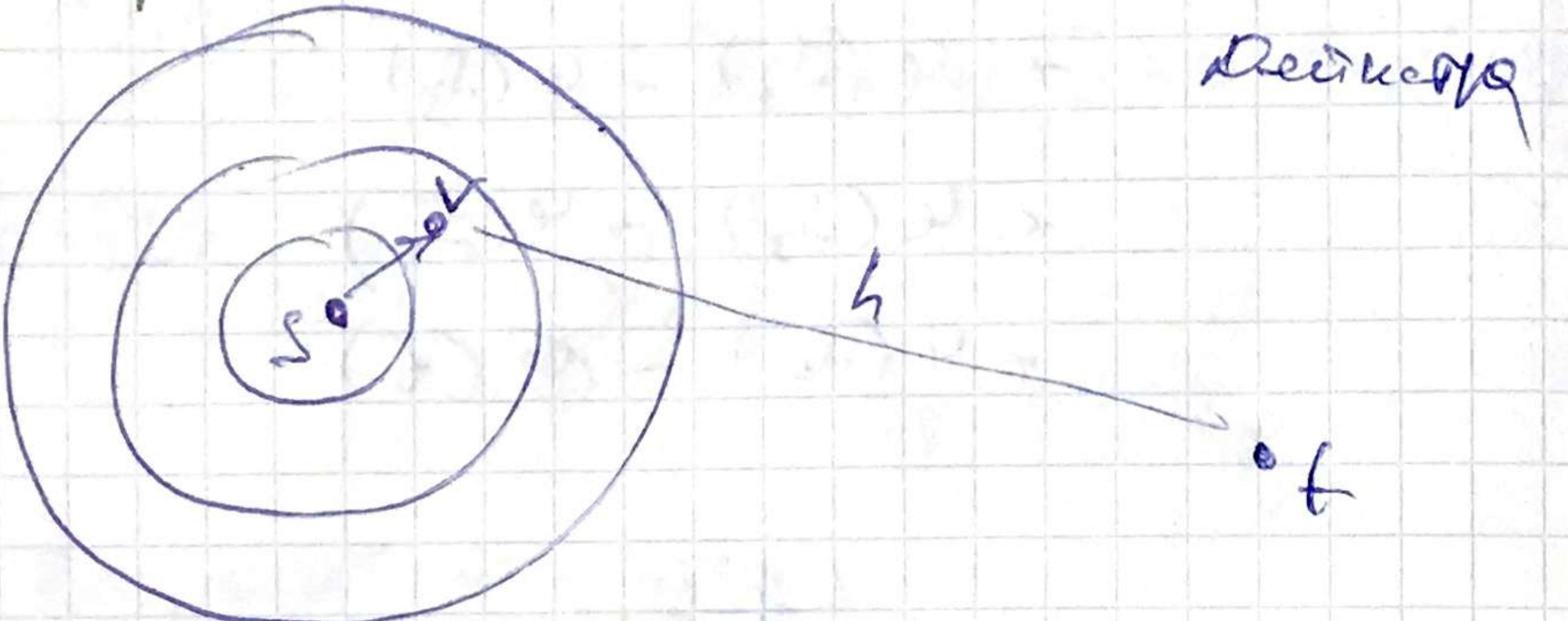
2) Всегда запускаем Динксп.  $O(VE \log E)$

$O(VE \log E)$

Если неограничен  $\Rightarrow$  Равн.; если предок, weight  $R \Rightarrow$   $\Rightarrow$  Охотник

Если разрекурсивен, weight  $> 0 \Rightarrow$  Динксп;

Обратная связь волок



$h$  - reflected сигнал на выходе от  $V_{\text{out}}$

Рецептор: Сумма d искажений  $R = d + h$



## Aleksey

Lemma При подаче ЭК на выход  
стремится КП не уменьшается

- 1) При изменении редиса  $\rightarrow$  бурса
- 2) При добавлении однотипных к КП это тоже бурса



Рассматриваемое промежуток 1) или 2)  $\rightarrow$   
 $\rightarrow$  КП не уменьшается

Ford-Bellman: Утверждение

$$d[v_0 \dots v_n] = \infty$$

$$d[s] = 0$$

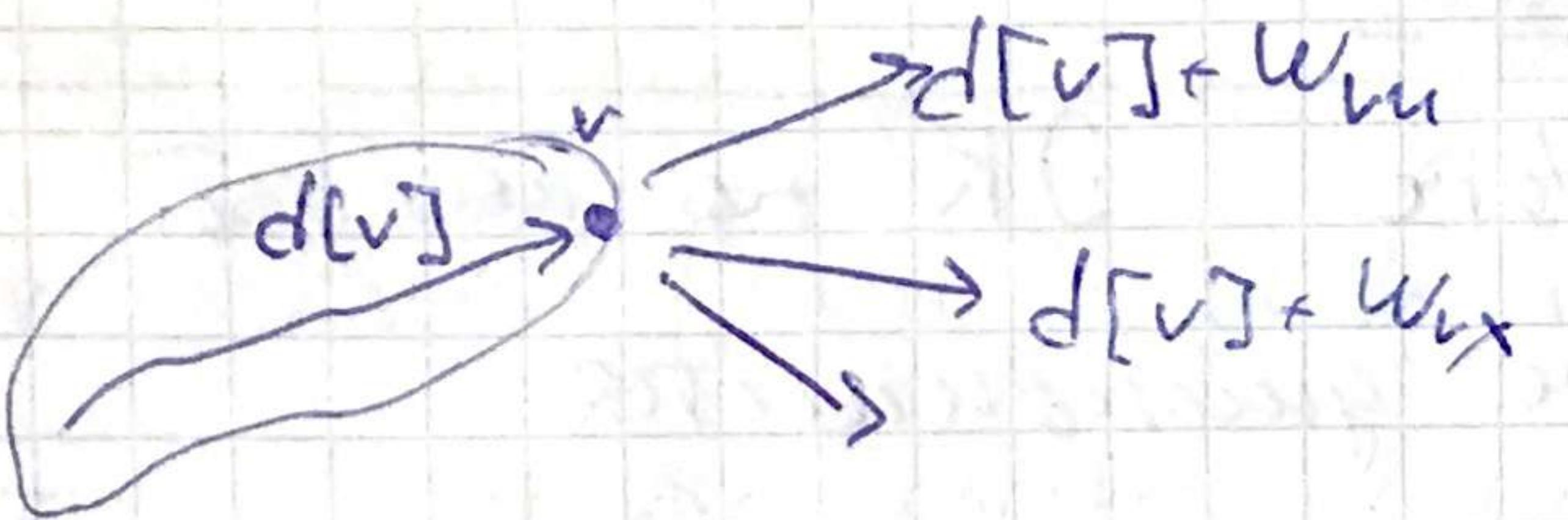
for  $i = 0$  to  $n-1$ :

```
for  $e \in E:$ 
    Relax( $e$ )
```

1) Если на выходе  $\rightarrow$  то  
стремится нет предиката,  
то нет смысла уменьшать

if no Relax  $\Rightarrow$  Break

- 2) Если на выходе нет уменьшения  $d[v]$  то  
изменяется, то нет смысла рассматривать  
редиса, выходящие из  $v$ .



## SPFA (shortest Path Faster Algorithm)

$$d[0 \dots n-1] = \infty$$

$$d[s] = 0$$

$Q = \{s\}$  - ордерація відповідно до відстані від початкової вершини

while ( $\exists Q. \text{Empty}()$ ):

$$v = Q.\text{pop}()$$

for  $(v, u) \in G.E$ :

if  $(\text{Relax}(v, u) \wedge u \notin Q)$ :

$Q.\text{push}(u)$

↙  
 кон-бд  
 Even  $v_x > (V-1)E \Rightarrow$   
 ↗  
 ефі оптим. сум

1) Алергічні коханки BFS

2) Художні виставки  $O(VE)$ , то сказавши  
котиком, що він є співзаконником  
 $O(E)$  (не згадавши нічого)

3) Не піддається оптимізації  
(декарт. while)

## Single Source

Ефі оптим. педна - Понг Бернхард  $O(VE)$

Педна НЕ оптим. педна - Деїктор

$O(E \log V)$  - дескриптор  
найменша

$O(E + V^2)$  - масив

## All Pairs: - Генератор кількох кратних вершин

до кількох  $d[v, u]$  - кількість відстаней

Практичне піднесення:  $V$  поз. засобами  $\Phi_5$  між

Деїктор

$O(V \cdot E \log V)$

$O(V \cdot (E + V^2))$

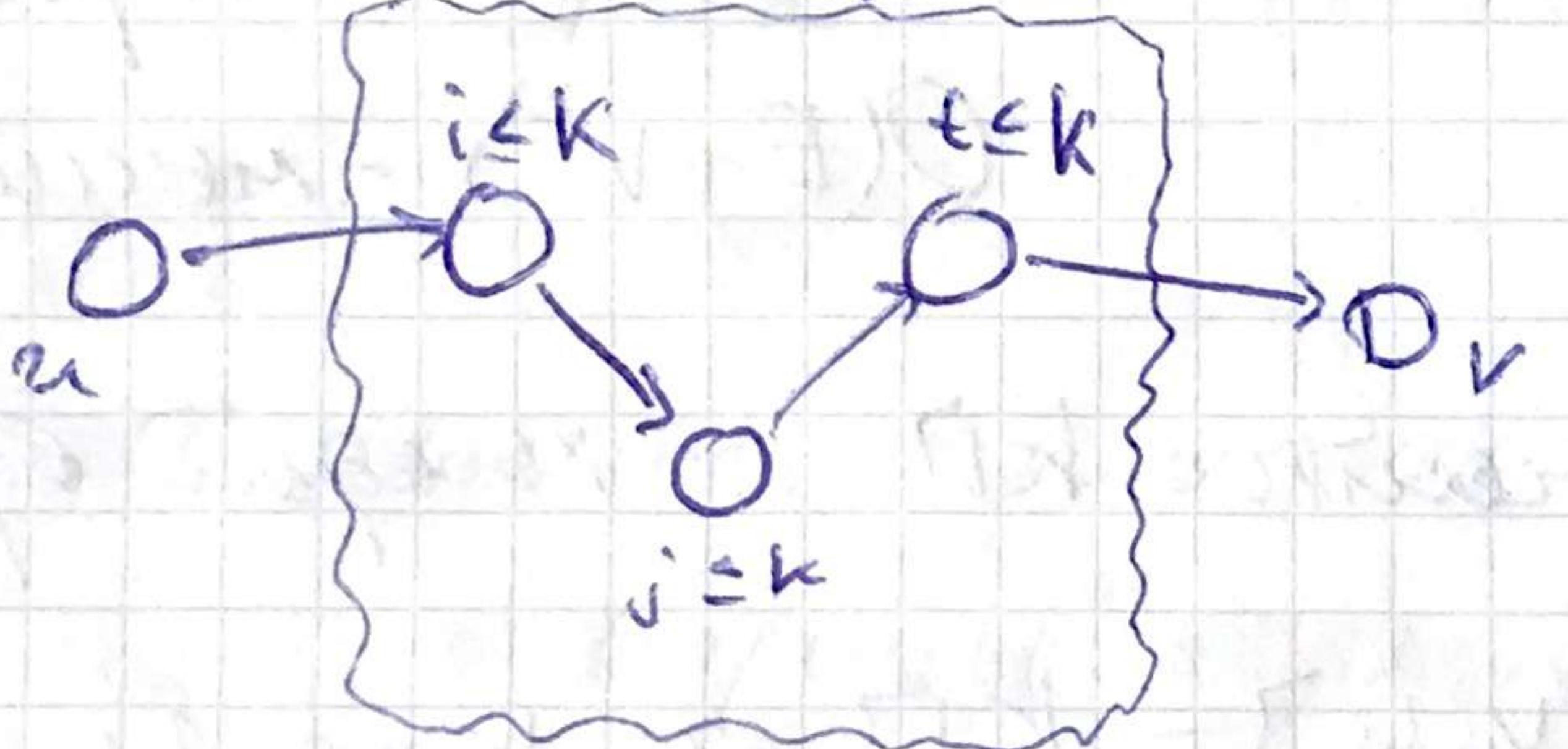
## P-FSPA

	Надане	Надійніше
$E = V$	$\Phi_5$ Декарт. Деїктор Фонінг	$O(V^3)$ $O(V^2 \log V)$ $O(V^2 \log V)$ $O(V^3)$
$E = V^2$	$\Phi_5$ Декарт. Деїктор Фонінг	$O(V^4)$ $O(V^3)$ $O(V^3)$ $O(V^3)$

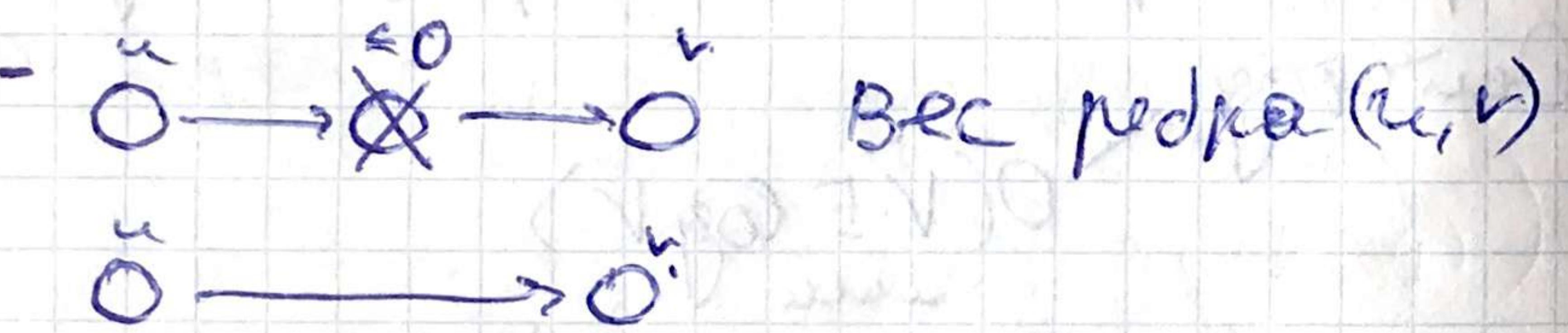
## Алгоритм Пронга

$d^k(u, v) - k\pi$  из  $u \in V$ ,  $v \in V$

максимальное количество ребер в пути  $\leq k$ .  
(где  $u$  и  $v$  — вершины с 1)



Вопрос: что такое  $d^0(u, v)$ ?



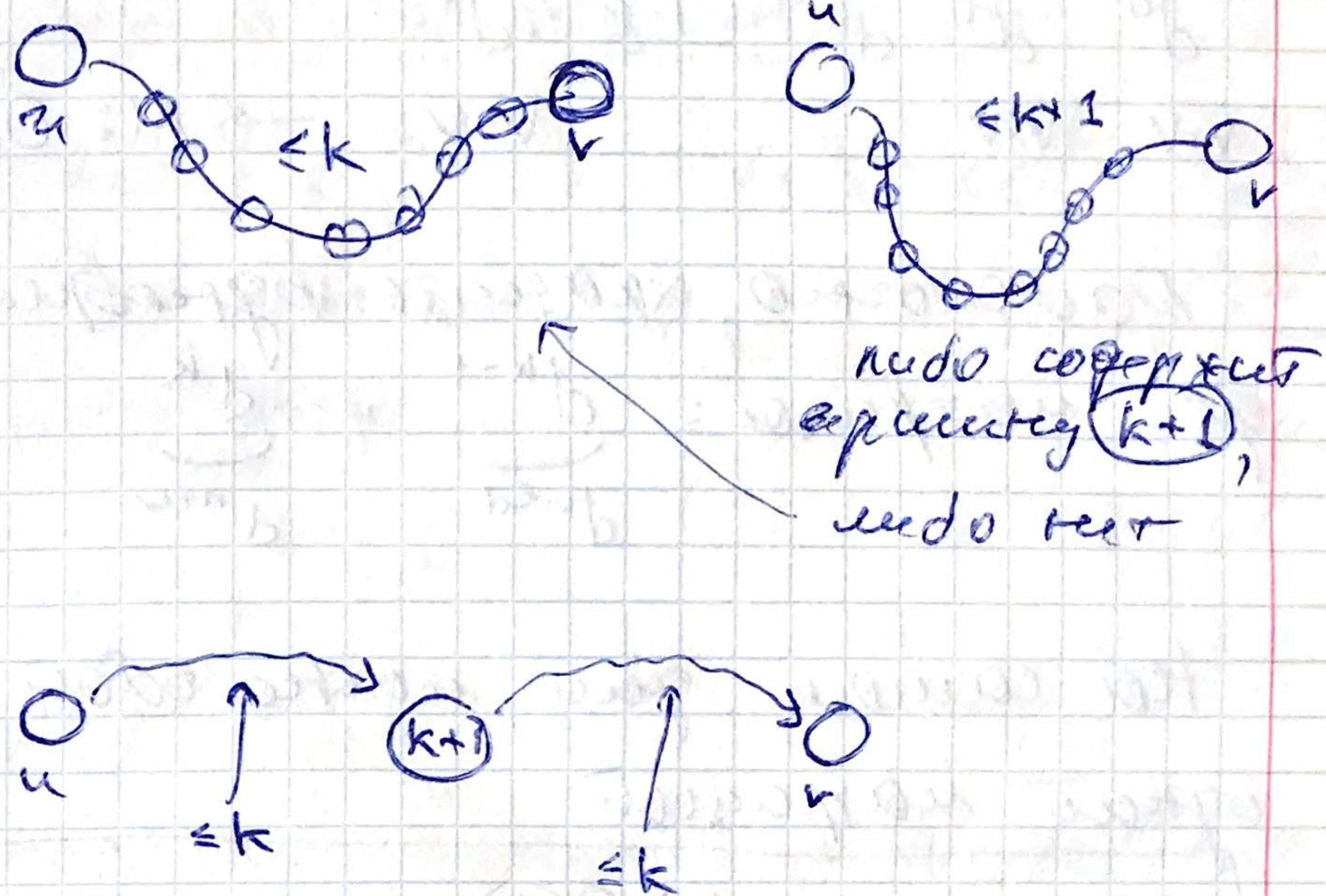
$d^0 = \infty$  — это значение бесконечности

$-d^v(u, v)$ ?

- это значение  $k\pi$

$$d^{k+1}(u, v) = \min(d^k(u, v), d^k(u, k+1) + d^k(k+1, v))$$

Более быстрый алгоритм



for  $k$  from 1 to  $V$ :

for  $x$  from 1 to  $V$ :

for  $y$  from 1 to  $V$ :

$$d^k(x, y) = \min(d^{k-1}(x, y), d^{k-1}(x, k) + d^{k-1}(k, y))$$

Комплексность алгоритма  $O(V^3)$   
(наличие неизвестных)

$$d^0 \quad d^1 \quad d^2 \quad \dots \quad d^V$$

$\frac{V \times V}{V \times V} \rightarrow M: O(V^3)$

Доступное хранение отображения  
это маппинг:  $d^{k-1}$   $\xrightarrow{\text{new}}$   $d^k$

Но с каждым шагом, количество одновременно  
открываемых маппингов

$M: O(1)$  константа

$T: O(V^3)$

$$d(x, y) = \min(d(x, y), d(x, k) + d(k, y))$$

Решение все ОК?

$$d^k(x, y) = d^{k-1}(x, k) + d^{k-1}(k, y)$$

$$d^{k-1}(x, k) + d^k(k, y)$$

$$d^k(x, k) + d^{k-1}(k, y)$$

$$d^k(x, k) + d^k(k, y)$$

$$d^k(x, k)$$

$$d^k(k, y)$$

$$\stackrel{!!}{d^{k-1}(x, k)}$$

$$\stackrel{!!}{d^{k-1}(k, y)}$$

Проверка отриц. связей:

Ут. Если  $x \in C^-$ , то  $d^V(x, x) < 0$



О-бз: очевидно

Число итераций:

Несмотря на то что отрицательные ребра, это не  
отрицательных связей. Хорошо предположить  
что, это же возвращается к действительной  
 $O(V^2 \log V)$   
[это сильно лучше  $O(V^3)$  и оправдано]

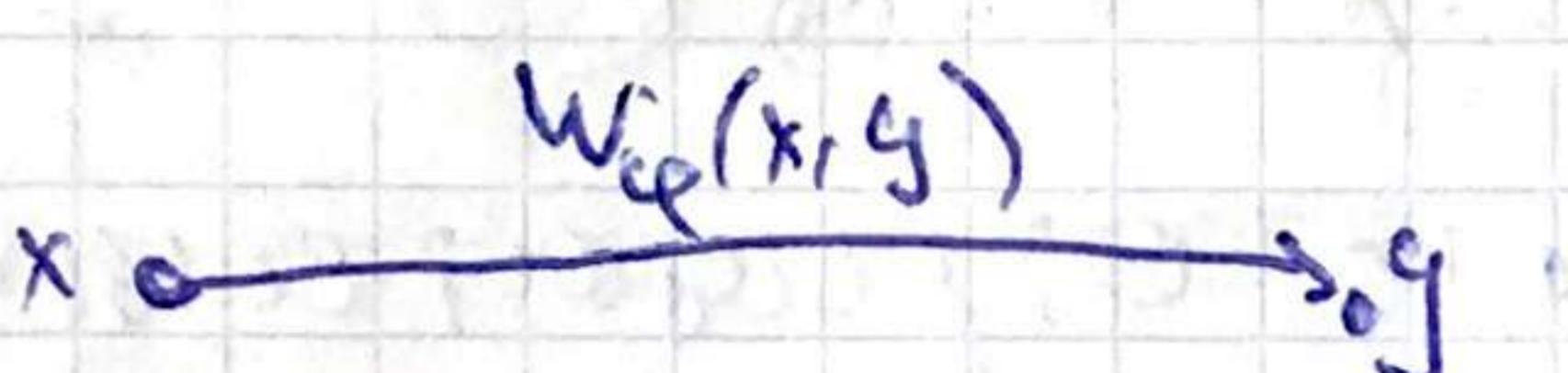
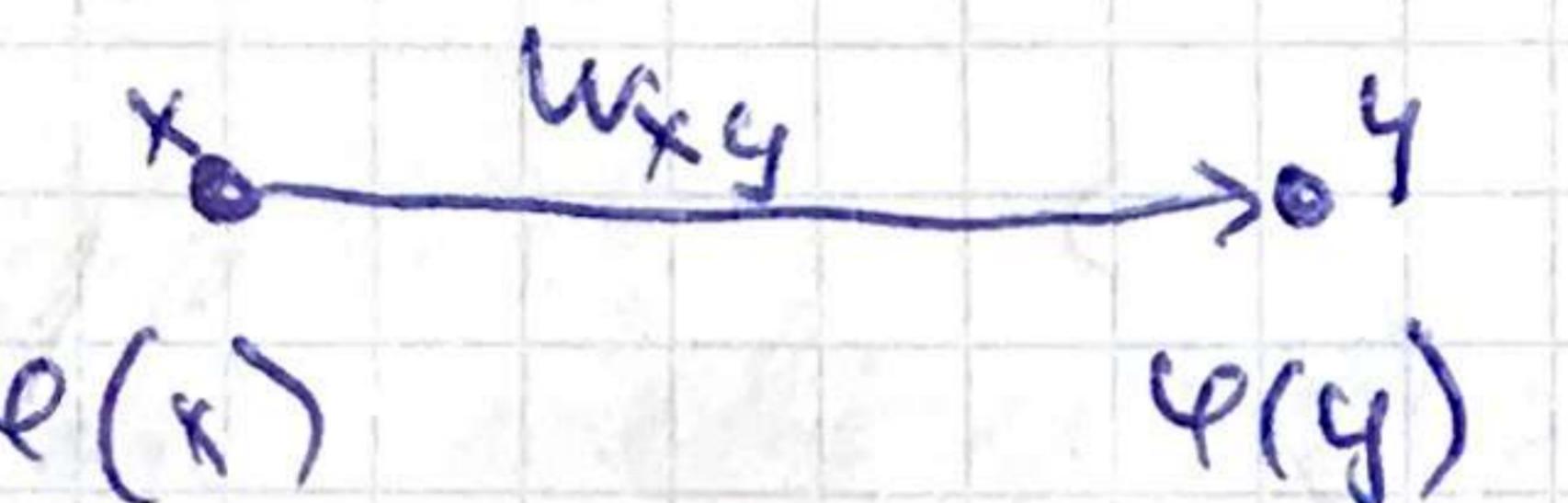
$\left\{ \begin{array}{l} G \text{ с отриц. ребрами} \\ G' \subset \text{неотриц. ребра} \\ k \cap G = k \cap G' \end{array} \right\} \Rightarrow$

Метод кратчайших

$$\varphi: V \rightarrow \mathbb{R} \quad \varphi(v) = 10$$

Определение задачи:  $G_\varphi : VG_\varphi = (G, V, G, E)$ ,

$$VW_\varphi(x, y) = W(x, y) + \varphi(x) - \varphi(y)$$



Задача: Рассмотрим  $P$ -множество  $G$ , а  $P'$ -дополнительное множество  $G_\varphi$  задачи  $G_\varphi$  (предполагается, что  $x$  и  $y$  вершины  $G$ )

$$P'_\varphi(u, v) = P_\varphi(u, v) + \underbrace{\varphi(u) - \varphi(v)}_{\text{веса } P'}$$

где  $P'$  дополнение  $P$

$$\begin{aligned} \text{Д-р: } P'_\varphi(u, v) &= \sum_{e \in P'} w_e(e) = \\ &= \sum_{e \in P'} (w(e) + \varphi(e.\text{from}) - \varphi(e.\text{to})) = \\ &= \sum_{e \in P} w(e) + \sum_{e \in P'} (\varphi(e.\text{from}) - \varphi(e.\text{to})) = \end{aligned}$$

Следствие 1: Если  $P$ -КНН в  $G$ , то  $P'$ -КНН в  $G'$  ( $u \neq v$ ), т.к.

одна определенная вершины, которая не является вершиной конкретного узла.

$$\begin{aligned} P'_\varphi(u, v) &= P(u, v) + \underbrace{\varphi(u) - \varphi(v)}_{\text{const}(u, v)} \\ &\downarrow \min \quad \downarrow \min \end{aligned}$$

Следствие 2: Веса гамильтонов в  $G_\varphi$  не монотонны (относительно  $G$ )

$$P'_\varphi(u, v) = P_\varphi(u, v) + \underbrace{\varphi(u) - \varphi(v)}_0, u=v$$

Задача: Предположим  $\varphi: W \geq 0$  и  $(u, v)$  th Тогда  $\varphi \exists \Leftrightarrow \nexists$  оптим. в  $G$

Д-р:  $\Rightarrow$  | Веса всех гамильтонов в  $G_\varphi \geq 0$ .

но  $\nexists$  оптим. в  $G$  все гамильтоны  $\geq 0$ .

$\Leftarrow$  Доказем, что  $s \in S^*$ .

Добавим в  $S^*$  вершину  $s$ :

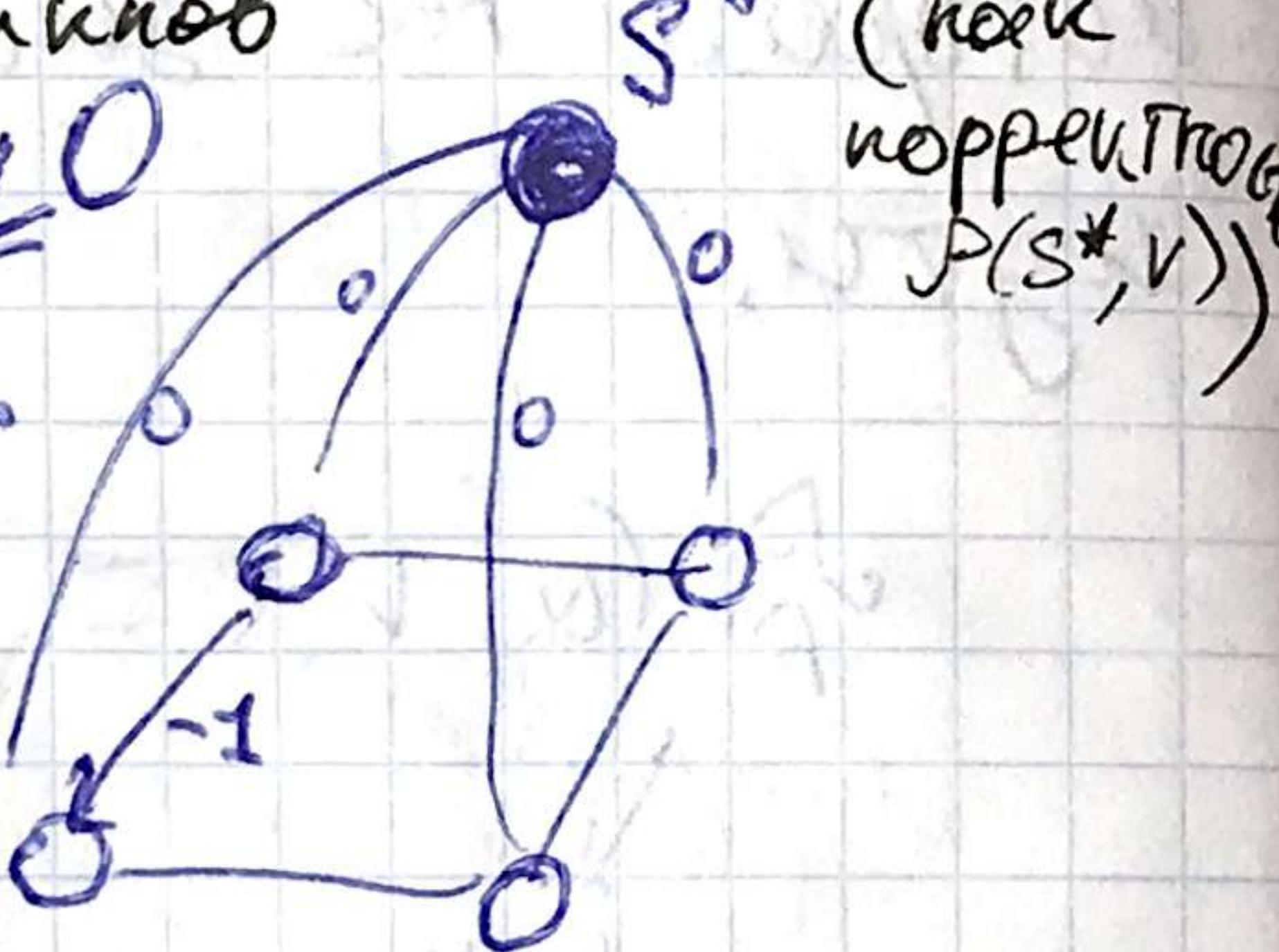
из нее будут идти все ее смежные с вершиной  $s$ .

т.к. исходное множество отсутствует  
отрицательных

$s \in S^*$  (так как  
проверено, что  
 $p(S^*, s) > 0$ )

$$\text{Покажем } \varphi(s) = p(S^*, s).$$

[Достаточно доказать, что для каждого  $v \in V$  выполнено



Нужно показать, что для каждого  $w \in V$  выполнено  $w_e \geq 0$ ?

$$w_e(u, v) = w(u, v) + \varphi(u) - \varphi(v) = \\ = w(u, v) + p(S^*, u) - p(S^*, v) \geq 0$$

покор. то  
запись из  $S^*$  в  $V$

$$S^* \xrightarrow{0} u \xrightarrow{0} v$$

запись из  $S^*$  в  $V$

$$S^* \xrightarrow{0} u \xrightarrow{0} v$$

## Алгоритм Дijkstra

0(V) 1) Задаем любую вершину  $s^*$  и проводим радиус  $O^{(r)}$  веса до всем остальным

0(VE) 2) Вычисляем  $\Phi$  в  $\Phi$  для каждой из  $s^*$ .

$$3) \varphi(v) = p(S^*, v)$$

4) На конкретной строке  $V$  разбираем алгоритм Dijkstra.  $O(VE \log V)$

$O(V(E+V^2))$  — для всех

Time:  $O(VE \log V)$  или  $O(V(E+V^2))$

POBREX.  $\underline{O(V^2 \log V)}$  или  $\underline{O(V^3)}$

настройки  $\underline{O(V^3 \log V)}$  или  $\underline{O(V^3)}$

Эвристическоеущий поиск для нахождения  
вершинами ( $A^*$ )



$$d(u) + h(u) \rightarrow \min$$



$h(u)$  — текущая  
оценка  
погибы  
 $B$  —  $h(u)$  — оценка  
погибы

# RMQ $\cup$ RSQ (Range Minimum Query, Range Sum Query)



Предусмотреть операции на запроцах:

- Min ( $l, r$ ) - минимальн ( $A[l..r]$ )

- Sum ( $l, r$ ) - сумма ( $A[l..r]$ )

0	1	2	5	3	4
---	---	---	---	---	---

$O(n)$

$$\text{Min}(2, 4) = 2$$

$$\text{Sum}(2, 4) = 10$$

1) static RMQ/RSQ - массив не меняется

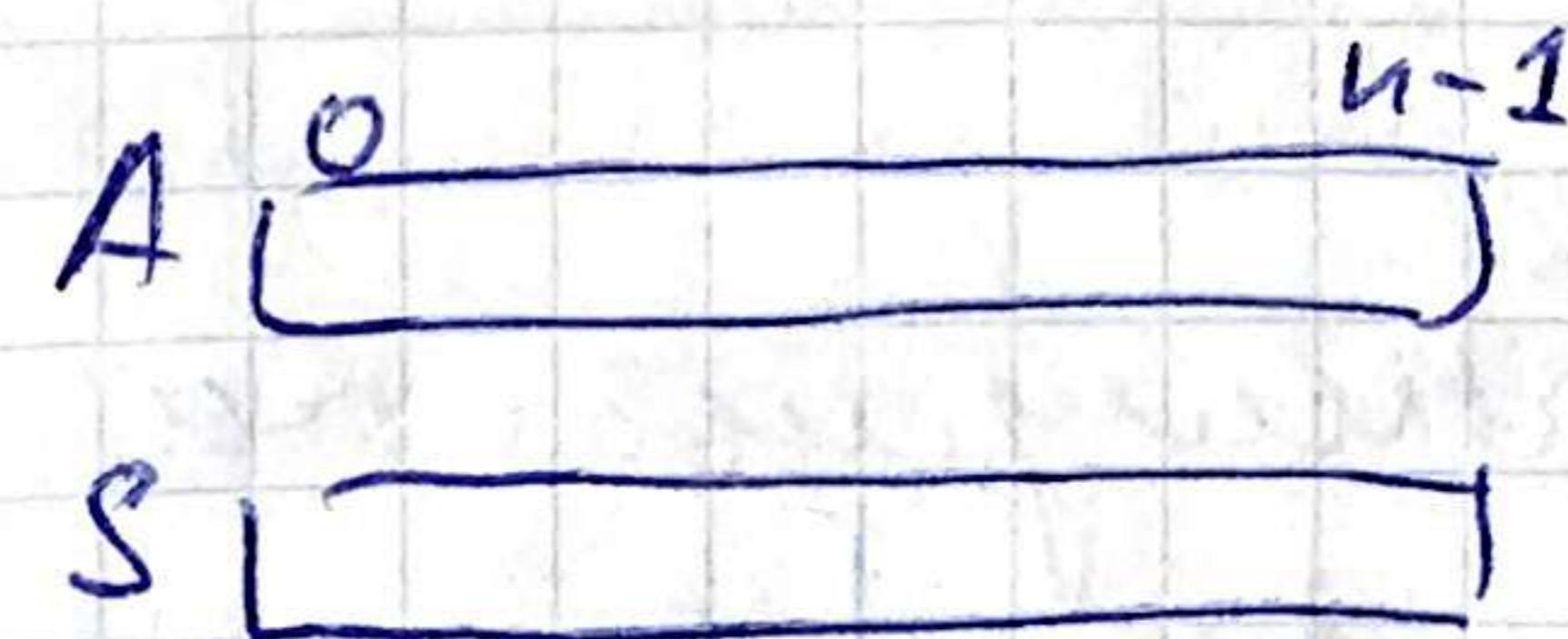
2) dynamic RMQ/RSQ -  $\Rightarrow$  1-го, неизменяется

(есть запрока update)

## 1) static RSQ

массив ~~запоминающий~~ скользящий (Partial Sum Array)  
 (Кумулятивный)

$$S[k] = S[k-1] + A[k]$$



$$S[k] = \sum_{i=0}^k A[i]$$

ногает  $O(n)$

A   0 5 2 3	-----
S   0 5 7 10	

операции:

✓ доступность

✓ обратимость

$$\sum_{i=l}^r A[i] = S[r] - S[l-1]$$

запрос  $O(1)$

## 2) static RMQ

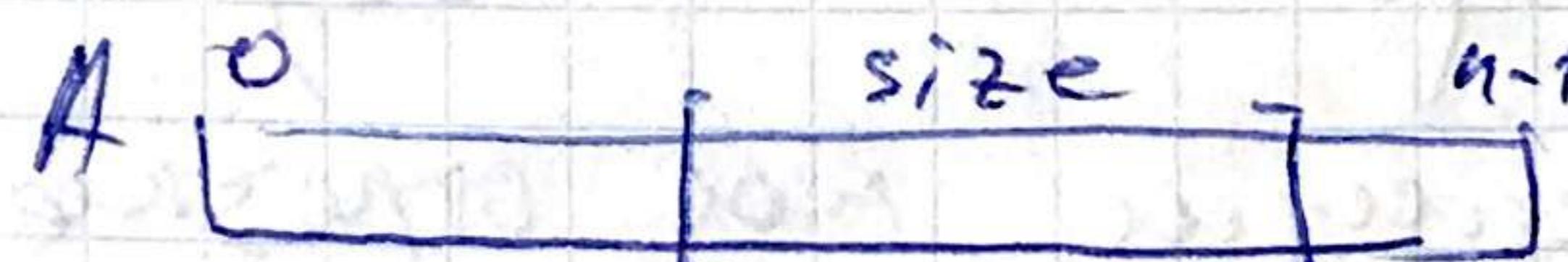
операция min:

цидемпотентна

• Недостаток - CBO-CBO Операция  $\oplus$ :

$$X \oplus X = X$$

$$\min(x, x) = x$$

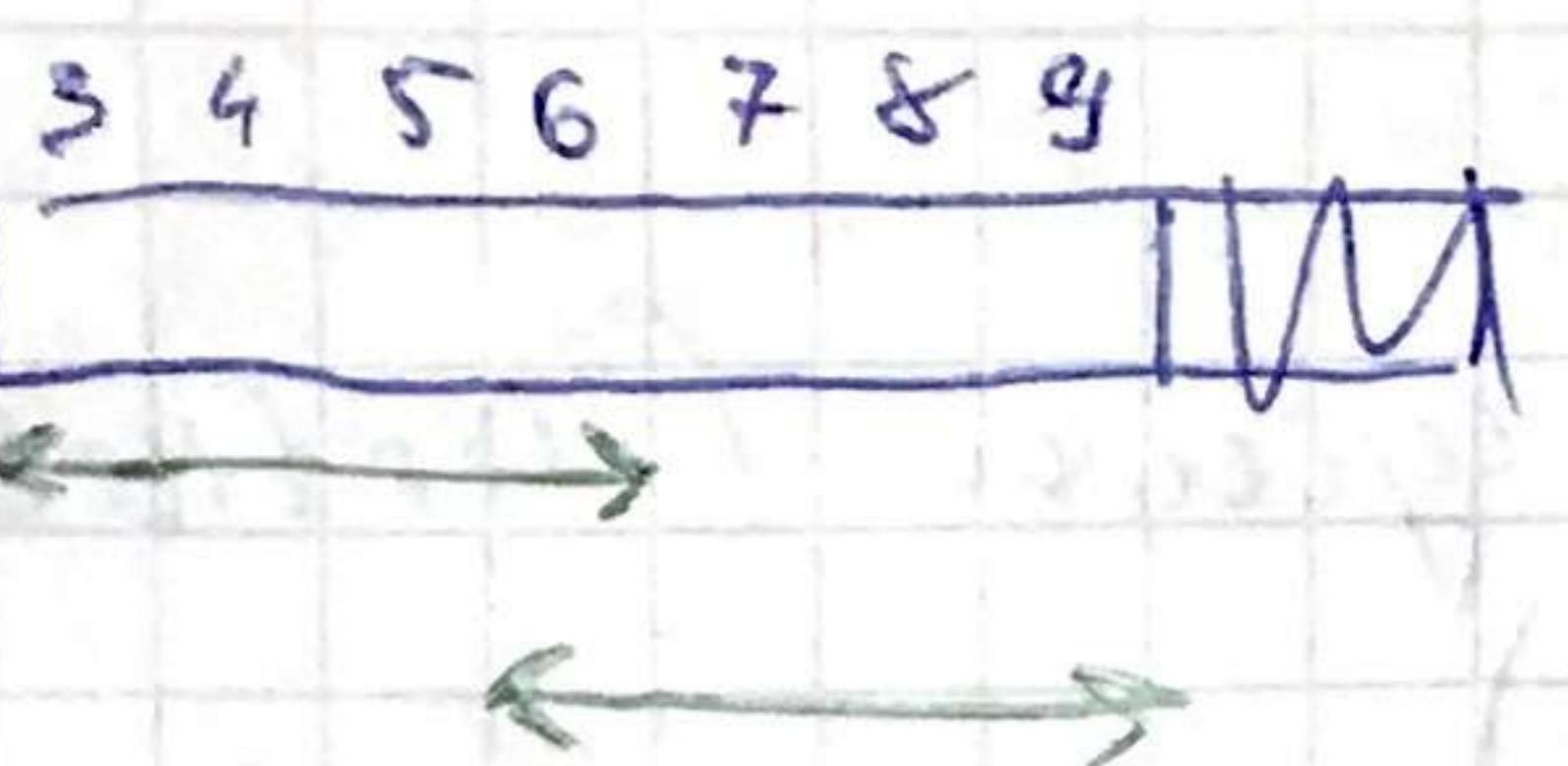


$$\text{size} = r-l+1$$

$$L = 2^k \leq \text{size}$$

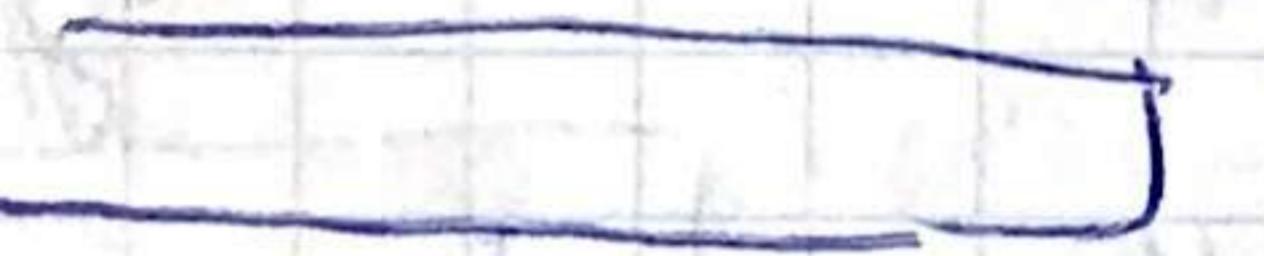


$$k = \arg\max_x (2^x \leq \text{size})$$



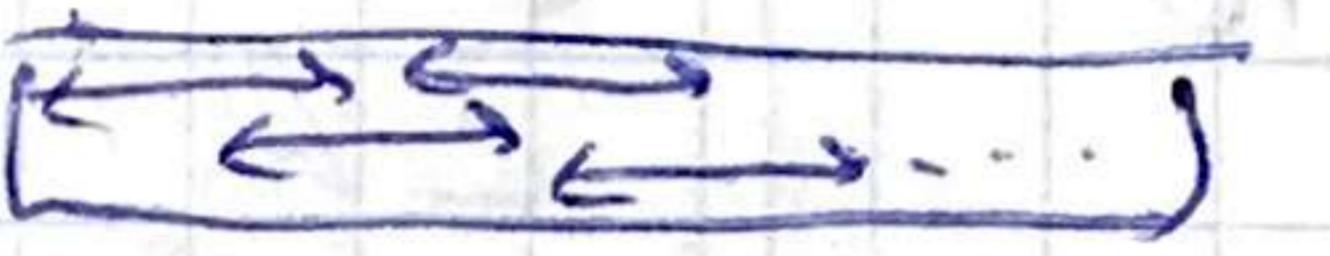
Числа: Рассматриваются на отрезках длины  $2^k$   $\forall k \leq \log n$   
как  $k$ -битные

$$2^k = 1$$



$$\sim O(1)$$

$$2^k = 2$$



$$\sim O(2)$$

$$2^k \leq n$$

$$\sim O(n)$$

Memory:  $O(n \log n)$

~~State~~ Table (параметрическое представление)

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

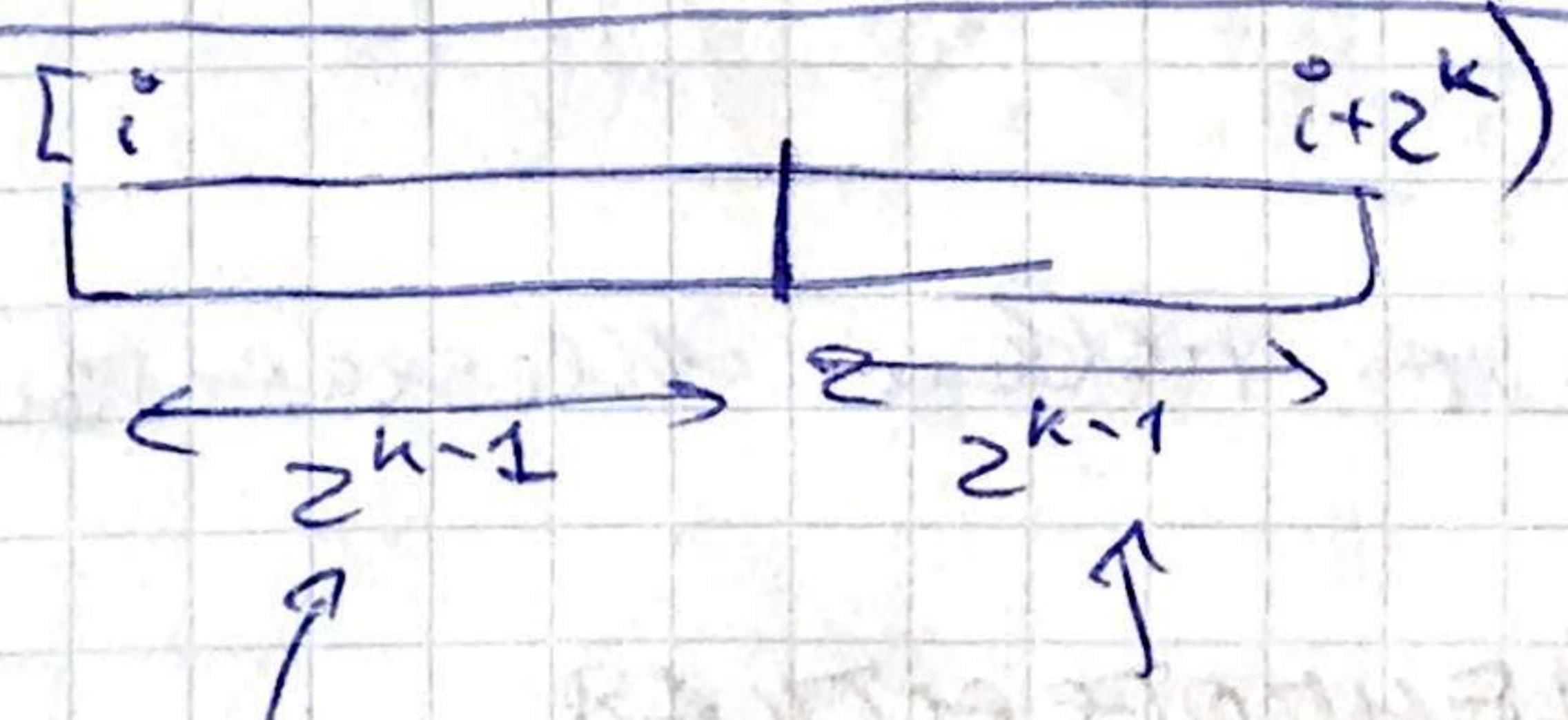
наибольшее значение на отрезке

длины  $2^k$  с центром в  $i$ .

$$ST[0][i] = \min(A[i, \dots, i+2^0-1]) = A[i]$$

Рекурсия  $ST[n-1][i]$  с обновлением  $\forall i$

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



$$ST[k-1][i] \quad ST[k-1][i+2^{k-1}]$$

A	5	0	3	2	1
---	---	---	---	---	---

ST

$$k=0$$

$$k=1$$

$$k=2$$

	0	1	2	3	4
5	5	5	5	5	5
0	0	0	0	0	0
2	2	2	2	2	2
1	1	1	1	1	1
X	X	X	X	X	X

Рассмотрение:  $O(n \log n)$

Задача: Query( $l, r$ ):  $[l, r]$  // Задача:  $O(1)$

$$[l, r] \xrightarrow{2^k} ST[k][l]$$

$$[l, r] \xrightarrow{2^k} ST[k][r-2^k+1]$$

refer to  $\min(ST[k][l], ST[k][r-2^k+1])$

$$k = \lfloor \log(r-l+1) \rfloor$$

5	0	3	2	1	e
0	0	3	1	X	
0	0	X	X	X	

Можно добиваться любой элемента

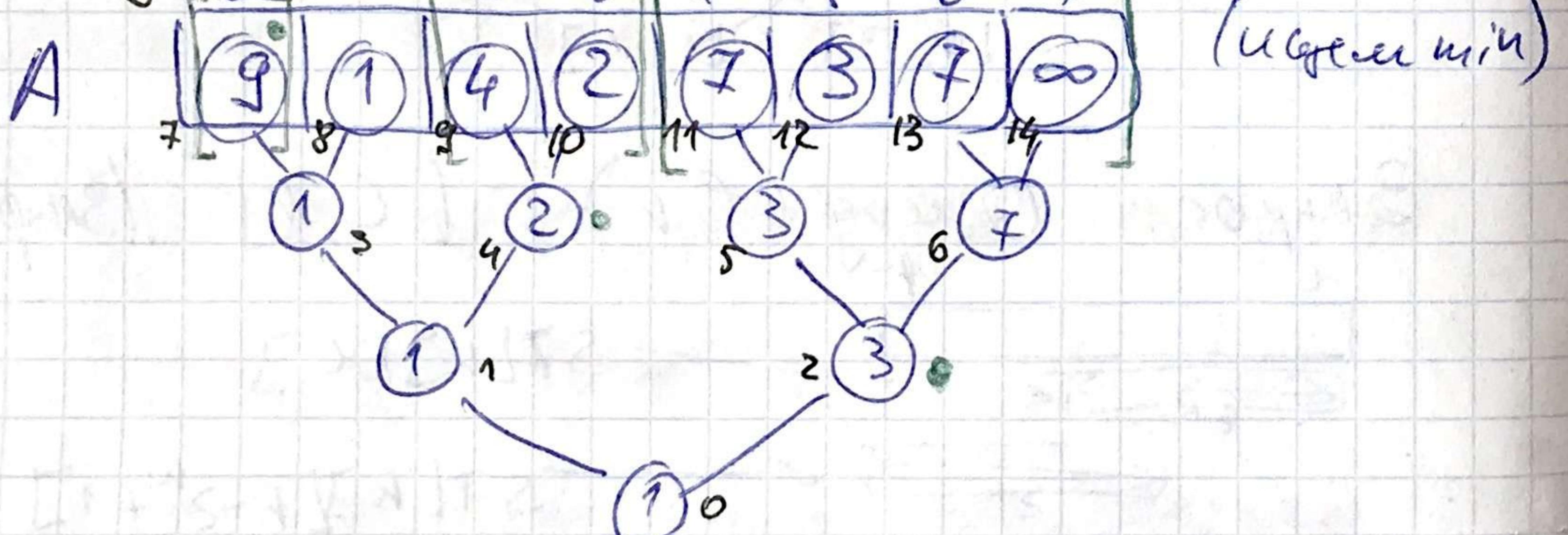
Операция  $\oplus$ : ИДЕМПОТЕНТНАЯ

АССОГИАТИВНАЯ

(коммутативность) - ???

3) dynamic RMQ/RSQ

Segment Tree (Дерево отрезков)



Построение: 1) Дополним A до следующего размера  $n$

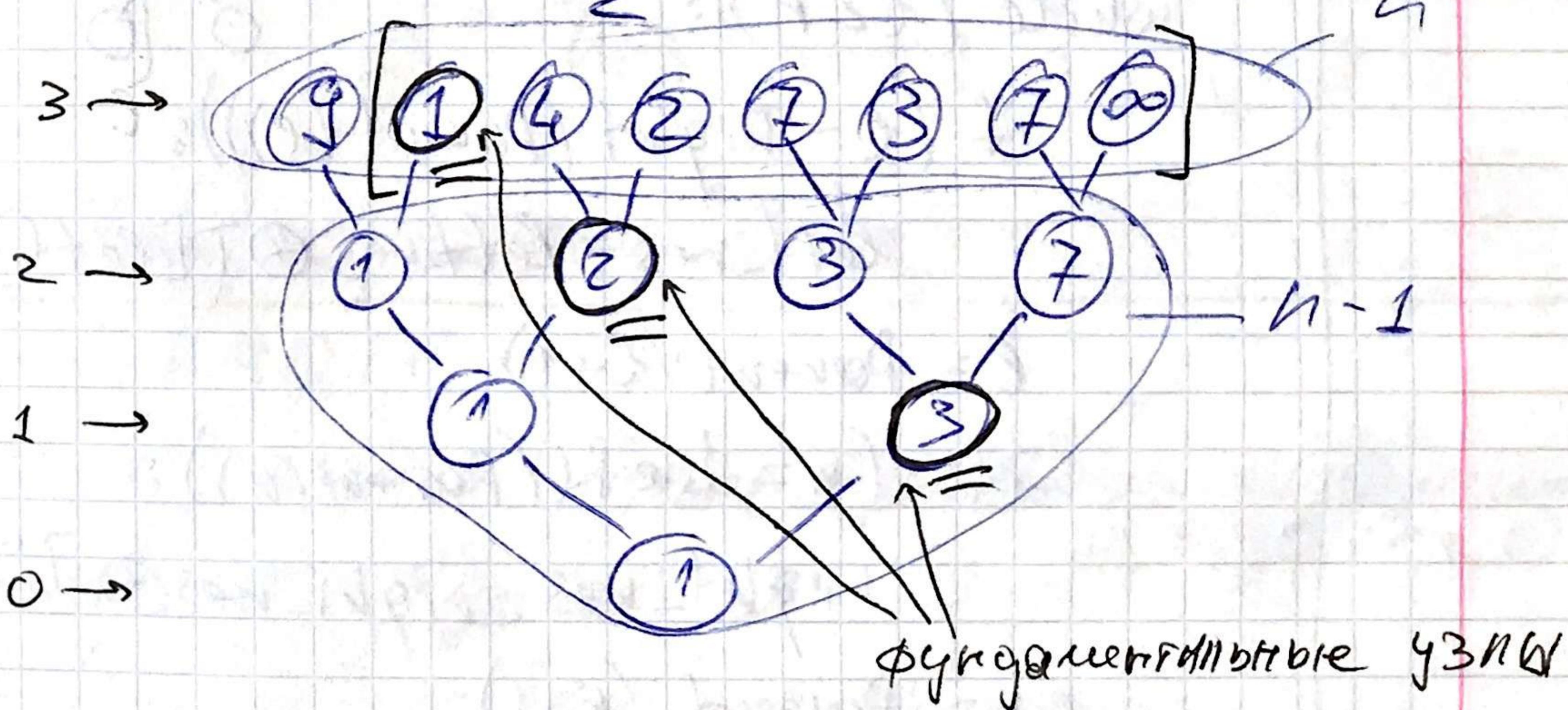
- 2) Задаем дерево  $T$  для  $n-1$
- 3)  $T[1, \dots, 2n-1] = A$  // нечто
- 4) for ( $i = n-1$  to 0):

$$T[i] = \min(T[\text{Left}(i)], T[\text{Right}(i)])$$

$$\text{Left}(i) = 2i + 1$$

$$\text{Right}(i) = 2i + 2$$

$$\text{Parent}(i) = \frac{i+1}{2}$$



Учт.: на каждом уровне  $\leq 2$  функциональных узлов  
допустимых  $n \geq 3$ .

Тогда 2 из 3<sup>1</sup> не функциональны! Поэтому то  
какие переходят в родителя и количество кузине  
разделение  $\rightarrow 3^k$  листей не может

Как устроены фрагменты деревьев в задаче?

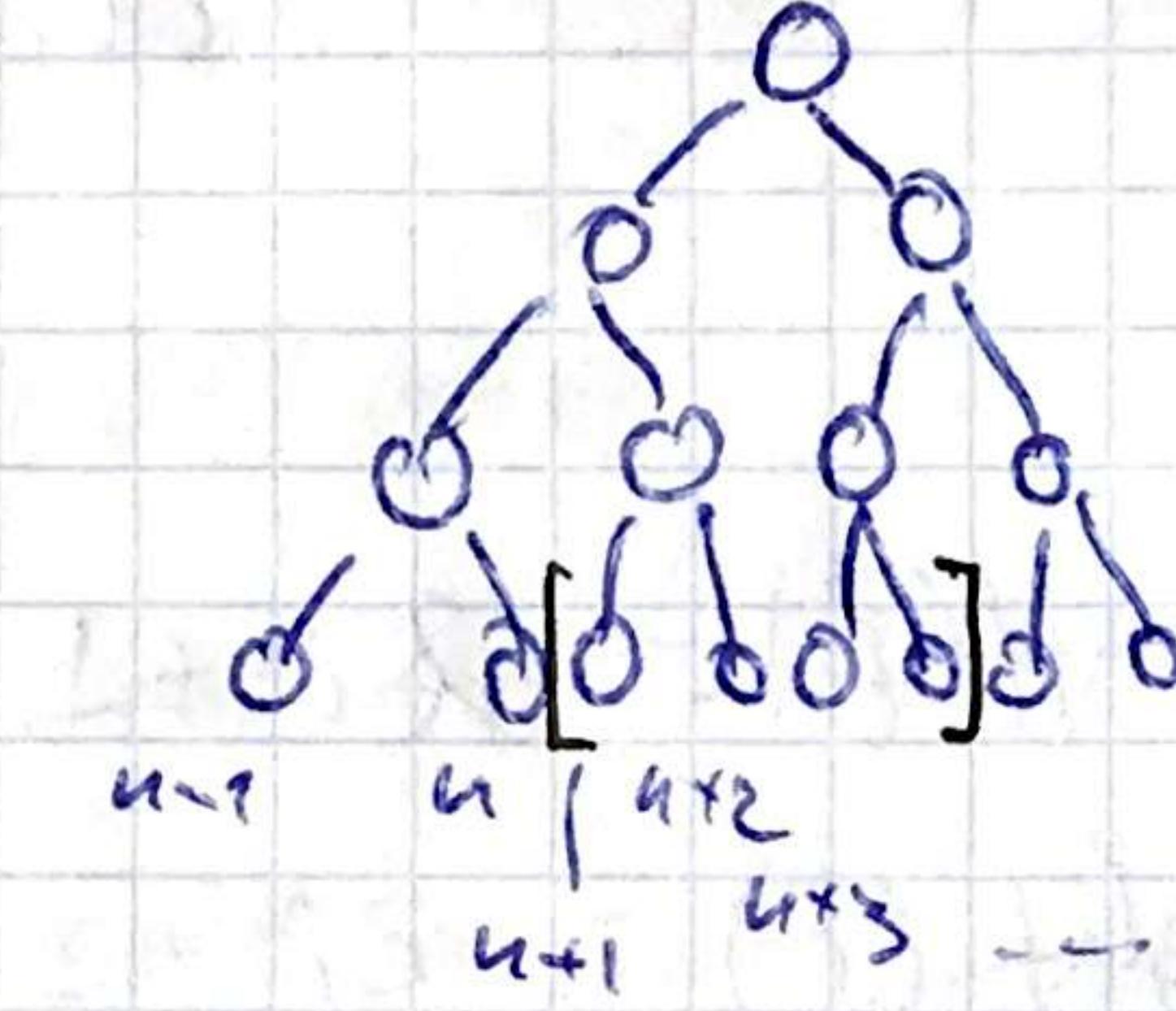
1) Запрос среза блоком (от  $l$  до  $r$  в кратце)

$RMQ(l, r)$ :

$$l += (n-1)$$

$$r += (n-1)$$

$left\_res = \emptyset$



$right\_res = \emptyset$

while ( $l < r$ ):

if ( $l = \text{Right}(\text{Parent}(e))$ ):

$left\_res = left\_res \oplus T[\text{left}]$

$l = \text{Parent}(l+1)$

if ( $r = \text{Left}(\text{Parent}(r))$ ):

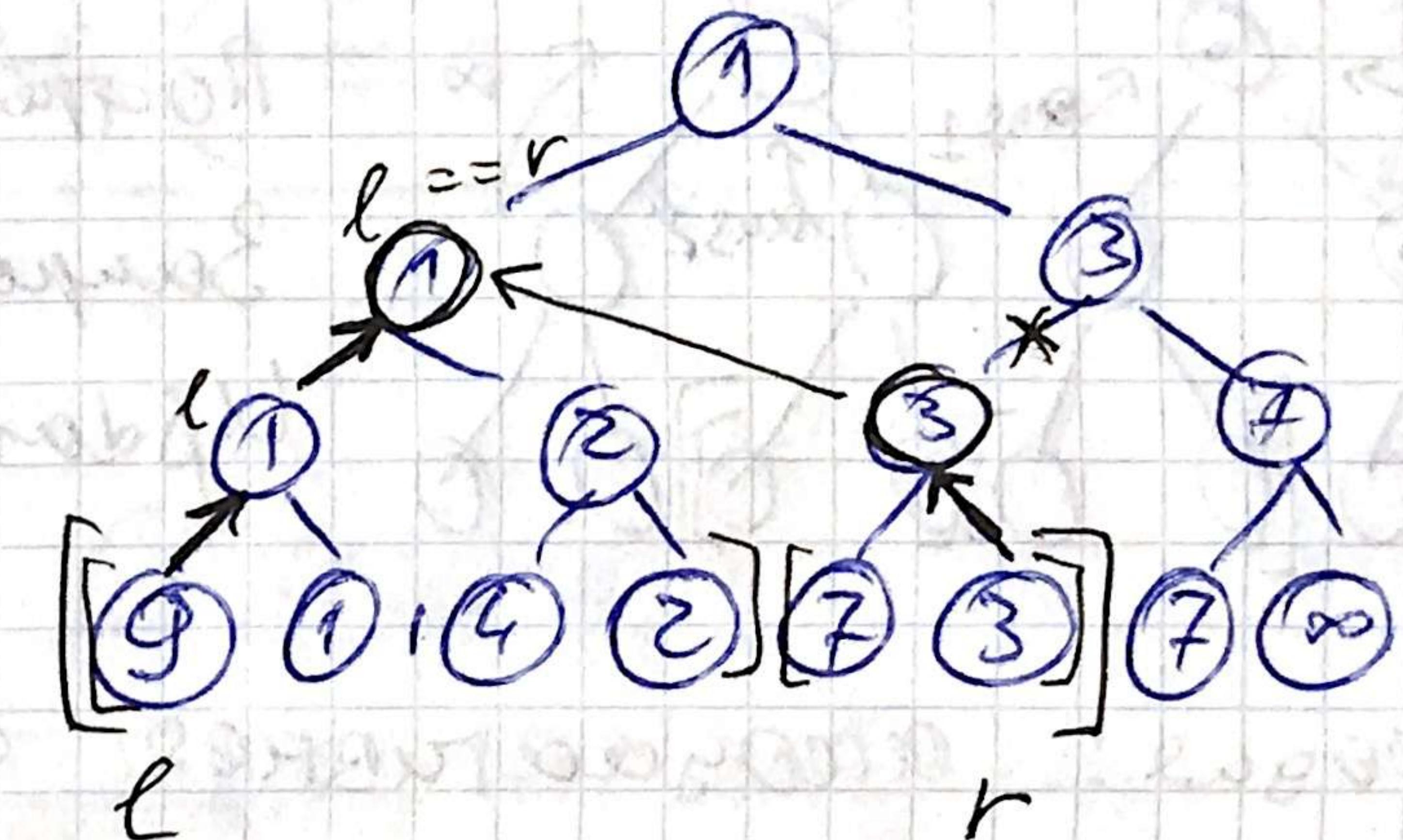
$right\_res = right\_res \oplus T[\text{right}]$

$r = \text{Parent}(r-1)$

if ( $e == r$ ):

$left\_res = left\_res \oplus T[\text{left}]$

return  $left\_res \oplus right\_res$



$left\_res = 1$

$right\_res = 3$

$(1, 3) \rightarrow 1$

2) Запрос среза блоком

$RMQ(l, r, i)$ :

$[l_6, r_6] = \text{node}(i)$  — корень поддеревьев в соответствии с задачей определения  $i^{th}$  задачи

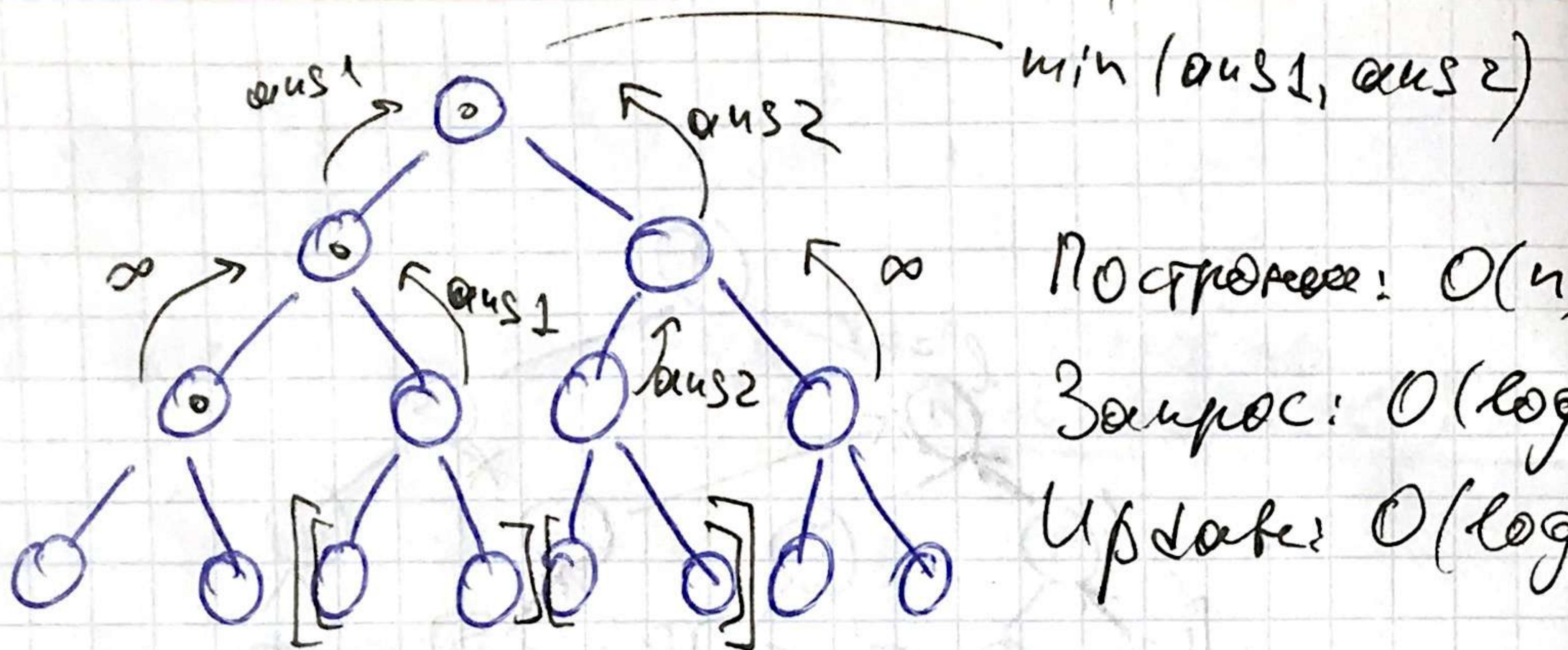
$RMQ(l, r, 0)$

if ( $[l_6, r_6] \cap [l, r]$ ) return  $\infty$ ;

if ( $[l_6, r_6] \subseteq [l, r]$ ):

return  $T[i]$

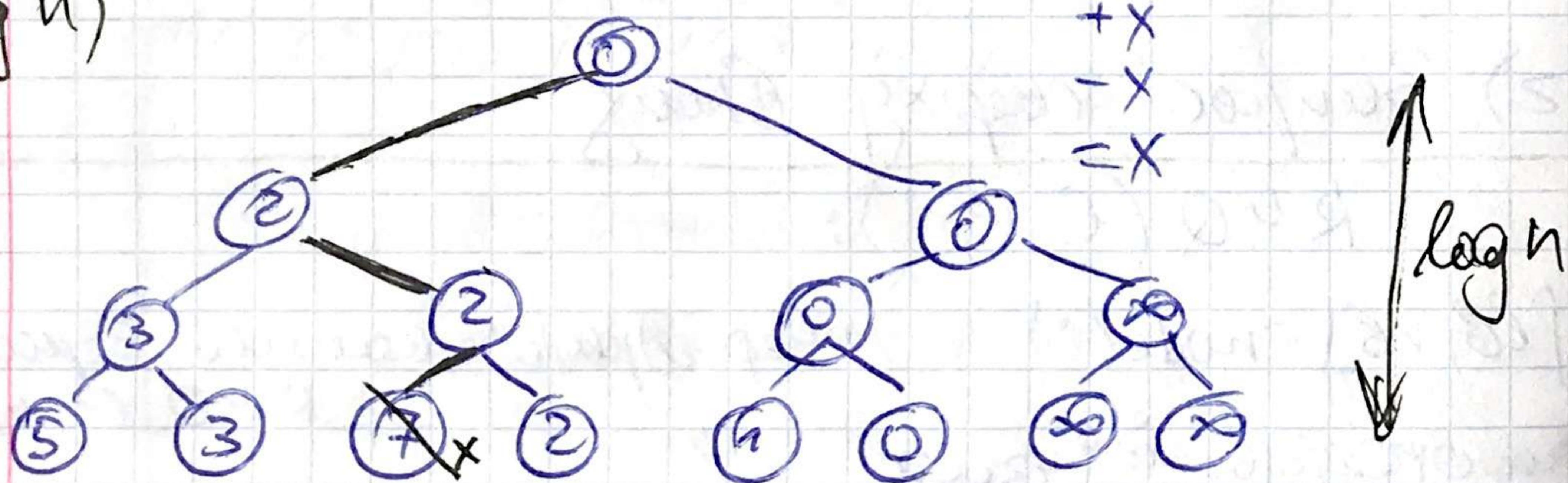
return  $RMQ(l, r, \text{Left}(i)) \oplus RMQ(l, r, \text{Right}(i))$



One page: ~~array structure~~ ~~tree~~ ~~recursion~~  
 2 pages

3) Update  $A[5|3|7|2|1|0]$

$O(\log n)$



$update(i, x) := (A[i] = x)$

$i += (n-1)$

$T[i] = x$

do

$i = \text{Parent}(i)$

$T[i] = \min(T[\text{Left}(i)], T[\text{Right}(i)])$

while ( $i \neq 0$ ):

## СЕМУНДАР

14.11.20

1)  $\#x \geq \frac{n}{2}$  page

2)  $\#x \geq \frac{n}{3}$  page

-2

-2

-2

-2

-2