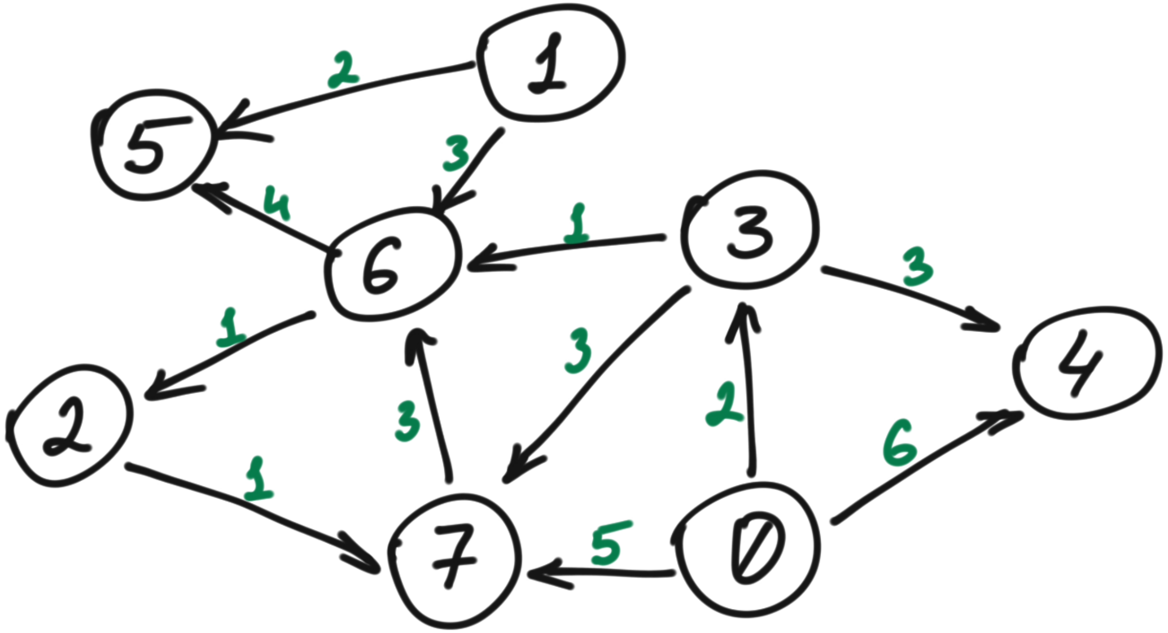
Кратчайшие пути из одной вершины

Shortest Paths Single Source

Постановка задачи

Дан граф $G = (V, E)$ (ориентированный или неориентированный), на котором задана весовая функция $w : E \to \mathbb{R}_+$.
Также дана вершина s - вершина, из которой нужно искать кратчайшие пути.

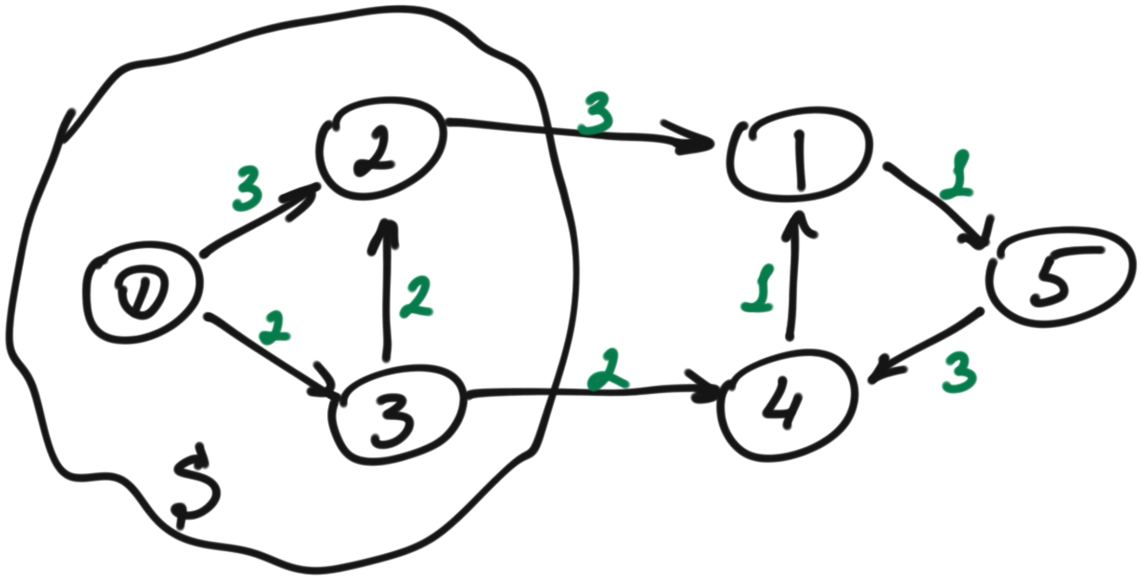
Для любой $v \in V$ найти $\rho(s, v) = \min\limits_{P} \sum\limits_{e \in P(s,v)} w(e)$, где $P(s, v)$ - путь из s в v.



Алгоритм Дейкстры

В любой момент времени будем поддерживать два множества: S - множество вершин, до которых уже найден кратчайший путь, $U = V \setminus S$ - вершины, кратчайший путь до которых еще неизвестен.

Также будем поддерживать массив d: $d[v]$ хранит длину кратчайшего пути до v, проходящего *только по вершинам из* S.



Алгоритм Дейкстры

0. Инициализация: $S = \{s\}, \forall v \neq s : d[v] = w(s, v), d[s] = 0$.

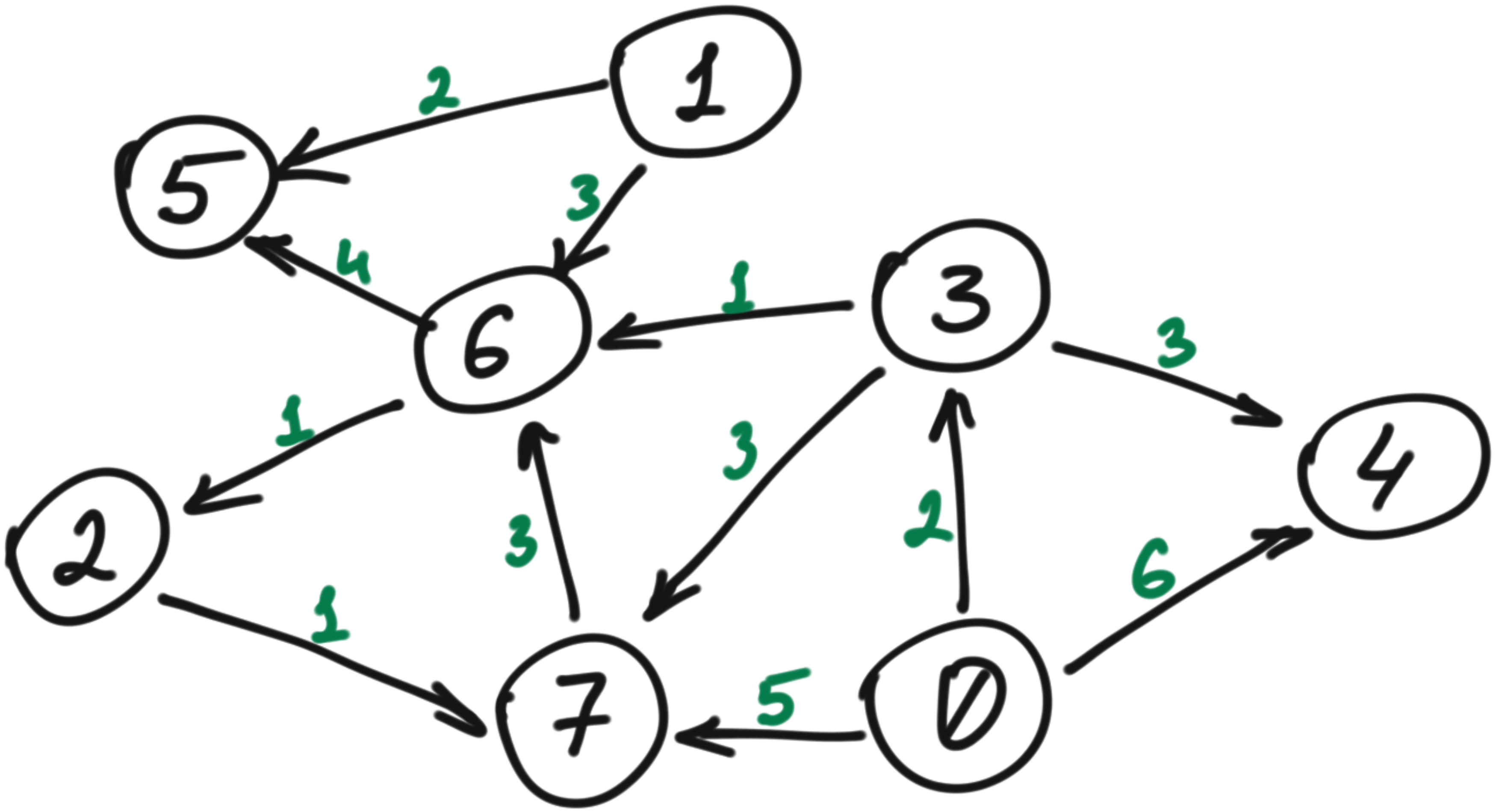
1. Ищем $u = \arg\min\limits_{v \notin S} d[v]$.

2. Добавляем u в S.

3. Обновляем $\forall v : d[v] = min(d[v], d[u] + w(u, v))$.

4. Повторяем 1-3 пока $S \neq V$.

Пример



Алгоритм Дейкстры: корректность

Теорема (Dijkstra E.W., 1959).

Если ребра графа неотрицательны, то по завершении работы алгоритма $\forall v : d[v] = \rho(s, v)$

Алгоритм Дейкстры: корректность

Теорема (Dijkstra E.W., 1959).

Если ребра графа неотрицательны, то по завершении работы алгоритма $\forall v : d[v] = \rho(s, v)$

Доказательство. (индукция по числу итераций)

База. $d[s] = 0 = \rho(s, s)$

Переход. Пусть $\forall v \in S : d[v] = \rho(s, v)$. Рассмотрим КП из s до u (из шага 1). Он пересекает разрез (S, U) по какому-то ребру xy:

$\rho(s, y) = \rho(s, x) + w(x, y) = d[x] + w(x, y)$, так как $x \in S$.

При добавлении x в S вычисляли $d[y] = \min(d_{old}[y], d[x] + w(x, y))$, поэтому $\rho(s, y) = d[y]$ (использовали очевидное свойство $\forall v : d[v] \geq \rho(s, v)$).

Тогда получаем $\rho(s, u) \leq d[u] \overset{\text{шаг1}}{\leq} d[y] = \rho(s, y) \overset{*}{\leq} \rho(s, u)$.

*Так как $P(s, y)$ подпуть $P(s, u)$ и веса ребер ≥ 0. То есть $d[u] = \rho(s, u)$ ■

Алгоритм Дейкстры: реализация с пирамидой

Алгоритм практически не отличается от алгоритма Прима

```
def Dijkstra(G, s):
  dist = [inf, ..., s: 0, ..., inf]
  prev = [None, ..., None]  # предок в минимальном пути
  heap.insert(..., (dist[v], v), ...)  # Все вершины из U в пирамиде
  while heap is not empty:
    v = heap.ExtractMin()
    for u in G.neighbors(v):
      if u in heap and dist[u] > dist[v] + w(v, u):
        prev[u] = v
        dist[u] = dist[v] + w(v, u)
        heap.DecreaseKey(u, dist[v] + w(v, u))
```

При использовании бинарной пирамиды $O(E \log V)$

При использовании фибоначчиевой пирамиды $O(E + V \log V)$

Алгоритм Дейкстры: реализация с массивом

Алгоритм практически не отличается от алгоритма Прима

```
def Dijkstra(G, s):
  dist = [inf, ..., s: 0, ..., inf]
  prev = [None, ..., None]
  S = {s}
  while |S| != |V|:
    v = argmin(dist)
    dist[v] = inf
    S.insert(v)
    for (v, u) in G.neighbors(v):
      if u not in S and dist[u] > dist[v] + w(v, u):
        prev[u] = v
        dist[u] = dist[v] + w(v, u)
```

$O(E + V^2)$

Алгоритм Дейкстры: сравнение реализаций

Граф/Алгоритм	Бинарная пирамида	Фибоначчиева пирамида	Массив
Разреженый ($E \sim V$)	$O(V \log V)$	$O(V \log V)$	$O(V^2)$
Плотный ($E \sim V^2$)	$O(V^2 \log V)$	$O(V^2)$	$O(V^2)$

Вывод: для разреженных графов лучше выбирать бинарную пирамиду, для плотных - массив.

Фибоначчиева пирамида на практике никогда не лучше (большая константа).

Отрицательные ребра

Отрицательные ребра: проблема

- Снимем ограничение на неотрицательность ребер, то есть $w : E \to \mathbb{R}$.
- Как это повлияет на постановку задачи поиска кратчайших путей?

Отрицательные ребра: проблема

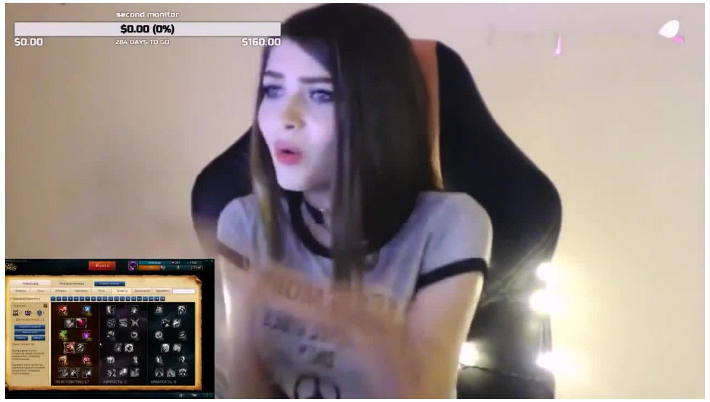
- Снимем ограничение на неотрицательность ребер, то есть $w : E \to \mathbb{R}$.
- Как это повлияет на постановку задачи поиска кратчайших путей?
- Задача может стать некорректной из-за возможного наличия циклов отрицательного веса.

Отрицательные ребра: проблема

- Снимем ограничение на неотрицательность ребер, то есть $w : E \to \mathbb{R}$.
- Как это повлияет на постановку задачи поиска кратчайших путей?
- Задача может стать некорректной из-за возможного наличия циклов отрицательного веса.
- Проблему можно обойти, если искать *простые пути*, но есть один нюанс...

Отрицательные ребра: проблема

- Снимем ограничение на неотрицательность ребер, то есть $w : E \to \mathbb{R}$.
- Как это повлияет на постановку задачи поиска кратчайших путей?
- Задача может стать некорректной из-за возможного наличия циклов отрицательного веса.
- Проблему можно обойти, если искать *простые пути*, но есть один нюанс...
- Задача поиска кратчайшего *простого* пути NP-сложная.



Отрицательные ребра: проблема

Утверждение.

Задача поиска кратчайших путей корректна $\Leftrightarrow$ *В графе нет циклов отрицательного веса.*

Доказательство.

$\Rightarrow$ Обсудили

$\Leftarrow$ Кратчайший путь может быть только простым. Это верно, так как если путь не простой, то в нем есть цикл. А так как этот цикл неотрицателен, то можно от него избавится и получить более короткий путь.

Так как число простых путей конечно, то минимальный среди них существует. $\blacksquare$

Алгоритм Форда-Беллмана

Релаксация ребра

Пусть найдены какие-то пути из s до v и u длины $dist[v]$ и $dist[u]$.

Релаксацией ребра vu назовем следующую процедуру:

```
def Relax(v, u):
  if dist[u] > dist[v] + w(v, u):
    dist[u] = dist[v] + w(v, u)
    prev[u] = v
    return True
  return False
```

Алгоритм Форда-Беллмана: идея

Любой кратчайший путь состоит из не более $V-1$ ребра (если нет циклов отрицательного веса), поэтому для нахождения всех путей достаточно $V-1$ раз отрелаксировать все ребра.

Алгоритм Форда-Беллмана

```
def BellmanFord(G, s):
  dist = [inf, ..., s: 0, ... inf]
  prev = [None, ..., None]
  for i from 0 to |V| - 1:  # V - 1 iterations
    for (v, u) in E:
      Relax(v, u)
```

Сложность $O(VE)$

Алгоритм Форда-Беллмана: корректность

Теорема 1 (Bellman R., 1958).

Если в графе G *отсутствуют циклы отрицательного веса, то по завершении работы алгоритма* $\forall v : dist[v] = \rho(s, v)$

Алгоритм Форда-Беллмана: корректность

Теорема 1 (Bellman R., 1958).

Если в графе G отсутствуют циклы отрицательного веса, то по завершении работы алгоритма $\forall v : dist[v] = \rho(s, v)$

Доказательство.

Рассмотрим произвольную вершину v и кратчайший путь до нее $P(s, v) = (v_0 = s, v_1, \dots, v_{n-1}, v_n = v)$. С помощью индукции покажем, что после k-й итерации внешнего цикла $dist[v_k] = \rho(s, v_k)$.

База. $dist[v_0] = dist[s] = 0 = \rho(s, s)$

Переход. Пусть после $k - 1$ итерации $dist[v_{k-1}] = \rho(s, v_{k-1})$. На k-й итерации отрелаксируется ребро (v_{k-1}, v_k), то есть

$dist[v_k] = dist[v_{k-1}] + w(v_{k-1}, v_k) = \rho(s, v_{k-1}) + w(v_{k-1}, v_k) = \rho(s, v_k)$. ■

Цикл отрицательного веса

Как определить наличие отрицательного цикла?

Цикл отрицательного веса

Как определить наличие отрицательного цикла?

Алгоритм Форда-Беллмана умеет и это.

```
def HasNegativeCycle(G, s):
  dist = [inf, ..., s: 0, ... inf]
  prev = [None, ..., None]
  for i from 0 to |V| - 1:  # V - 1 iterations
    for (v, u) in E:
      Relax(v, u)

  for (v, u) in E: # еще раз отрелаксируем все ребра
    if Relax(v, u):
      return True
  return False
```

Цикл отрицательного веса

Теорема 2 (Bellman R., 1958).

Алгоритм корректно определяет наличие/отсутствие цикла отрицательного веса.

Цикл отрицательного веса

Теорема 2 (Bellman R., 1958).
Алгоритм корректно определяет наличие/отсутствие цикла отрицательного веса.

Доказательство.

1. Если отрицательных циклов нет, то все кратчайшие пути уже найдены и новых релаксаций не произойдет.
2. Допустим есть отрицательный цикл $(v_0, v_1, \ldots, v_{n-1}, v_n = v_0)$, но алгоритм об этом не сообщил.
   То есть $\forall k : dist[v_{k+1}] \leq dist[v_k] + w(v_k, v_{k+1})$.
   Просуммируем неравенства: $\sum\limits_{k=0}^{n-1} dist[v_k] \leq \sum\limits_{k=0}^{n-1} dist[v_k] + \sum\limits_{k=0}^{n-1} w(v_k, v_{k+1})$.
   $0 \leq \sum\limits_{k=0}^{n-1} w(v_k, v_{k+1})$, противоречие (цикл-то отрицательный) ■