# Лекция 4

## Кратчайшие пути во взвешенном графе

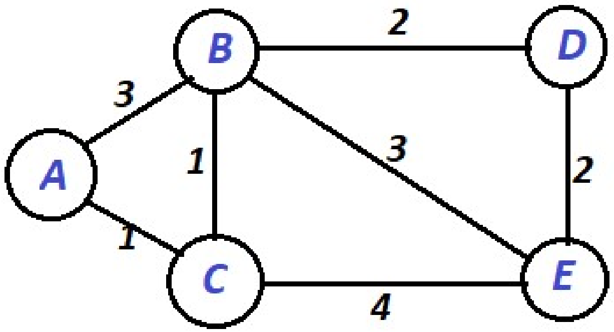
Алгоритм Дейкстры. Ускорение с помощью двоичной кучи. Алгоритм Беллмана -Форда

# Постановка задачи

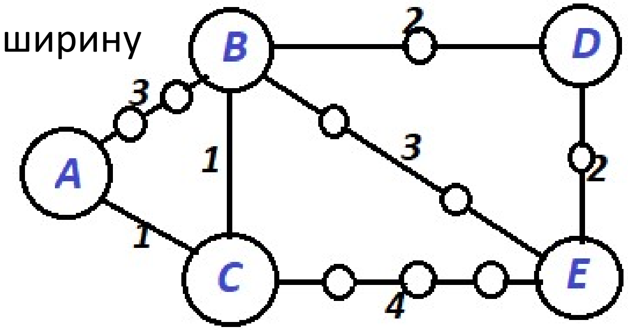
Рассмотрим взвешенный неориентированный граф G(V, E).

Любому ребру $(v, w)$ соответствует неотрицательное число $l_{vw}$ (вес, длина, стоимость). Необходимо найти пути из $s \in V$ во все вершины графа так, чтобы длины этих путей были минимальны.

Пример:




Применим поиск в ширину




$len(A) = 0$
$len(B) = 2$
$len(C) = 1$
$len(D) = 4$
$len(E) = 5$

Взрывной рост размеров графа

# Алгоритм Декстеры

Дейкстрова бальная оценка ребра $(vw)$: $len\ (v) + l_{vw}$

**DIJKSTRA**

**Вход**: ориентированный граф $G = (V, E)$, представленный в виде списков смежности, вершина $s \in V$, длина $l_e \geq 0$ для каждого $e \in E$.

**Постусловие**: для каждой вершины $v$ значение $len(v)$ равно истинному кратчайшему расстоянию $dist\ (s, v)$.

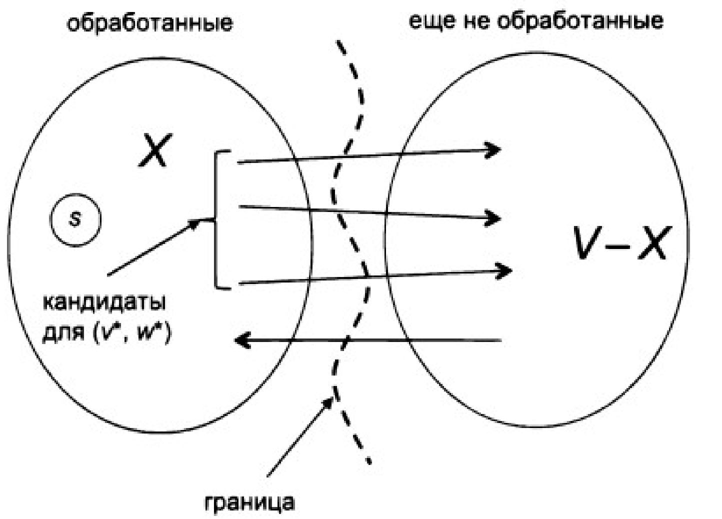
---

`// инициализация`

1) $X := \{s\}$
2) $len(s) := 0$, $len(v) := +\infty$ для каждого $v \neq s$

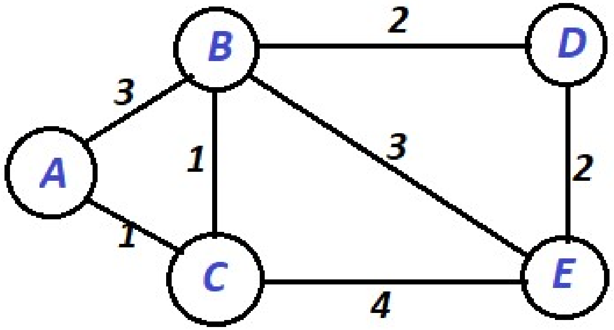
`// главный цикл`

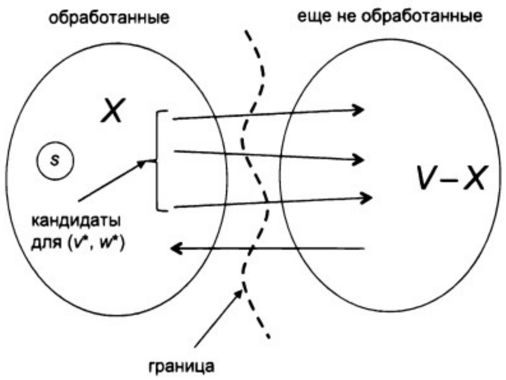
3) **while** существует ребро $(v, w)$, где $v \in X$, $w \notin X$ **do**
4) $(v^*, w^*)$ := такое ребро, которое минимизирует $\boxed{len(v) + l_{vw}}$
5) $\text{prev}[w^*] = v^*$
6) добавить $w^*$ в $X$
7) $len\ (w^*) := len\ (v^*) + l_{v^*w^*}$.




Каждая итерация алгоритма Дейкстры обрабатывает одну новую вершину, голову ребра, переходящего из $X$ в $V - X$

# Продолжение примера



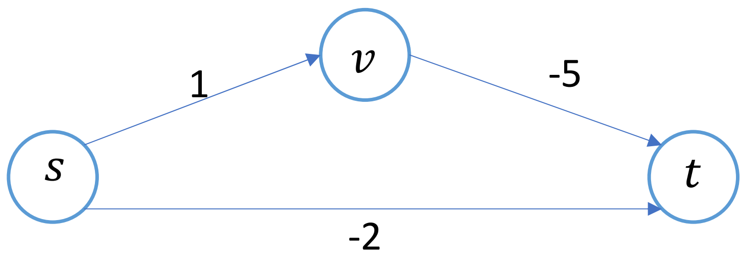




Каждая итерация алгоритма Дейкстры обрабатывает одну новую вершину, голову ребра, переходящего из $X$ в $V - X$

| X | V-X |
|---|---|
| A(0) | B(3), C(1), D(∞), E(∞) |
| A(0), C(1) | B(2), D(∞), E(5) |
| A(0), C(1), B(2) | D(4), E(5) |
| A(0), C(1), B(2), D(4) | E(5) |
| A(0), C(1), B(2), D(4), E(5) | ∅ |

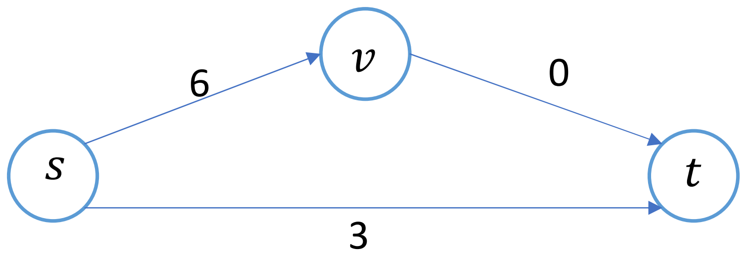
# Когда не работает алгоритм Дейкстры

## Графы с отрицательными весами ребер




По Дейкстре $len(t) = -2$
На самом деле $len(t) = -4$

Прибавим 5 к каждому весу:




$len(t) = 3$ неверно!

Алгоритм Дейкстры не работает на графах с отрицательными весами

# <u>Теор.</u> (Правильность алгоритма Дейкстры)

**Для каждого ориентированного графа G=(V, E), каждой стартовой вершины s каждого варианта неотрицательных реберных весов (длин) по завершении алгоритма**

$$\boldsymbol{len(v) = dist(s,v)\ \forall v \in V.}$$

Доказательство (по индукции) – по числу вершин.

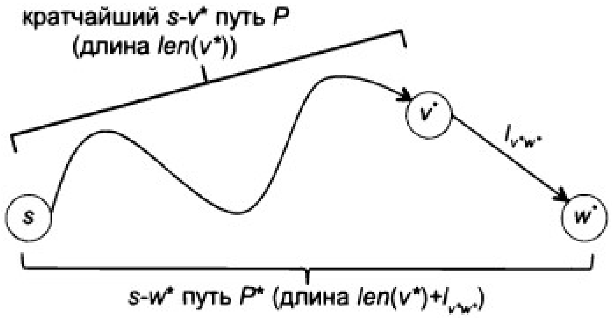
Базовый случай $k = |V| = 1$ – очевидно ($len(s) = 0 = dist(s,s)$).

Пусть для первых $k-1$ вершин утверждение верно. На $k$-ом шаге выбрано ребро $(v^*, w^*)$, $v^* \in X, w^* \in V - X,\ \ len(w^*) = len(v^*) + l_{v^* w^*}$.

Докажем, что $len(w^*) = dist(s, w^*)$. Для этого докажем, что

1) $dist(s, w^*) \le len(w^*)$;
2) $dist(s, w^*) \ge len(w^*)$

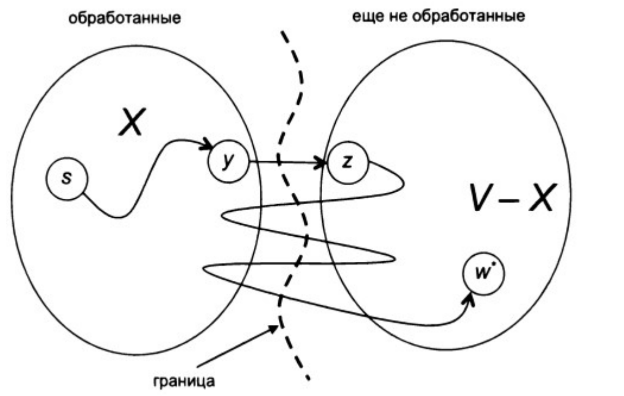
(1) $dist(s, w^*) \leq len(w^*)$. По индукционной гипотезе $\boldsymbol{dist(s, v^*) = len(v^*)}$




Прикрепление ребра $(v^*, w^*)$ в конце кратчайшего $s$-$v^*$-пути $P$ производит $s$-$w^*$-путь $P^*$ длины $len(v^*) + l_{v^*w^*}$

Кратчайшее расстояние до вершины не превышает длину пути в эту вершину, поэтому $dist(s, w^*) \leq len(w^*)$

(2) $dist(s, w^*) \geq len(w^*)$. Рассмотрим какой-нибудь конкурирующий путь $P'$ $(s \longrightarrow w^*)$. Докажем, что длина $P' \geq len(w^*)$




Каждый путь $s$-$w^*$ пересекает не менее одного раза из $X$ в $V - X$

Пусть $(y, z)$ – первое ребро в $P'$, которое пересекает границу. Разобьём $P'$ на 3 части: $len(y)$, $(y, z)$, $z \longrightarrow w^*$

длины: $dist(s, y)$, $l_{yz}$, неотрицательно

Имеем: длина $P' \geq \underbrace{dist(s, y) + l_{yz}}_{\text{дейкстрова оценка ребра } (y,z)} + 0$

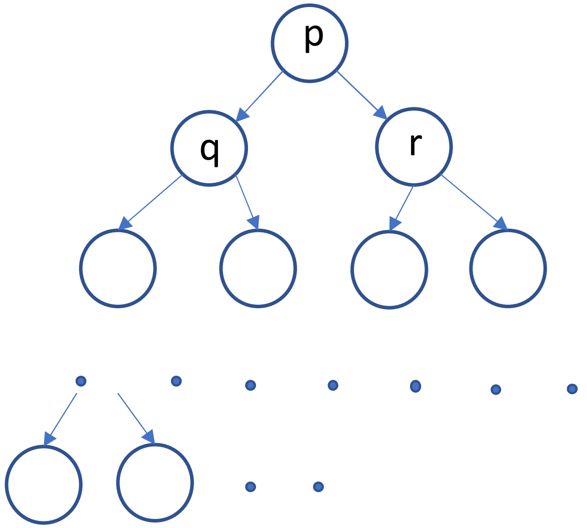
Т.к. алгоритм выбрал ребро с наименьшей дейкстровой оценкой, то $len(v^*) + l_{v^*w^*} \leq len(y) + l_{yz}$

и длина $P' \geq len(v^*) + l_{v^*w^*} = len(w^*)$. Среди всех путей $P'$ выберем кратчайший путь, поэтому $dist(s, w^*) \geq len(w^*)$. Если вершина не добавлялась в *X*, $dist(s, v) = \infty = len(v)$. **Теорема доказана.**

# Время выполнения алгоритма Дейкстры

Если на шаге 4 ( 4) $(v^*, w^*) :=$ такое ребро, которое минимизирует $len(v) + l_{vw}$ ) выбирать $\min\limits_{\substack{v \in X, \\ w \in V-X}} (len(v) + l_{vw}) = l_{v^* w^*}$

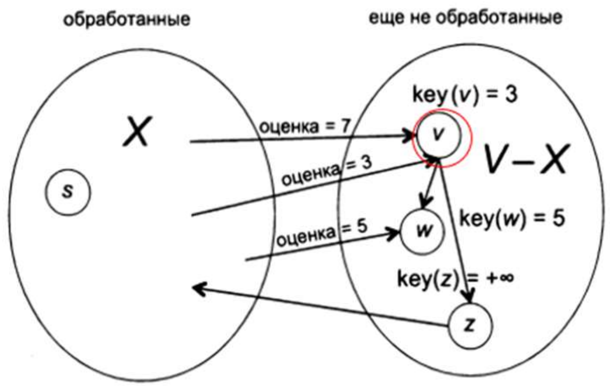
для каждой из $O(|V|)$ вершин перебирать $O(|E|)$ ребер, то время работы $O(|V| \cdot |E|)$ . Можно ли быстрее? – да, если использовать двоичную кучу.



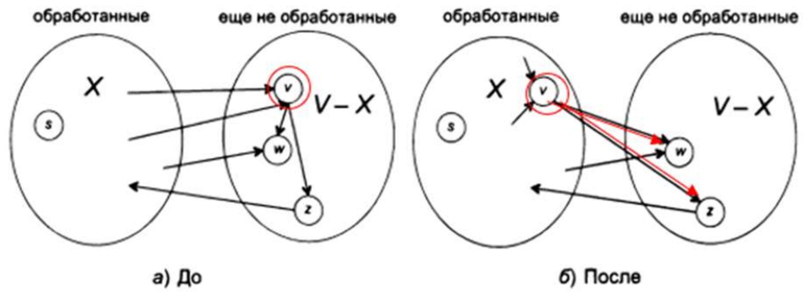

Массив можно представить в виде полного двоичного дерева, где ключ родителя не больше ключа потомка $(p \leq q, p \leq r)$

Куча поддерживает основные операции «вставить» и «извлечь минимум», которые выполняются за время $O(\log n)$.
Будем хранить в куче необработанные вершины из V-X.
При этом будем поддерживать инвариант:
Ключом вершины $w \in V - X$ является минимальная дейкстрова оценка ребра с хвостом $v \in X$ и с головой $w \in V - X$ либо $+\infty$, если такое ребро не существует:
$key(w) = \min\limits_{\substack{v \in X, \\ w \in V-X}} (len(v) + l_{vw})$

Инвариант $key(w) = \min_{\substack{v \in X, \\ w \in V-X}} (len(v) + l_{vw})$



Ключ вершины $w \in V - X$ определяется как минимальная дейкстрова оценка ребра с головой $w$ и хвостом в $X$



Когда новая вершина $v$ перемещается из $V - X$ в $X$, ребра, выходящие из $v$, могут стать пересекающими ребрами

После перемещения вершины-победителя $w^*$ в X необходимо перебрать список ребер, исходящих из $w^*$ и проверить вершины $y \in V - X$ с ребром $(w^*, y)$, вычислив для них ключи согласно инварианту.

# Алгоритм Дейкстры на основе кучи

**DIJKSTRA (НА ОСНОВЕ КУЧИ, ЧАСТЬ 1)**

**Вход**: ориентированный граф $G = (V, E)$, представленный в виде списков смежности, вершина $s \in V$, длина $l_e \geq 0$ для каждого $e \in E$.

**Постусловие**: для каждой вершины $v$ значение $len(v)$ равно истинному кратчайшему расстоянию $dist(s, v)$.

---

```
    // Инициализация
1)  X := пустое множество, H := пустая куча
2)  key(s) := 0
3)  for каждая v ≠ s do
4)      key(v) := +∞; prev := nil
5)  for каждая v ∈ V do
6)      Вставить v в H  // либо использовать операцию
                        // «Объединить в кучу»
    // Главный цикл
7)  while H является непустой do
8)      w* := Извлечь минимум (H)
9)      добавить w* в X
10)     len(w*) := key(w*)
        // обновить кучу для поддержания инварианта
```

**DIJKSTRA (НА ОСНОВЕ КУЧИ, ЧАСТЬ 2)**

```
11) // обновить кучу для поддержания инварианта
12) for каждое ребро (w*, y) do
13)     Удалить y из H
14)     key(y) := min{key(y), len(w*) + l_{w*y}}
15)     prev := w*
16)     вставить y в H
```

**Время работы:**

Обозначим $n = |V|, m = |E|$

Строка 6: $O(n)$ или $O(n \cdot log\ n)$

Цикл стр.7 :$O(n)$

в цикле стр.8 по 1 разу $O(log\ n)$

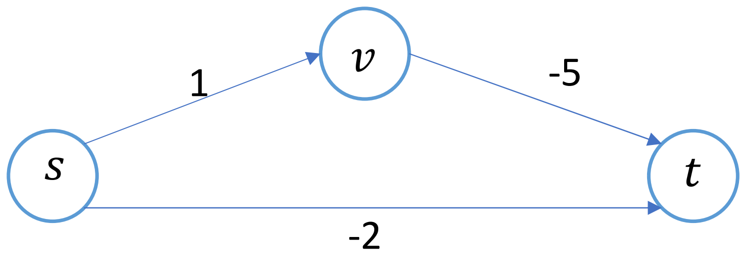
цикл for стр.12 : $O(m)$

стр. 13 по 1 разу на ребро: $O(log\ n)$

стр. 16 по 1 разу на ребро: $O(log\ n)$

$$O(n) + O(n \cdot log\ n) + O(m \cdot log\ n) + O(m \cdot log\ n) = O((m+n)\log n)$$

# Ребра с отрицательным весом

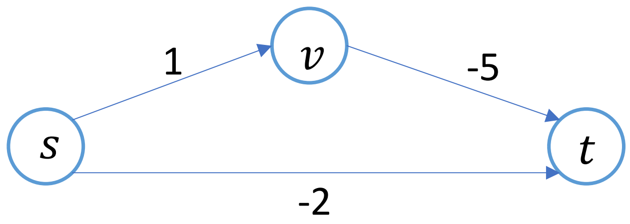



Если в некоторую вершину существует кратчайший путь, то длина этого пути не больше $|V| - 1$.

Если последовательность вершин кратчайшего пути известна $(s, v_1, v_2, \ldots, v_k, t)$, то $len(v_m) = \min\{len(v_m), len(v_{m-1}) + l(v_{m-1}, v_m)\}$

Если заранее кратчайшие пути неизвестны, то можно сделать $len(v_m)$итераций, обновляя каждое ребро по одному разу. Тогда наверняка минимальные расстояния до каждой вершины будут найдены верно.

# Ребра с отрицательным весом (продолжение)

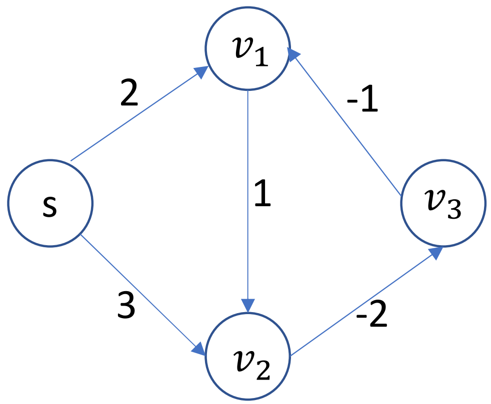



Каждая итерация либо оставляет либо уменьшает расстояние до вершины.

| итерации / Вершина | 0 | 1 | 2 |
|---|---|---|---|
| $s$ | 0 | 0 | 0 |
| $v$ | ∞ | 1 | 1 |
| $t$ | ∞ | -2 | -4 |

Рассмотрим граф с отрицательным циклом.

| итерации / вершина | 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| $s$ | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| $v_1$ | ∞ | 2 | 2 | 0 | 0 | 0 | -2 |
| $v_2$ | ∞ | 3 | 3 | 3 | 1 | 1 | -1 |
| $v_3$ | ∞ | ∞ | 1 | 1 | 1 | -1 | -1 |




Метки вершин цикла можно неограниченно уменьшать. Признак наличия отрицательного цикла:

$$len(v) < len(u) + l_{uv}$$

после $|V| - 1$ итерации

# Алгоритм Беллмана-Форда

Вход: орграф G=($V$, $E$), $\forall x \;\; l(x) \in \mathbb{R}, s \in V$

Выход: $\forall x \in V \;\; len(v) = dist(s, v)$ либо объявление, что граф содержит отрицательный цикл

for каждой $v \neq s$ do

  $len(v) := \infty; \;\; prev[v] := nil$

  $len[s] := 0$

for $i := 1$ to $|V| - 1$ do // перебрать $|V| - 1$ раз все ребра

  for каждого ребра $(u, v) \in E$

    if $len[v] > len[u] + l_{uv}$ then

      $len[v] \coloneqq len[u] + l_{uv}$

      $prev[v] \coloneqq u$

for каждого ребра $(u, v) \in E$

    if $len[v] > len[u] + l_{uv}$ then

      return false // есть цикл отрицательной длины

return true

Время работы : $O(|V| \cdot |E|)$

# Корректность алгоритма Беллмана-Форда

Если есть цикл отрицательной длины, то последний контрольный проход уменьшит метку какой-либо вершины. Докажем, что если есть цикл отрицательной длины, то будет «false». Допустим, что это не так:

Есть цикл c=$< v_0, v_1, \dots, v_k >$ отрицательной длины, но программа выдает «true».

$\sum_{i=1}^{k} l_{v_{i-1}v_i} < 0$, $v_0 = v_k$. $len[v_i] \le len[v_{i-1}] + l_{v_{i-1}v_i}$. Тогда $\sum_{i=1}^{k} len[v_i] \le \sum_{i=1}^{k} len[v_{i-1}] + \sum_{i=1}^{k} l_{v_{i-1}v_i} \Rightarrow$

$$\Rightarrow \sum_{i=1}^{k} len[v_i] + \cancel{len[v_0]} = \sum_{i=1}^{k} len[v_{i-1}] + \cancel{len[v_k]}$$

$0 \le \sum_{i=1}^{k} l_{v_{i-1}v_i}$ противоречие

Если граф ациклический, то можно решить быстрее. Применим топологическое упорядочение и затем пройдем по вершинам в найденном порядке.