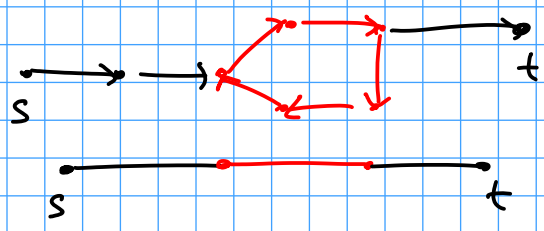
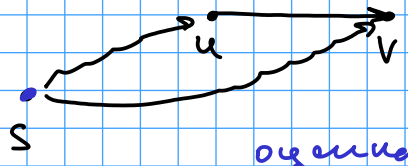
Кратчайшие пути в графах с отриц. рёбр.

Утв Если в графе есть цикл отрицательного веса (или отрицательное ребро в неор. графе), то кратчайшие пути не опр.



$$d(s,t) = -\infty$$

Операция релаксации ребра



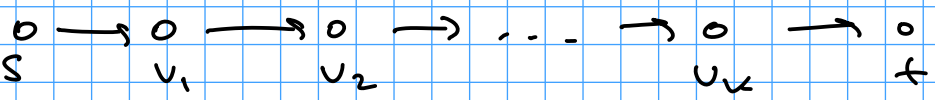
$$d[v] = \min\{d[v], d[u] + w(u,v)\}$$

оценка $dist(s,v)$

Утв. $d[v] \geq dist(s,v)$

т.е. релаксация ребра не ухудшает расстояние

Утв. Пусть $s \to v_1 \to v_2 \to \dots \to v_k \to t$ — кратчайший путь от s до t.
Давайте последовательно сделаем релакс. для рёбер $(s,v_1), (v_1,v_2), \dots, (v_{k-1},v_k), (v_k,t)$
$\Rightarrow$ вычислим кратчайший путь от s до t (т.е. вычислим расстояние для $v_1, v_2, \dots, v_k$ и t)

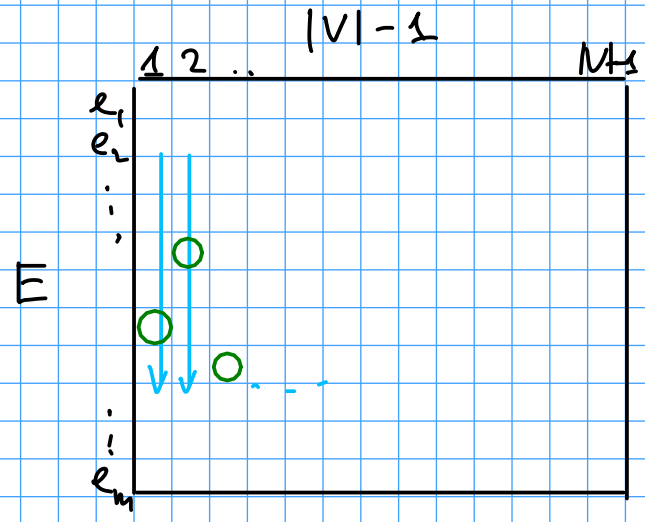


Идея 1. Давайте перебирём все последовательности из $|V|-1$ ребра

$$|E|! / (|E| - |V| + 1)! \sim |E|^{|V|}$$

Слишком много!

Идея 2. Давайте выберем посл-ть релаксаций такую, что $\forall$ последующим $|V|-1$ ребро является её подпоследовательностью.




```
Bellman-Ford((V, E), s):
    for v ∈ V:
        d[v] = +∞
        prev[v] = 0
    d[s] = 0
    for i = 1 to |V|-1:
        for (u, v) ∈ E:
            d[v] = min(d[v], d[u] + w(u,v))
            if d[v] changed: prev[v] = u
```

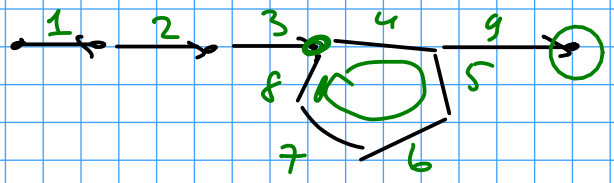
Время: $O(V \cdot E)$

Утв: Если на какой-то итерации расстояния не улучшились (массив d не изменился) $\Rightarrow$ можно остановить алгоритм

Утв: Если допустить алг. Беллмана-Форда на $|V|$ итерацией и на последней итерации что-то улучшится $\Leftrightarrow$ в графе есть отрицательный цикл.

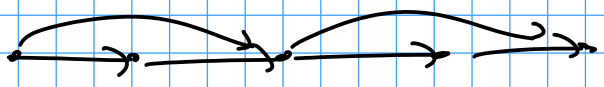
1) ⇒ Если нет отриц. циклов ⇒ остановится не |V|-1 итерацию

⇐



4

Напоминание: В ор. графах без циклов мы умеем искать кратчайшие пути за $O(V+E)$.



Кратчайшие пути м/у всеми парами вершин

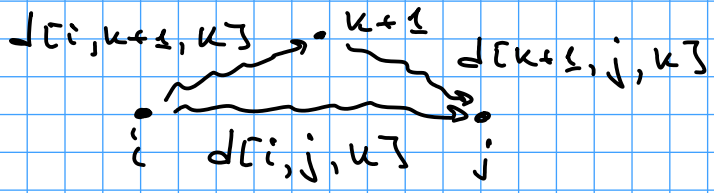
Алгоритм Флойда-Уоршалла

Динамическое программирование:

1. Подзадача:

$d[i,j,k]$ = dist(i,j), если промежуточ. вершины из множества $\{1,2,\dots k\}$

2. $d[i,j,k+1] = \min\{ d[i,j,k],$ // k+1 не помогла
$d[i,k+1,k] + d[k+1,j,k] \}$ // k+1 помогла



$d[i,j,0] = w(i,j)$

3. Порядок: for k { for i { for j } }

Итог: Достаточно двухмерного массива d для реализации того же алгоритма.

```
Floyd-Warshall (⟨V, E⟩):
    for i = 1 to |V|:
        for j = 1 to |V|:
            d[i, j] = w(i, j)
            if d[i, j] < +∞: prev[i, j] = i
            if i = j:
                d[i, j] = 0
    for k = 1 to |V|:
        for i = 1 to |V|:
            for j = 1 to |V|:
                if d[i, j] > d[i, k] + d[k, j]:
                    d[i, j] = d[i, k] + d[k, j]
                    prev[i, j] = prev[k, j]
```

Время $O(V^3)$

Память $O(V^2)$

Итог: Работает с отрицательными весами.