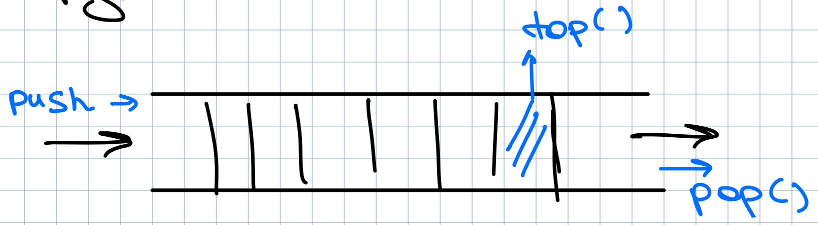
(I) Вспомним

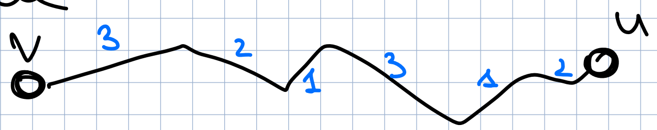
1) Очередь   Queue



FIFO

- push(v)
- top()
- pop()
- is_empty() — True / False

2) Графы   расстояние



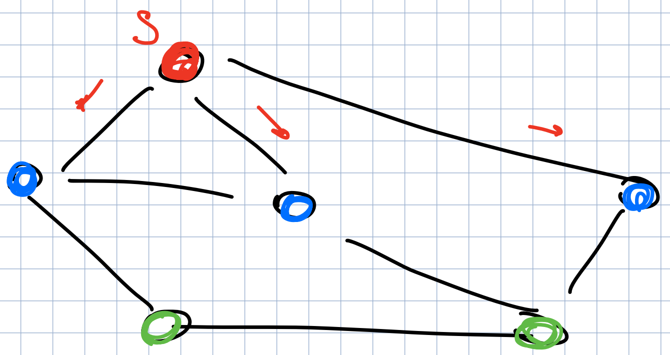
длина пути

distance(v,u) → расстояние от v до u
"минимальное"

(II) BFS (breadth-first search)

поиск в ширину
(обход)

Идея:

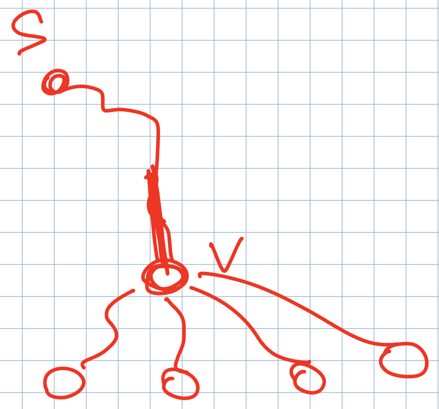


стартовая вершина

dist[k] = ... кратчайшие расстояние от вершины s до k

Реализация

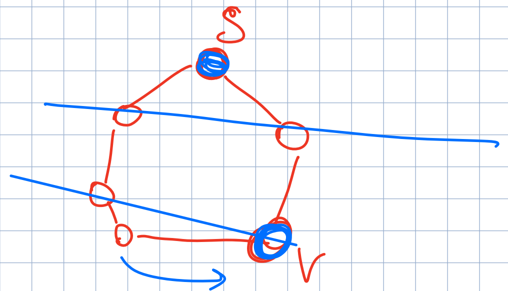
```
dist = [+∞, +∞, .... +∞]
dist[s] = 0
queue = Queue()   // очередь
queue.push(s)
while not queue.isempty()
        v = queue.top()
        queue.pop()   // удалить из очереди
```



```
ch = G.get_children(v)
for i = 0 ... ch.size()
    to = ch[i]            // ребенок
    if dist[to] = $+\infty$:
        dist[to] = dist[v] + 1
        queue.push(to)
        prev[to] = v      // восстановление пути.
```

Асимптотика $\underline{O}(N + M)$

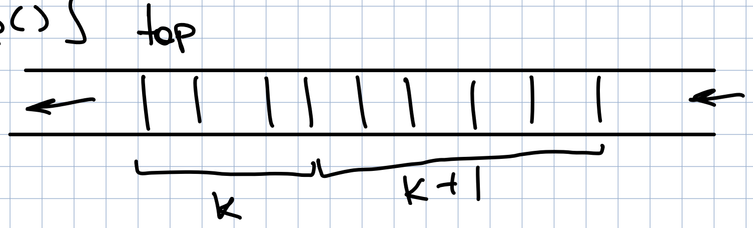
вершин ребра



Ⅲ Корректность BFS

1) $k = $ dist[queue.top()]

queue выглядит

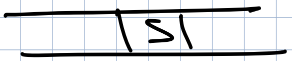


2) все v т.ч. distance$(s,v) \leq k$ имеют dist[v] = distance(s,v)

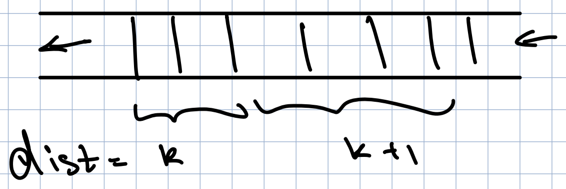
3) Любое обновление dist[v] корректно

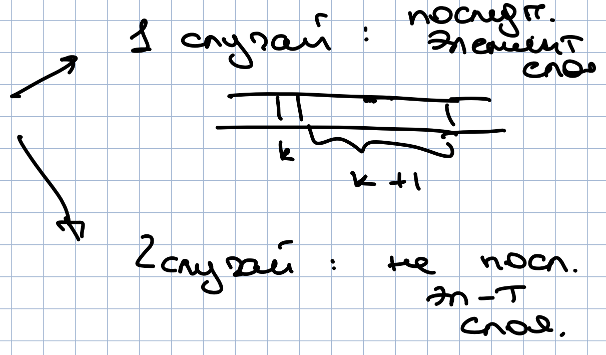
Д-во (индукция по # итераций while)

База: dist[s] = 0



Шаг:

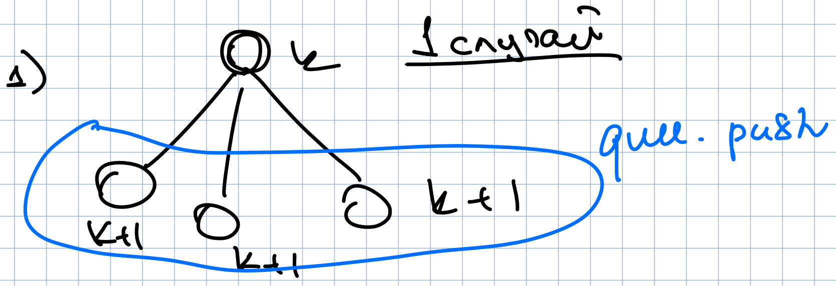


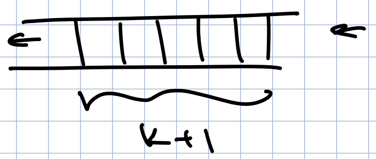


1 случай: последн. элемент слоя

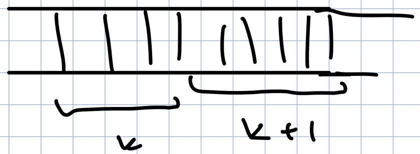
2 случай: не посл. эл-т слоя.

1)





2случай



2) ∀v, distanse(s,v) < k dist[v] → корректен

2 случай: k не изменилось → теперь все тоже ок

1 случай $\quad k = k+1$

$\forall v \quad distance(s,v) \le k+1$

$\swarrow \qquad \searrow$

$distance(s,v) \le k$ □

$distance(s,v) = k+1$ из св-ва 1 очереди. □

(IV) 1-k BFS

веса рёбер $w \in [1, \dots k]$
$w \in \mathbb{Z}$

__Замечание:__ максимальное "кратчайшее расстояние" $< N * k$

$\uparrow$
# вершина

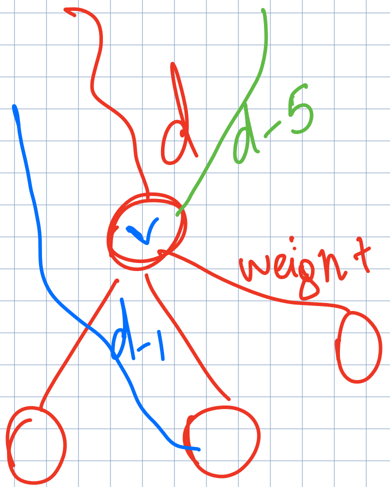
```
dist = [+∞, +∞, +∞, ... +∞]
Queue  queues[N*k]    // В очереди queues[d] хранятся верш. с длиной пути d

queues[0].push(s)
dist[s] = 0
```

```
for d = 0 ... N*k
    while not queues[d].is_empty():
        v = queues[d].top()
        queues[d].pop()
        if dist[v] < d:          ] уже обработали на пред. итер.
            continue

        ch = G.get_children(v)
        for i = 0... ch.size:
            to = ch[i]  // [ to.weight
                        //   to.vertex ]

            if dist[to.vertex] > dist[v] + to.weight:
                dist[to.vertex] = dist[v] + to.weight
                queues[dist[to.vertex]].push(to.vertex)
```

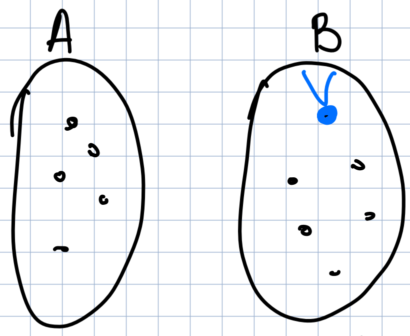


Асимптотика: $O(N*k + M)$

(V) Алгоритм Дейкстры

любые неотрицательные ребра. [от S до всех]

Идея



used = True

100%
все известно
distance

used = False

не уверены
в dist

из части B
(непосещенных)
выбираю вершину v,
с min dist[v]

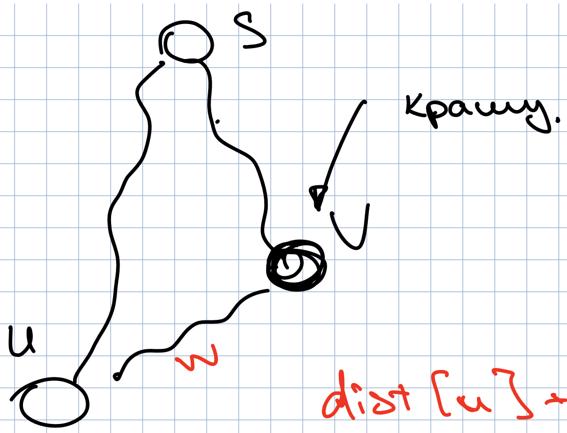
Асимптотика : $\underline{O}(N * N + M)$

(VI) Реализация Дейкстры

dist = [+∞, ... +∞] // размера N
used = [false, ... false]

```
dist [s] = 0
for  i = 0 ....  N :
    v = -1
    поиск минимума
    for  u = 0...  N :
        если  not  used [u]  and
              ( v = -1  or
                dist[u] < dist[v] ) :
            v = u
    if  dist [v] = +∞ :
        break
    used [v] = TRUE
    ch = G.get.children(v)
    for  i = 0...  ch.size() :
        to = ch [i]
        dist [to.vertex] = min(dist[to.vertex],
                               dist [v] + to.weight)
```

Корректность



$dist[u] + w < dist[v]$