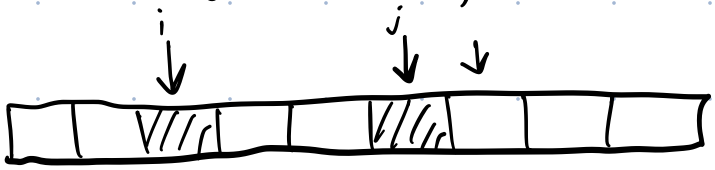
ОЧЕРЕДЬ (queue)

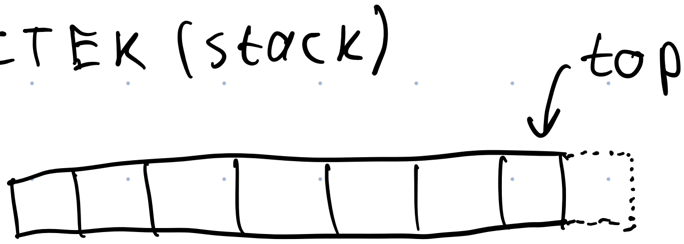
FIFO (first in, first out)



q.push_back(new_elem) O(1)

q.pop() O(1)

СТЕК (stack)



LIFO (last in, first out)

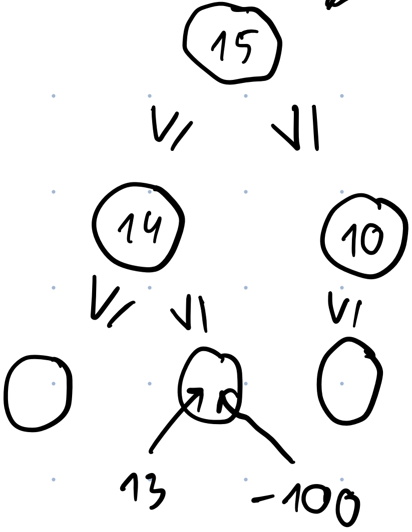
push

pop

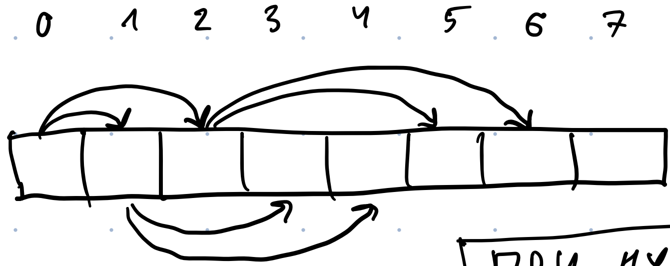
```
for ......
    if (stack.top() == new_elem):
        stack.push(new_elem)
    else:
        stack.pop()
```

КУЧА (Heap)

КУЧА НА МАКСИМУМ



Почти полное бин. деревом



$i \to 2i+1$

$i \to 2i+2$

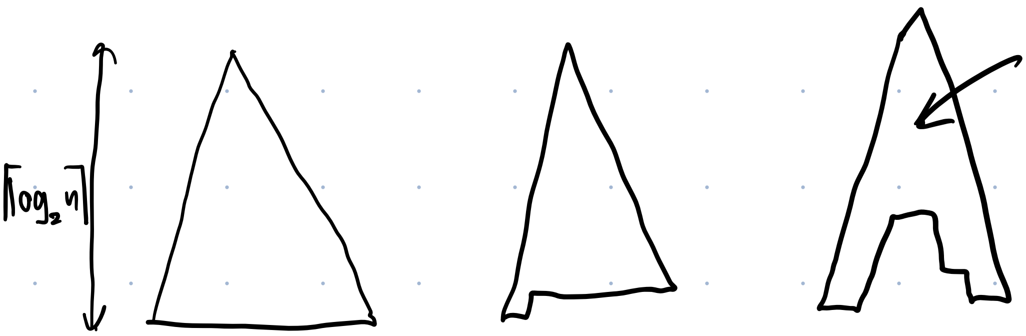
(при нум. с 0)

При нум. с 1

$i \to 2i$

$i \to 2i+1$

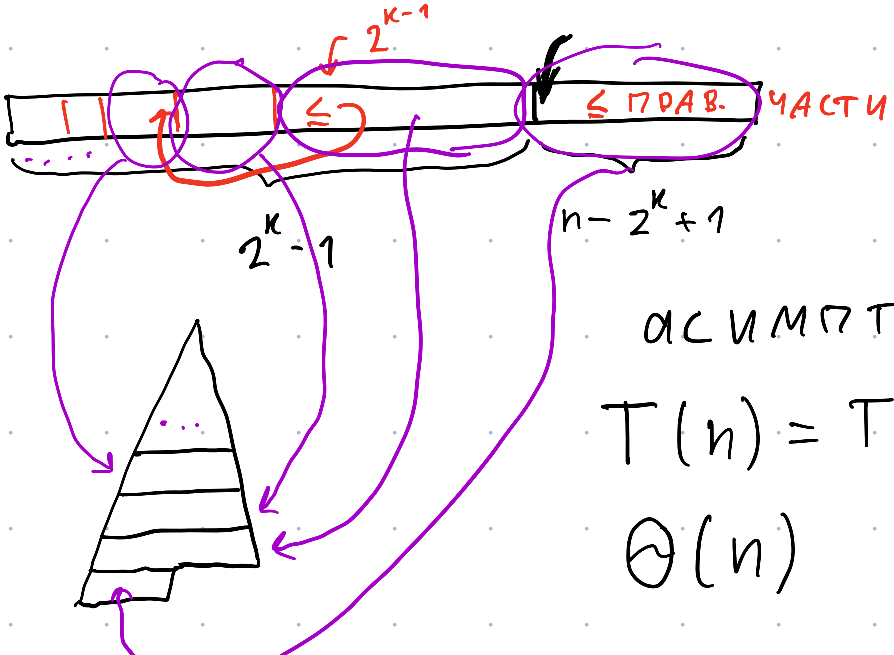
Почти полн. бин. дерево:



не поч. полн. бин. дерево

Для n элем. $\lceil \log_2 n \rceil$

Как построить кучу на n элем.?

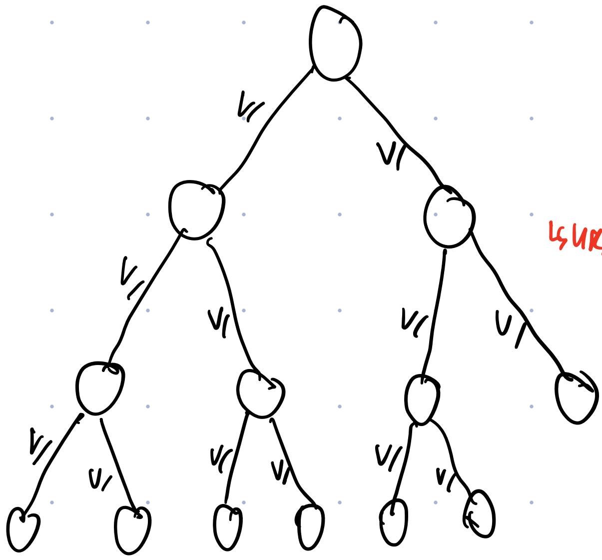


асимптотика:

$$T(n) = T\left(\frac{n}{2}\right) + cn$$

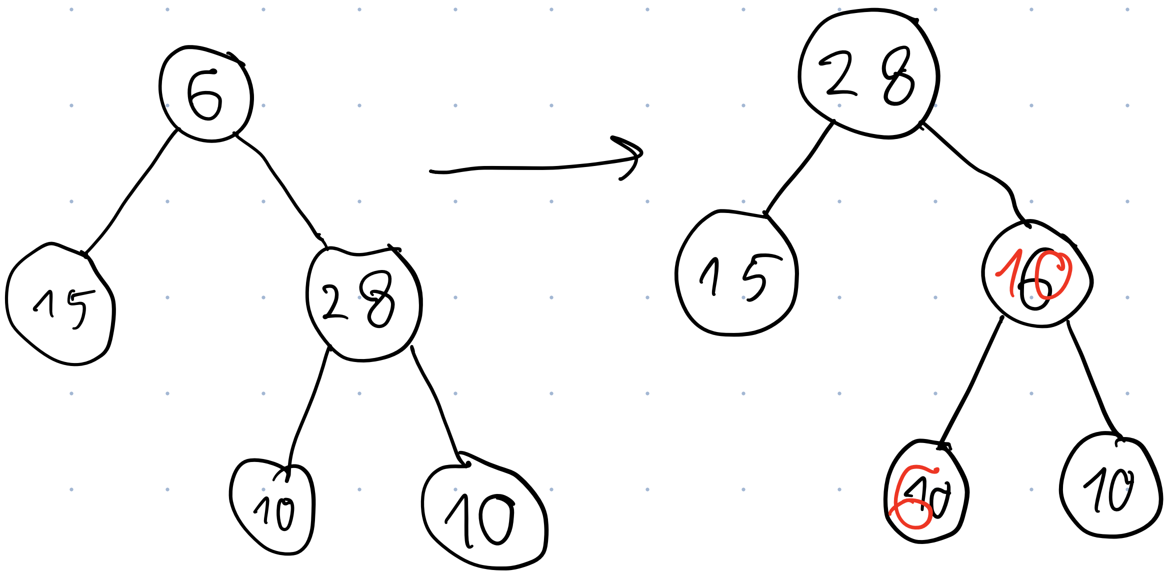
$$\Theta(n)$$

СОРТИРОВКА КУЧЕЙ (HEAP sort)



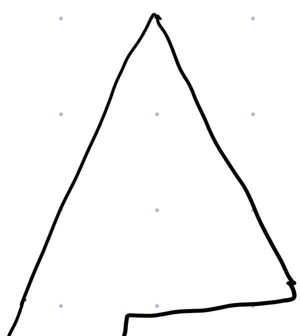
цикл

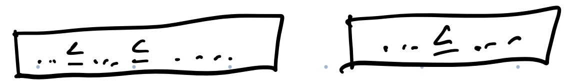
1. ИЗВЛЕК. max
   ЗАПИС. В ОТВЕТ
2. ВОССТ. СВ-ВО КУЧИ
   а) ПОМЕЩ. ПОСЛЕДН. ЭЛ. В КОРЕНЬ
   б) heapify ДЛЯ КОРНЯ



heapify $O(\log n)$

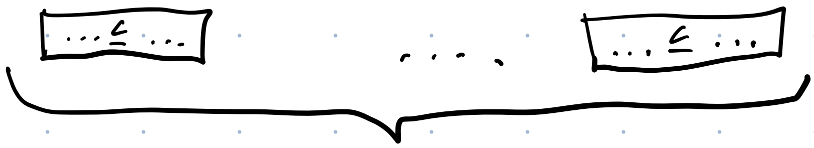
$n \cdot \log n = \Theta(n \log n)$





merge

merge_3

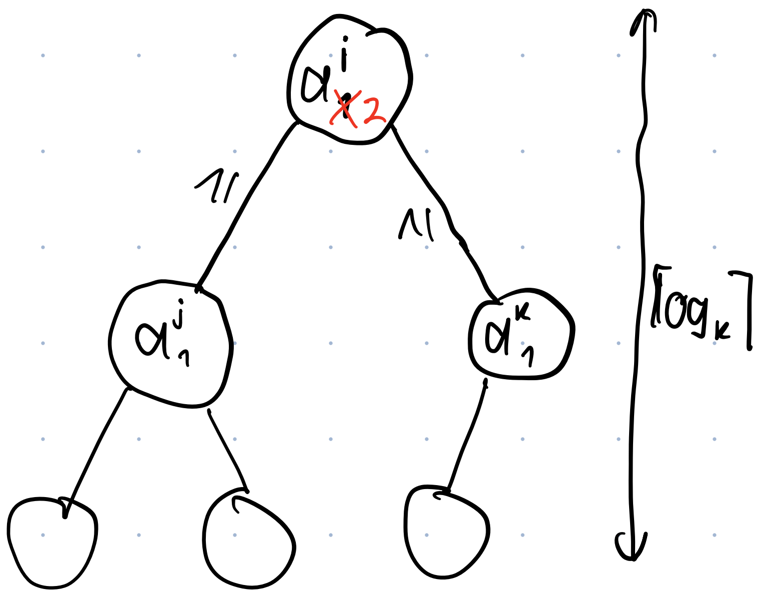


к массивов

всего n элементов

Слияние к отсорт. массивов

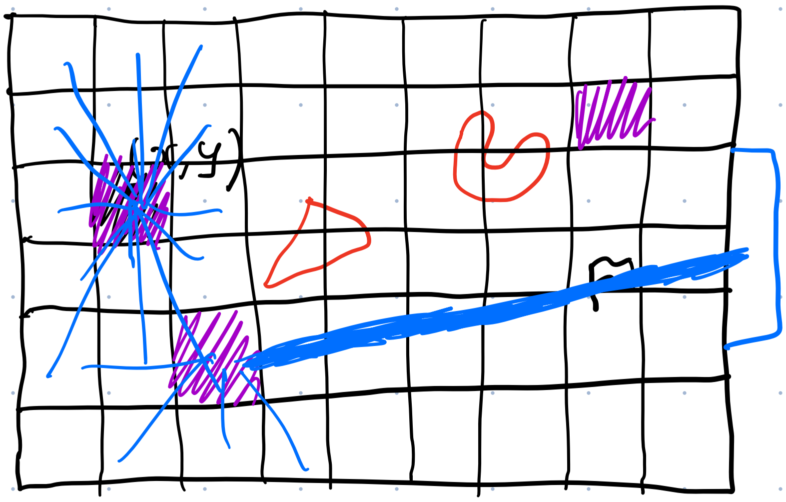
куча на массивах



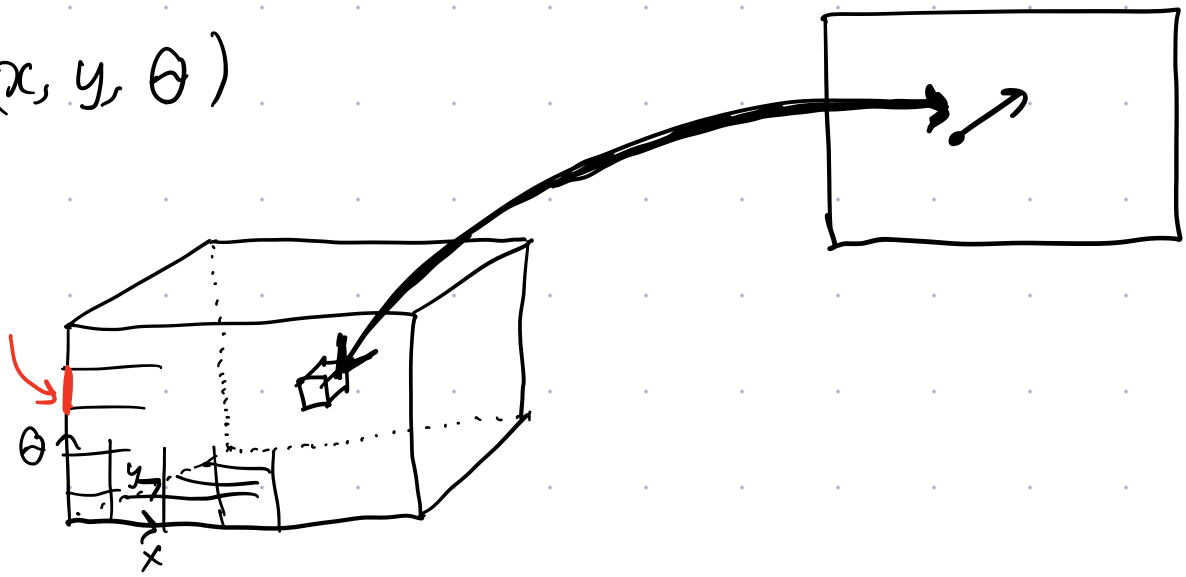
$O(n \log k)$

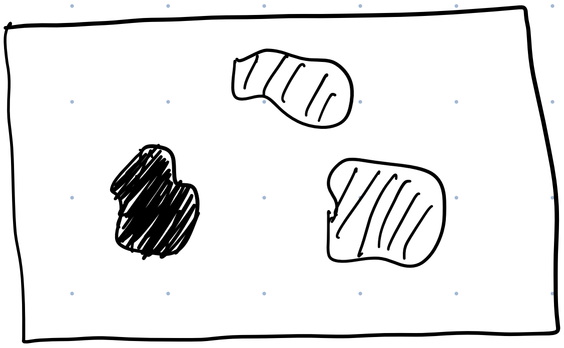
$k = \sqrt{n}$????

$n \log n^{\frac{1}{2}} = \frac{n \log n}{2}$



(x, y, θ)



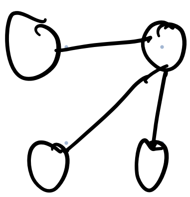


$$G = (V, E)$$

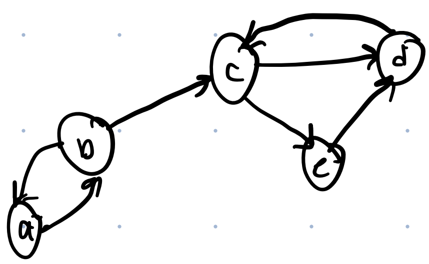
vertices edges

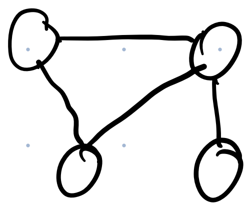
кол-во верш. $|V|$

- связные и несвязные

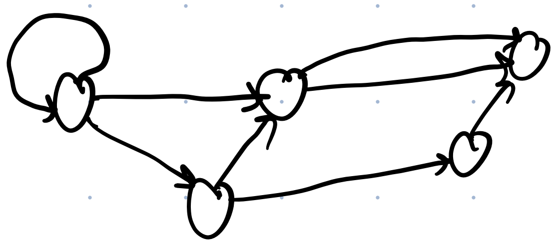


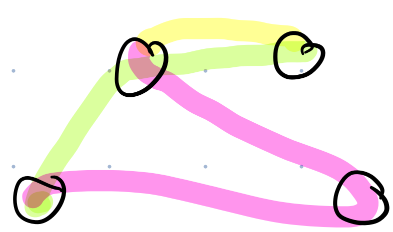
- ориентированные и неориент.



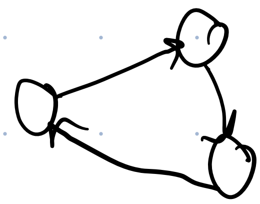


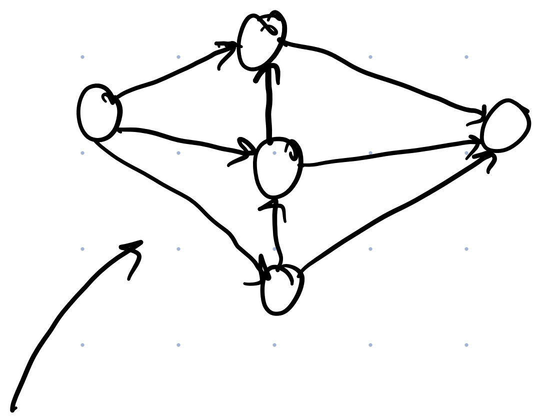
- рассм. графы без петель и кратн. рёбер





- бывают циклические и ациклические



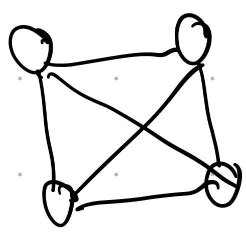


ориентированный ацикл. граф

oriented acyclic graph (DAG)

- сложн. измеряем по $|V|$, $|E|$

$|E| = O(|V|^2)$ для гр. без пет. и кр. рёб.



$$|V| = 4$$

$$|E| = \frac{|V|(|V|-1)}{2}$$

- бывают плотные и разреженные

$O(|V|\log(|E|))$ $\quad$ $O(|V|\log(|V|^2))$

$O(|V||E|)$

e_{ij}

- как хранить?

v_1 : [v_{15}, v_6, ...]

v_2 : [...]

... .

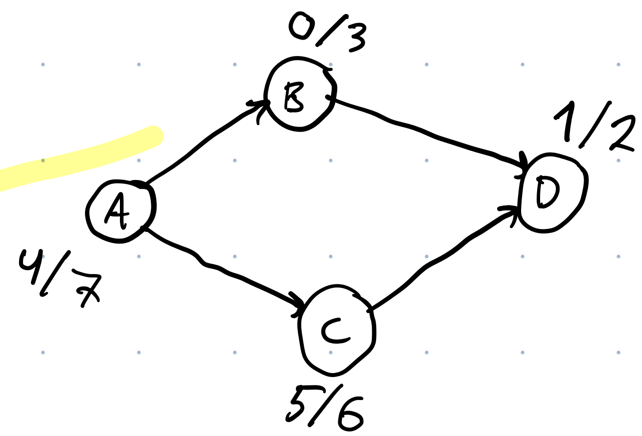
$$\begin{array}{cc} & j \\ i & \begin{pmatrix} \ddots & & \boxed{1} \\ & \ddots & \\ \boxed{-1} & & \ddots \end{pmatrix} \end{array}$$

- вход. степень (исход. степень)
- def. сток – верш., в кот. ∃ путь из любой

Поиск в глубину (Depth First search, DFS)

visited

1	2	...	|V|-1	|V|
F	F	...	F	F



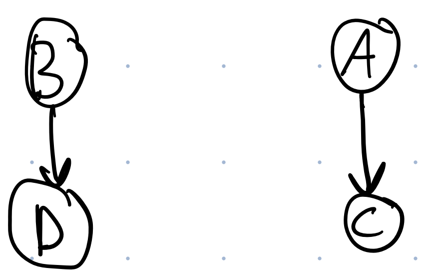

```
def dfs(v):
    visited[v] = True

    for e_vu in E:
        if (visited[u] == False):
            dfs(u)
```

"содержательная часть"

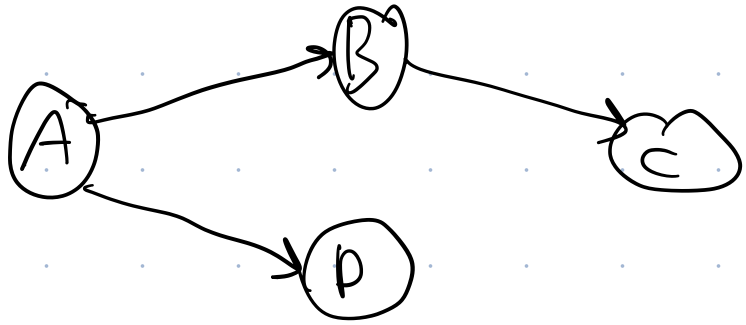
def. дерево – связн. граф без циклов

def. лес – мн-во деревьев

Лес поиска в глубину:



- те же вершины
- подмн-во рёбер



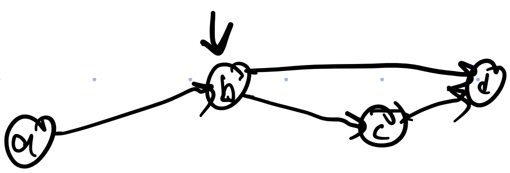
Время работы

v $O(1)$

e $O(1)$

$O(|V|+|E|)$

Можно использ. стек



[ ]

[b]

[b c]

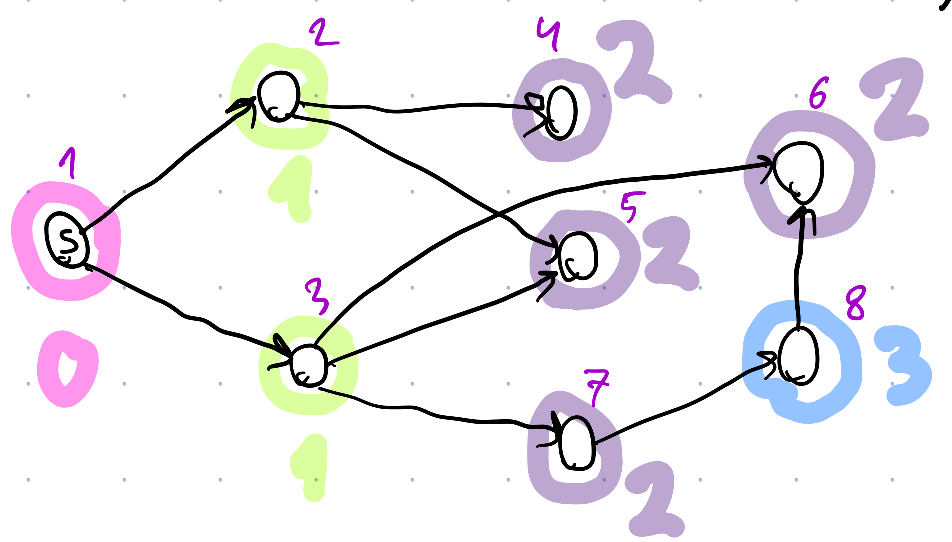
[b c d]

[b c]

[b]

[ ]

Поиск в ширину
(Breadth first search, BFS)



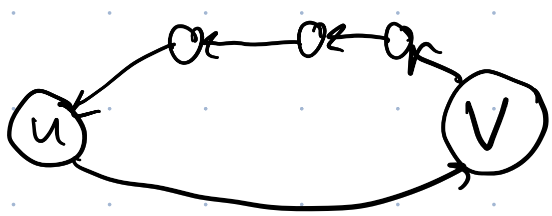
def. Рёберная длина пути $u \to \ldots \to v$ — это кол-во рёбер в пути

def. Рёберное расст. от u до v — min. рёб. длина пути (кратч. рёберный путь)

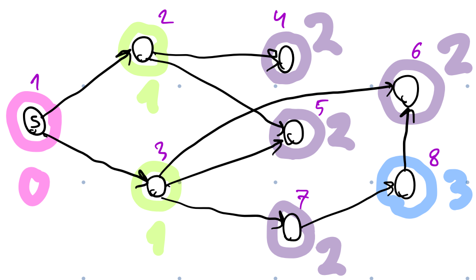
def. Длина пути $u \to \ldots \to v$ — это Σ весов рёбер

$$u \xrightarrow{3} a \xrightarrow{-5} v$$

Дл. пути $= -2$



ИСПОЛЬЗУЕТ ОЧЕРЕДЬ




```
def bfs(u):
    q.push(u)

    while q.not_empty():
        v = q.pop()
        //ОБРАБ. ВЕРШ.
        visited[v] = true
        for e_vs in E:
            if (visited[s] == False):
                q.push(s)
```

[ ]
[s]
[, 2, 3]
[, , 3, 4, 5]
[, , , 4, 5, 6, 7]
[, , , , 5, 6, 7]
[, , , , , 6, 7]
[, , , , , , 7]
[, , , , , , , 8]
[, , , , , , , ,]

$O(|V| + |E|)$