Алгоритм Дейкстры

План:
1) Напоминание дистур и опрос
2) Повторение сканлайн, 2 указателя + тест
3) Графы повторение
4) Дейкстра. Простая версия
5) Дейкстра. Сложная версия
6) Восстановление пути

1 Время проведения 15:00 23.04 — 21:00 $\boxed{03.05}$!

2 Сканлайн: Разбиваем на события и „сканируем"
Используется сортировка N + M чисел
$O((N+M)\log(N+M))$ $O(\cancel{N+M})$
Как бы декомпозируем задачу, работать с целыми отрезками тяжело

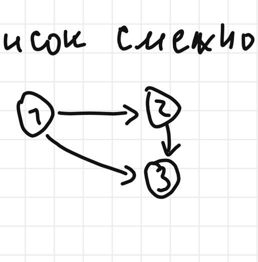
2 Указателя: Пользуемся $[l,r]$ - Ok $\Rightarrow$ $[l,r+1]$ - Ok
$[l,r]$ - Не Ok $\Rightarrow$ $[l+1,r]$ - Не Ok
Асимптотика $O(N)$

2) $[1,4]$, $[5,7]$, $[-1,10]$, $[5,5]$ $-2, 5, 7$
(1, B), (4, E), (5, B), (7, E), (-1, B), (10, E), (5, B), (5, E)
(-2, Q), (5, Q), (7, Q)
B < Q < E
(-2, Q) (-1, B) (1, B), (4, E) (5, B) (5, B), (5, Q) (5, E)
(7, Q) (7, E) (10, E)

5) -1 2 3 8 9 10 14 n d=5
0 1 2 3 4 5 6 7
3 3 4 6 7 7 7 — Ответ

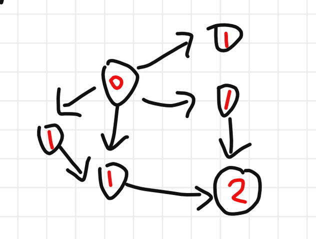
3 Графы

g - список смежности   g[v] - список соседей v



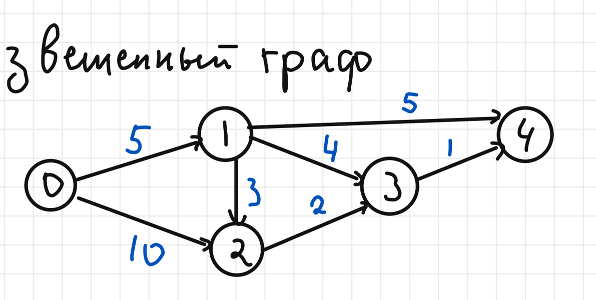
g[1] = {2, 3}
g[2] = {3}
g[3] = ∅ — пустой

Ор. граф (без весов) с помощью BFS найти от заданной до всех расстояние.



Расстояние = кол-во рёбер.
$O(N+M)$ — легко

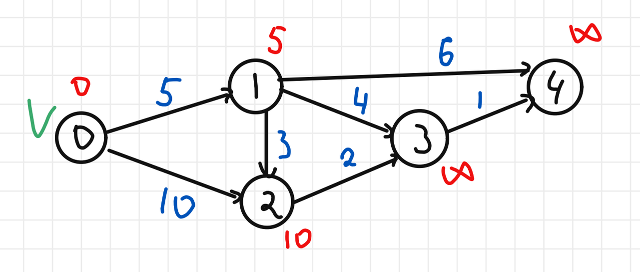
4 Взвешенный граф



Расстояние = сумма весов на пути

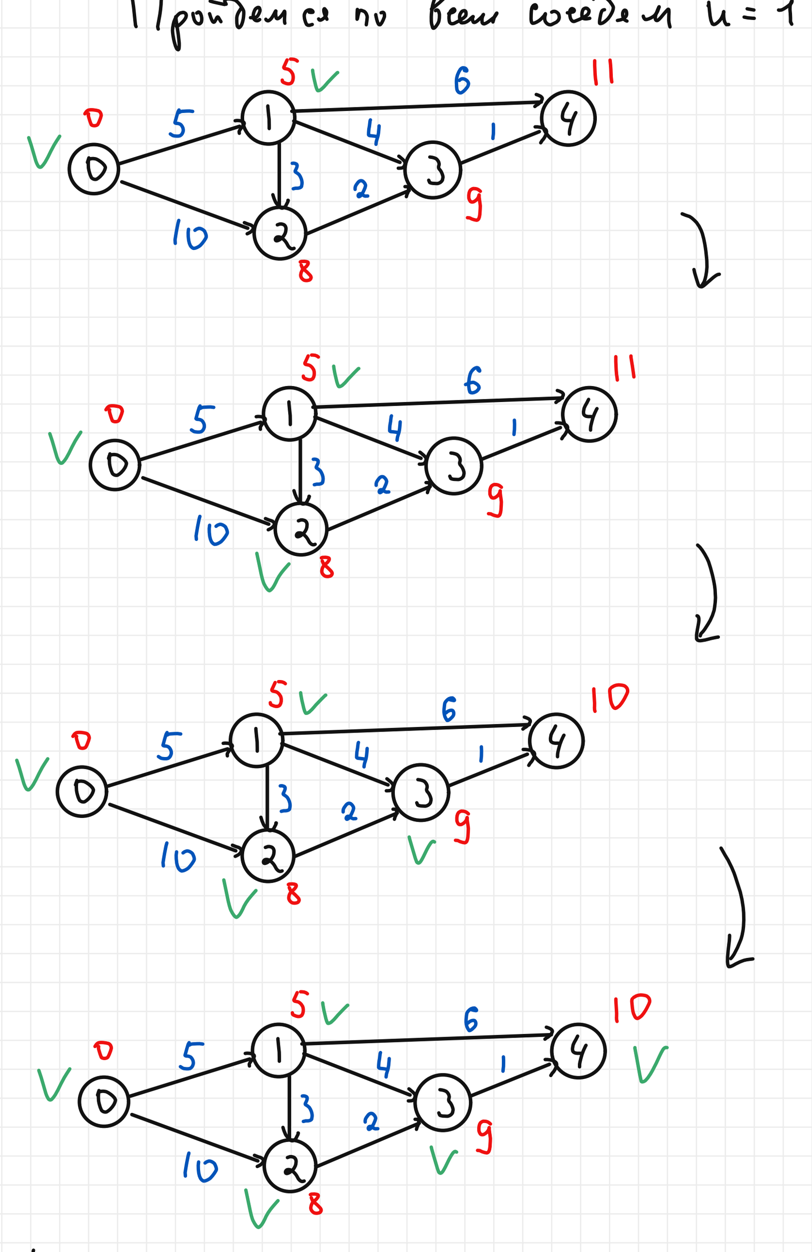
Хотим найти расстояние от стартовой до всех остальных. Веса рёбер ≥ 0. (Нельзя отрицательные числа на рёбрах)

Алгоритм:
S - старт
dist[S] = 0
1) Пройдёмся по всем соседям s: пересчитаем dist —
$dist[v] = \min(dist[v], \underbrace{dist[s]}_{=0} + w)$
v — сосед s; w — вес ребра из s в v

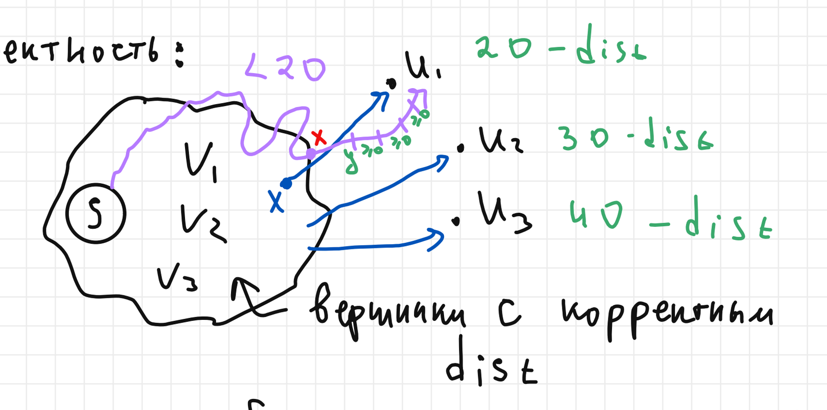


— dist
S = 0   Рассмотрим S = 0

2) Из ещё не рассмотренных берём ту, у которой dist минимален
Пройдёмся по всем соседям u = 1   dist[v] = min(.....)

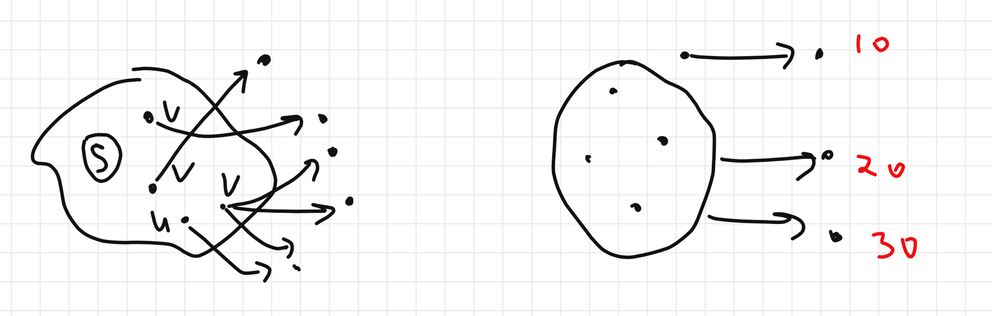


Корректность:

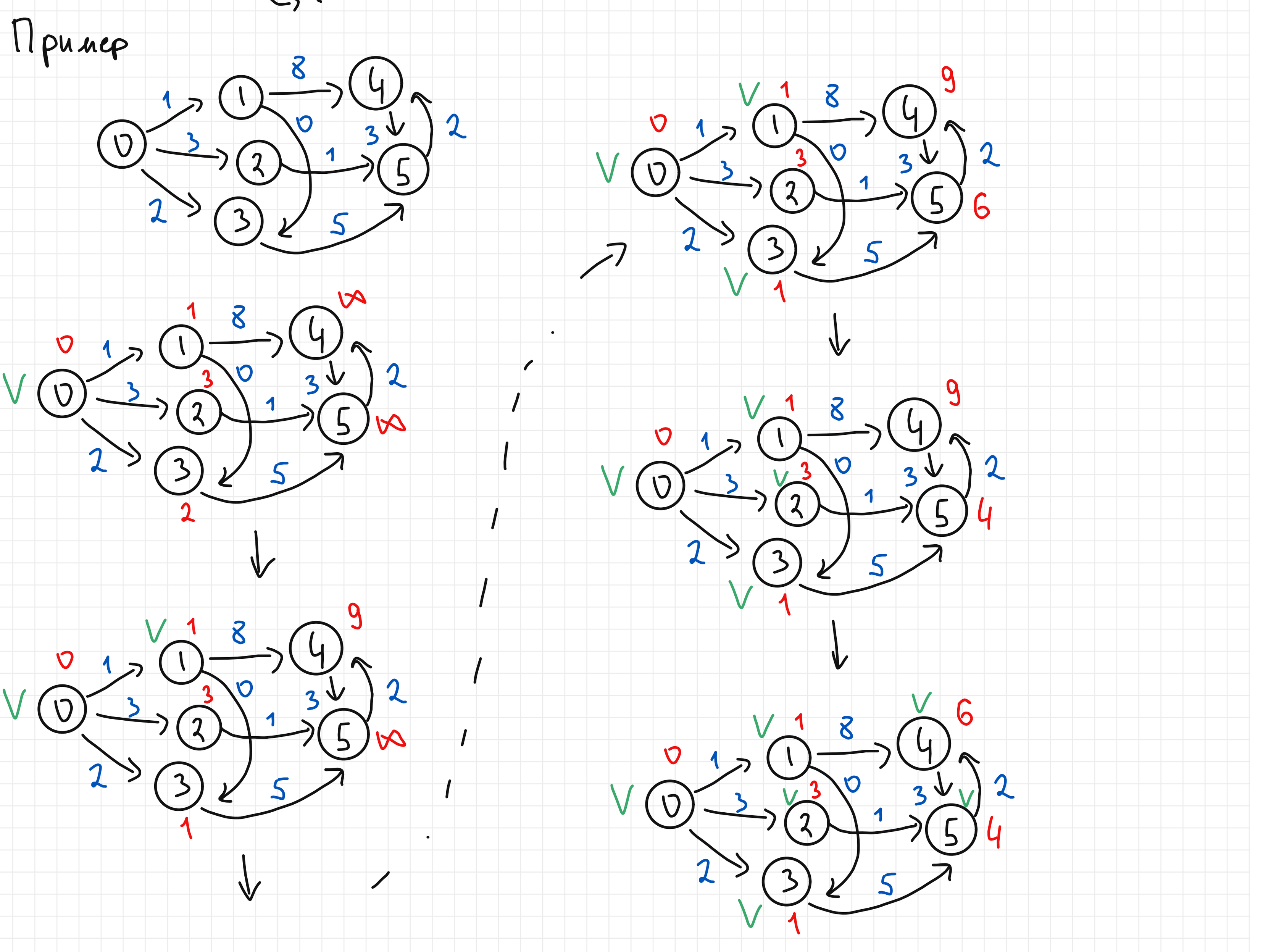


На шаге 1: облачко состоит только из s — всё хорошо

u_1 добавляем в облачко и пересчитываем из неё соседей в алгоритме
Пусть это не так x - последняя вершина из облака на пути
Тогда $dist[y] < dist[u_1]$ — противоречие



Пример



Код:
```
vector<vector<pair<int, int>>> g(n);
vector<int> dist(n, INF);
vector<bool> used(n, false)
s - стартовая вершина
dist[s] = 0
for i = 0...n-1:
    minv = -1
    mind = INF
    for v = 0...n-1:
        if (mind > dist[v] && used[v] == false):
            mind = dist[v]
            minv = v
    if minv == -1: break
    used[minv] = true.
    for (u, w) in g[minv]:
        dist[u] = min(dist[u], mind + w)      # mind = dist[minv]
print(dist)      # dist[u] = INF <=> нет пути в графе
```

Асимптотика: $O(N^2 + M)$

5 set — упорядоченное множ-во   #include <set>
set<int> — .insert(x) — добавить x
.erase(x) — удалить x
} $\log_2$ от размера
*name.begin() — взять минимум — $O(1)$

Код:
```
vector<vector<pair<int, int>>> g(n);
vector<int> dist(n, INF);
s - стартовая вершина
dist[s] = 0
set<pair<int, int>> q.
q.insert({dist[s], s})
while (q.size()):                      — N раз выполнится
    kek = *q.begin()
    mind = kek.first
    minv = kek.second
    q.erase(q.begin())
    for (u, w) in g[minv]:             — Суммарно M раз
        if (dist[u] > mind + w):
            q.erase({dist[u], u})      log2|q| = log2 N
            dist[u] = mind + w
            q.insert({dist[u], u})     log2
print(dist)
```

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

6

Код:
```
vector<vector<pair<int, int>>> g(n);  vector<int> pred(n, -1)
vector<int> dist(n, INF);
s - стартовая вершина
dist[s] = 0
set<pair<int, int>> q.
q.insert({dist[s], s})
while (q.size()):
    kek = *q.begin()
    mind = kek.first
    minv = kek.second
    q.erase(q.begin())
    for (u, w) in g[minv]:
        if (dist[u] > mind + w):
            q.erase({dist[u], u})
            dist[u] = mind + w
            q.insert({dist[u], u})
            pred[u] = minv
```

$s \xrightarrow{mind} minv \xrightarrow{w} u$

```
f - финиш
vector<int> ans
cur = f
while (cur != s):
    ans.push(cur)
    cur = pred[cur]
ans.push(s)
reverse(ans.begin(), ans.end())
print(ans)
```

$\underset{=s}{0} \to 1 \to \underset{=f}{2}$
pred[2] = 1
pred[1] = 0
ans = {2, 1, 0}
⇒ нужно развернуть