Кратчайшие пути в графах

Н.Н. Писарук
pisaruk@yandex.by

Экономический факультет
Белорусский государственный университет

Минск - 2014

План лекции

1. Дерево кратчайших путей
   - Принцип оптимальности
   - Критерий существования дерева кратчайших путей
2. Алгоритмы поиска кратчайших путей
   - Алгоритм Форда — Беллмана
   - Алгоритм Дейкстры
   - Кратчайшие пути в ациклических графах

Задачи о кратчайших путях

В орграфе $G = (V, E)$, каждой дуге которого приписана *стоимость* (*длина*) $c(v, w)$, нужно найти

1. путь $P = \{s = v_0, v_1, \ldots, v_{k-1}, v_k = t\}$

   от вершины $s \in V$ до вершины $t \in V$ минимальной стоимости (*кратчайший путь*)

   $$c(P) \stackrel{\text{def}}{=} \sum_{i=1}^{k} c(v_{i-1}, v_i).$$

2. кратчайшие пути от выделенной вершины $s \in V$ до всех остальных вершин графа.
3. кратчайшие пути от всех вершин до выделенной вершины $t \in V$.
4. кратчайшие пути между всеми парами вершин $s, t \in V$.

Как это не парадоксально, но найти кратчайший путь между двумя выделенными вершинами не легче, чем искать кратчайшие пути от одной вершины до всех остальных.

Задачи о кратчайших путях

В орграфе $G = (V, E)$, каждой дуге которого приписана *стоимость* (*длина*) $c(v, w)$, нужно найти

1. путь $P = \{s = v_0, v_1, \dots, v_{k-1}, v_k = t\}$

   от вершины $s \in V$ до вершины $t \in V$ минимальной стоимости (*кратчайший путь*)

   $$c(P) \stackrel{\text{def}}{=} \sum_{i=1}^{k} c(v_{i-1}, v_i).$$

2. кратчайшие пути от выделенной вершины $s \in V$ до всех остальных вершин графа.
3. кратчайшие пути от всех вершин до выделенной вершины $t \in V$.
4. кратчайшие пути между всеми парами вершин $s, t \in V$.

Как это не парадоксально, но найти кратчайший путь между двумя выделенными вершинами не легче, чем искать кратчайшие пути от одной вершины до всех остальных.

Задачи о кратчайших путях

В орграфе $G = (V, E)$, каждой дуге которого приписана *стоимость* (*длина*) $c(v, w)$, нужно найти

1. путь $P = \{s = v_0, v_1, \dots, v_{k-1}, v_k = t\}$

   от вершины $s \in V$ до вершины $t \in V$ минимальной стоимости (*кратчайший путь*)

   $$c(P) \stackrel{\text{def}}{=} \sum_{i=1}^{k} c(v_{i-1}, v_i).$$

2. кратчайшие пути от выделенной вершины $s \in V$ до всех остальных вершин графа.
3. кратчайшие пути от всех вершин до выделенной вершины $t \in V$.
4. кратчайшие пути между всеми парами вершин $s, t \in V$.

Как это не парадоксально, но найти кратчайший путь между двумя выделенными вершинами не легче, чем искать кратчайшие пути от одной вершины до всех остальных.

Задачи о кратчайших путях

В орграфе $G = (V, E)$, каждой дуге которого приписана *стоимость* (*длина*) $c(v, w)$, нужно найти

1. путь $P = \{s = v_0, v_1, \dots, v_{k-1}, v_k = t\}$

   от вершины $s \in V$ до вершины $t \in V$ минимальной стоимости (*кратчайший путь*)

   $$c(P) \stackrel{\text{def}}{=} \sum_{i=1}^{k} c(v_{i-1}, v_i).$$

2. кратчайшие пути от выделенной вершины $s \in V$ до всех остальных вершин графа.
3. кратчайшие пути от всех вершин до выделенной вершины $t \in V$.
4. кратчайшие пути между всеми парами вершин $s, t \in V$.

Как это не парадоксально, но найти кратчайший путь между двумя выделенными вершинами не легче, чем искать кратчайшие пути от одной вершины до всех остальных.

Задачи о кратчайших путях

В орграфе $G = (V, E)$, каждой дуге которого приписана *стоимость* (*длина*) $c(v, w)$, нужно найти

1. путь $P = \{s = v_0, v_1, \dots, v_{k-1}, v_k = t\}$

   от вершины $s \in V$ до вершины $t \in V$ минимальной стоимости (*кратчайший путь*)

   $$c(P) \stackrel{\text{def}}{=} \sum_{i=1}^{k} c(v_{i-1}, v_i).$$

2. кратчайшие пути от выделенной вершины $s \in V$ до всех остальных вершин графа.
3. кратчайшие пути от всех вершин до выделенной вершины $t \in V$.
4. кратчайшие пути между всеми парами вершин $s, t \in V$.

Как это не парадоксально, но найти кратчайший путь между двумя выделенными вершинами не легче, чем искать кратчайшие пути от одной вершины до всех остальных.

Задачи о кратчайших путях

В орграфе $G = (V, E)$, каждой дуге которого приписана *стоимость* (*длина*) $c(v, w)$, нужно найти

1. путь $P = \{s = v_0, v_1, \dots, v_{k-1}, v_k = t\}$

   от вершины $s \in V$ до вершины $t \in V$ минимальной стоимости (*кратчайший путь*)

   $$c(P) \stackrel{\text{def}}{=} \sum_{i=1}^{k} c(v_{i-1}, v_i).$$

2. кратчайшие пути от выделенной вершины $s \in V$ до всех остальных вершин графа.
3. кратчайшие пути от всех вершин до выделенной вершины $t \in V$.
4. кратчайшие пути между всеми парами вершин $s, t \in V$.

Как это не парадоксально, но найти кратчайший путь между двумя выделенными вершинами не легче, чем искать кратчайшие пути от одной вершины до всех остальных.

Задачи о кратчайших путях

В орграфе $G = (V, E)$, каждой дуге которого приписана *стоимость* (*длина*) $c(v, w)$, нужно найти

1. путь $P = \{s = v_0, v_1, \ldots, v_{k-1}, v_k = t\}$

   от вершины $s \in V$ до вершины $t \in V$ минимальной стоимости (*кратчайший путь*)

$$c(P) \stackrel{\text{def}}{=} \sum_{i=1}^{k} c(v_{i-1}, v_i).$$

2. кратчайшие пути от выделенной вершины $s \in V$ до всех остальных вершин графа.
3. кратчайшие пути от всех вершин до выделенной вершины $t \in V$.
4. кратчайшие пути между всеми парами вершин $s, t \in V$.

Как это не парадоксально, но найти кратчайший путь между двумя выделенными вершинами не легче, чем искать кратчайшие пути от одной вершины до всех остальных.

План лекции

1. Дерево кратчайших путей
   - Принцип оптимальности
   - Критерий существования дерева кратчайших путей
2. Алгоритмы поиска кратчайших путей
   - Алгоритм Форда — Беллмана
   - Алгоритм Дейкстры
   - Кратчайшие пути в ациклических графах

Принцип оптимальности

Пусть $G = (V, E)$ есть орграф с выделенной вершиной $s \in V$, каждой дуге $(v, w) \in E$ приписана стоимость $c(v, w)$.

Принцип оптимальности динамического программирования

- Пусть $P = (s = v_0, v_1 \ldots, v_k = t)$ есть кратчайший путь от вешины s до вершины t в графе G.
- Тогда для любого $0 < i < k$ начальная часть пути $P_i = (s = v_0, v_1 \ldots, v_i)$ есть кратчайший путь от s до v_i,
- так как иначе отрезок P_i пути P можно заменить кратчайшим путем из s в v_i и получить более короткий путь P' из s в t.

Принцип оптимальности

Пусть $G = (V, E)$ есть орграф с выделенной вершиной $s \in V$, каждой дуге $(v, w) \in E$ приписана стоимость $c(v, w)$.

Принцип оптимальности динамического программирования

- Пусть $P = (s = v_0, v_1 \ldots, v_k = t)$ есть кратчайший путь от вешины s до вершины t в графе G.
- Тогда для любого $0 < i < k$ начальная часть пути $P_i = (s = v_0, v_1 \ldots, v_i)$ есть кратчайший путь от s до v_i,
- так как иначе отрезок P_i пути P можно заменить кратчайшим путем из s в v_i и получить более короткий путь P' из s в t.

Принцип оптимальности

Пусть $G = (V, E)$ есть орграф с выделенной вершиной $s \in V$, каждой дуге $(v, w) \in E$ приписана стоимость $c(v, w)$.

Принцип оптимальности динамического программирования

- Пусть $P = (s = v_0, v_1 \dots, v_k = t)$ есть кратчайший путь от вешины s до вершины t в графе G.
- Тогда для любого $0 < i < k$ начальная часть пути $P_i = (s = v_0, v_1 \dots, v_i)$ есть кратчайший путь от s до v_i,
- так как иначе отрезок P_i пути P можно заменить кратчайшим путем из s в v_i и получить более короткий путь P' из s в t.

Принцип оптимальности

Пусть $G = (V, E)$ есть орграф с выделенной вершиной $s \in V$, каждой дуге $(v, w) \in E$ приписана стоимость $c(v, w)$.

Принцип оптимальности динамического программирования

- Пусть $P = (s = v_0, v_1 \dots, v_k = t)$ есть кратчайший путь от вешины s до вершины t в графе G.
- Тогда для любого $0 < i < k$ начальная часть пути $P_i = (s = v_0, v_1 \dots, v_i)$ есть кратчайший путь от s до v_i,
- так как иначе отрезок P_i пути P можно заменить кратчайшим путем из s в v_i и получить более короткий путь P' из s в t.

Принцип оптимальности

Пусть $G = (V, E)$ есть орграф с выделенной вершиной $s \in V$, каждой дуге $(v, w) \in E$ приписана стоимость $c(v, w)$.

Принцип оптимальности динамического программирования

- Пусть $P = (s = v_0, v_1 \ldots, v_k = t)$ есть кратчайший путь от вешины s до вершины t в графе G.
- Тогда для любого $0 < i < k$ начальная часть пути $P_i = (s = v_0, v_1 \ldots, v_i)$ есть кратчайший путь от s до v_i,
- так как иначе отрезок P_i пути P можно заменить кратчайшим путем из s в v_i и получить более короткий путь P' из s в t.

Принцип оптимальности

Пусть $G = (V, E)$ есть орграф с выделенной вершиной $s \in V$, каждой дуге $(v, w) \in E$ приписана стоимость $c(v, w)$.

Принцип оптимальности динамического программирования

- Пусть $P = (s = v_0, v_1 \dots, v_k = t)$ есть кратчайший путь от вешины s до вершины t в графе G.
- Тогда для любого $0 < i < k$ начальная часть пути $P_i = (s = v_0, v_1 \dots, v_i)$ есть кратчайший путь от s до v_i,
- так как иначе отрезок P_i пути P можно заменить кратчайшим путем из s в v_i и получить более короткий путь P' из s в t.

Уравнения Беллмана

- Стоимость кратчайшего пути в графе G от вершины s до вершины $v \in V$ обозначим через $\sigma(s, v)$
- (если такого пути не существует, то $\sigma(s, v) = +\infty$).
- Если в графе G нет циклов отрицательной стоимости,
- то все кратчайшие пути являются простыми
- и из принципа оптимальности можно сделать вывод, что стоимости кратчайших путей удовлетворяют следующим уравнениям Беллмана:

$$\sigma(s, s) = 0,$$

$$\sigma(s, v) = \min_{(w,v) \in E(V,v)} (\sigma(s, w) + c(w, v)) \quad \text{для всех } v \in V \setminus s.$$

Уравнения Беллмана

- Стоимость кратчайшего пути в графе G от вершины s до вершины $v \in V$ обозначим через $\sigma(s, v)$
- (если такого пути не существует, то $\sigma(s, v) = +\infty$).
- Если в графе G нет циклов отрицательной стоимости,
- то все кратчайшие пути являются простыми
- и из принципа оптимальности можно сделать вывод, что стоимости кратчайших путей удовлетворяют следующим уравнениям Беллмана:

$$\sigma(s, s) = 0,$$

$$\sigma(s, v) = \min_{(w,v) \in E(V,v)} (\sigma(s, w) + c(w, v)) \quad \text{для всех } v \in V \setminus s.$$

Уравнения Беллмана

- Стоимость кратчайшего пути в графе G от вершины s до вершины $v \in V$ обозначим через $\sigma(s, v)$
- (если такого пути не существует, то $\sigma(s, v) = +\infty$).
- Если в графе G нет циклов отрицательной стоимости,
- то все кратчайшие пути являются простыми
- и из принципа оптимальности можно сделать вывод, что стоимости кратчайших путей удовлетворяют следующим уравнениям Беллмана:

$$\sigma(s, s) = 0,$$

$$\sigma(s, v) = \min_{(w,v) \in E(V,v)} (\sigma(s, w) + c(w, v)) \quad \text{для всех } v \in V \setminus s.$$

Уравнения Беллмана

- Стоимость кратчайшего пути в графе G от вершины s до вершины $v \in V$ обозначим через $\sigma(s, v)$
- (если такого пути не существует, то $\sigma(s, v) = +\infty$).
- Если в графе G нет циклов отрицательной стоимости,
- то все кратчайшие пути являются простыми
- и из принципа оптимальности можно сделать вывод, что стоимости кратчайших путей удовлетворяют следующим уравнениям Беллмана:

$$\sigma(s, s) = 0,$$

$$\sigma(s, v) = \min_{(w,v) \in E(V,v)} (\sigma(s, w) + c(w, v)) \quad \text{для всех } v \in V \setminus s.$$

Уравнения Беллмана

- Стоимость кратчайшего пути в графе G от вершины s до вершины $v \in V$ обозначим через $\sigma(s,v)$
- (если такого пути не существует, то $\sigma(s,v) = +\infty$).
- Если в графе G нет циклов отрицательной стоимости,
- то все кратчайшие пути являются простыми
- и из принципа оптимальности можно сделать вывод, что стоимости кратчайших путей удовлетворяют следующим уравнениям Беллмана:

$$\sigma(s,s) = 0,$$

$$\sigma(s,v) = \min_{(w,v)\in E(V,v)} (\sigma(s,w) + c(w,v)) \quad \text{для всех } v \in V \setminus s.$$

Уравнения Беллмана

- Стоимость кратчайшего пути в графе G от вершины s до вершины $v \in V$ обозначим через $\sigma(s,v)$
- (если такого пути не существует, то $\sigma(s,v) = +\infty$).
- Если в графе G нет циклов отрицательной стоимости,
- то все кратчайшие пути являются простыми
- и из принципа оптимальности можно сделать вывод, что стоимости кратчайших путей удовлетворяют следующим уравнениям Беллмана:

  $\sigma(s,s) = 0,$

  $$\sigma(s,v) = \min_{(w,v)\in E(V,v)} (\sigma(s,w) + c(w,v)) \quad \text{для всех } v \in V \setminus s.$$

Уравнения Беллмана

- Стоимость кратчайшего пути в графе G от вершины s до вершины $v \in V$ обозначим через $\sigma(s,v)$
- (если такого пути не существует, то $\sigma(s,v) = +\infty$).
- Если в графе G нет циклов отрицательной стоимости,
- то все кратчайшие пути являются простыми
- и из принципа оптимальности можно сделать вывод, что стоимости кратчайших путей удовлетворяют следующим уравнениям Беллмана:

$$\sigma(s,s) = 0,$$

$$\sigma(s,v) = \min_{(w,v)\in E(V,v)} (\sigma(s,w) + c(w,v)) \quad \text{для всех } v \in V \setminus s.$$

План лекции

1. Дерево кратчайших путей
   - Принцип оптимальности
   - Критерий существования дерева кратчайших путей
2. Алгоритмы поиска кратчайших путей
   - Алгоритм Форда — Беллмана
   - Алгоритм Дейкстры
   - Кратчайшие пути в ациклических графах

Приведенные стоимости

- *Функция цен* есть функция $p : V \to \mathbb{R}$.
- *Приведенная функция стоимости* $c_p : V \to \mathbb{R}$ *относительно функции цен* p определяется по правилу:

$$c_p(v, w) = p(v) + c(v, w) - p(w).$$

- Цены вершин имеют натуральную экономическую интерпретацию как действующие рыночные цены на некоторый продукт.
- Мы можем интерпретировать приведенную стоимость $c_p(v, w)$, как суму затрат на
- закупку единицы продукта в вершине v по цене $p(v)$
- и затрат $c(v, w)$ на транспортировку в вершину w
- минус доход от продажи ее там по цене $p(w)$.

Приведенные стоимости

- *Функция цен* есть функция $p : V \to \mathbb{R}$.
- *Приведенная функция стоимости* $c_p : V \to \mathbb{R}$ *относительно функции цен* p определяется по правилу:
$$c_p(v, w) = p(v) + c(v, w) - p(w).$$
- Цены вершин имеют натуральную экономическую интерпретацию как действующие рыночные цены на некоторый продукт.
- Мы можем интерпретировать приведенную стоимость $c_p(v, w)$, как суму затрат на
- закупку единицы продукта в вершине v по цене $p(v)$
- и затрат $c(v, w)$ на транспортировку в вершину w
- минус доход от продажи ее там по цене $p(w)$.

Приведенные стоимости

- *Функция цен* есть функция $p : V \to \mathbb{R}$.
- *Приведенная функция стоимости* $c_p : V \to \mathbb{R}$ *относительно функции цен* p определяется по правилу:

$$c_p(v, w) = p(v) + c(v, w) - p(w).$$

- Цены вершин имеют натуральную экономическую интерпретацию как действующие рыночные цены на некоторый продукт.
- Мы можем интерпретировать приведенную стоимость $c_p(v, w)$, как суму затрат на
- закупку единицы продукта в вершине v по цене $p(v)$
- и затрат $c(v, w)$ на транспортировку в вершину w
- минус доход от продажи ее там по цене $p(w)$.

Приведенные стоимости

- *Функция цен* есть функция $p : V \to \mathbb{R}$.
- *Приведенная функция стоимости* $c_p : V \to \mathbb{R}$ *относительно функции цен* p определяется по правилу:

$$c_p(v, w) = p(v) + c(v, w) - p(w).$$

- Цены вершин имеют натуральную экономическую интерпретацию как действующие рыночные цены на некоторый продукт.
- Мы можем интерпретировать приведенную стоимость $c_p(v, w)$, как суму затрат на
- закупку единицы продукта в вершине v по цене $p(v)$
- и затрат $c(v, w)$ на транспортировку в вершину w
- минус доход от продажи ее там по цене $p(w)$.

Приведенные стоимости

- *Функция цен* есть функция $p : V \to \mathbb{R}$.
- *Приведенная функция стоимости* $c_p : V \to \mathbb{R}$ *относительно функции цен* p определяется по правилу:
$$c_p(v, w) = p(v) + c(v, w) - p(w).$$
- Цены вершин имеют натуральную экономическую интерпретацию как действующие рыночные цены на некоторый продукт.
- Мы можем интерпретировать приведенную стоимость $c_p(v, w)$, как суму затрат на
- закупку единицы продукта в вершине v по цене $p(v)$
- и затрат $c(v, w)$ на транспортировку в вершину w
- минус доход от продажи ее там по цене $p(w)$.

Приведенные стоимости

- *Функция цен* есть функция $p : V \to \mathbb{R}$.
- *Приведенная функция стоимости* $c_p : V \to \mathbb{R}$ *относительно функции цен* p определяется по правилу:
$$c_p(v,w) = p(v) + c(v,w) - p(w).$$
- Цены вершин имеют натуральную экономическую интерпретацию как действующие рыночные цены на некоторый продукт.
- Мы можем интерпретировать приведенную стоимость $c_p(v,w)$, как суму затрат на
- закупку единицы продукта в вершине v по цене $p(v)$
- и затрат $c(v,w)$ на транспортировку в вершину w
- минус доход от продажи ее там по цене $p(w)$.

Приведенные стоимости

- *Функция цен* есть функция $p : V \to \mathbb{R}$.
- *Приведенная функция стоимости* $c_p : V \to \mathbb{R}$ *относительно функции цен* p определяется по правилу:

$$c_p(v,w) = p(v) + c(v,w) - p(w).$$

- Цены вершин имеют натуральную экономическую интерпретацию как действующие рыночные цены на некоторый продукт.
- Мы можем интерпретировать приведенную стоимость $c_p(v,w)$, как суму затрат на
- закупку единицы продукта в вершине v по цене $p(v)$
- и затрат $c(v,w)$ на транспортировку в вершину w
- минус доход от продажи ее там по цене $p(w)$.

Приведенные стоимости путей и циклов

Лемма 1

- *Пусть $G = (V, E)$ есть орграф, на дугах которого определена функция стоимости c, а на вершинах функция цен p.*
- *Тогда для пути P из вершины v в вершину w в графе G имеет место равенство $c_p(P) = c(P) + p(v) - p(w)$.*
- *В частности, если P — цикл, то $c_p(P) = c(P)$.*

Доказательство.

Пусть $P = (v = v_0, v_1, \dots, v_{k-1}, v_k = w)$. Тогда

$$\begin{aligned} c_p(P) &= \sum_{i=1}^k c_p(v_{i-1}, v_i) = \sum_{i=1}^k (c(v_{i-1}, v_i) + p(v_{i-1}) - p(v_i)) \\ &= \sum_{i=1}^k c(v_{i-1}, v_i) + \sum_{i=1}^k p(v_{i-1}) - \sum_{i=1}^k p(v_i) \\ &= c(P) + p(v) - p(w). \end{aligned}$$

$\square$

Приведенные стоимости путей и циклов

Лемма 1

- *Пусть $G = (V, E)$ есть орграф, на дугах которого определена функция стоимости c, а на вершинах функция цен p.*
- *Тогда для пути P из вершины v в вершину w в графе G имеет место равенство $c_p(P) = c(P) + p(v) - p(w)$.*
- *В частности, если P — цикл, то $c_p(P) = c(P)$.*

Доказательство.

Пусть $P = (v = v_0, v_1, \ldots, v_{k-1}, v_k = w)$. Тогда

$$c_p(P) = \sum_{i=1}^k c_p(v_{i-1}, v_i) = \sum_{i=1}^k (c(v_{i-1}, v_i) + p(v_{i-1}) - p(v_i))$$
$$= \sum_{i=1}^k c(v_{i-1}, v_i) + \sum_{i=1}^k p(v_{i-1}) - \sum_{i=1}^k p(v_i)$$
$$= c(P) + p(v) - p(w).$$

□

Приведенные стоимости путей и циклов

Лемма 1

- *Пусть $G = (V, E)$ есть орграф, на дугах которого определена функция стоимости c, а на вершинах функция цен p.*
- *Тогда для пути P из вершины v в вершину w в графе G имеет место равенство $c_p(P) = c(P) + p(v) - p(w)$.*
- *В частности, если P — цикл, то $c_p(P) = c(P)$.*

Доказательство.

Пусть $P = (v = v_0, v_1, \ldots, v_{k-1}, v_k = w)$. Тогда

$$\begin{aligned} c_p(P) &= \sum_{i=1}^k c_p(v_{i-1}, v_i) = \sum_{i=1}^k (c(v_{i-1}, v_i) + p(v_{i-1}) - p(v_i)) \\ &= \sum_{i=1}^k c(v_{i-1}, v_i) + \sum_{i=1}^k p(v_{i-1}) - \sum_{i=1}^k p(v_i) \\ &= c(P) + p(v) - p(w). \end{aligned}$$

□

Приведенные стоимости путей и циклов

Лемма 1

- *Пусть $G = (V, E)$ есть орграф, на дугах которого определена функция стоимости c, а на вершинах функция цен p.*
- *Тогда для пути P из вершины v в вершину w в графе G имеет место равенство $c_p(P) = c(P) + p(v) - p(w)$.*
- *В частности, если P — цикл, то $c_p(P) = c(P)$.*

Доказательство.

Пусть $P = (v = v_0, v_1, \dots, v_{k-1}, v_k = w)$. Тогда

$$
\begin{aligned}
c_p(P) &= \sum_{i=1}^k c_p(v_{i-1}, v_i) = \sum_{i=1}^k (c(v_{i-1}, v_i) + p(v_{i-1}) - p(v_i)) \\
&= \sum_{i=1}^k c(v_{i-1}, v_i) + \sum_{i=1}^k p(v_{i-1}) - \sum_{i=1}^k p(v_i) \\
&= c(P) + p(v) - p(w).
\end{aligned}
$$

$\square$

Приведенные стоимости путей и циклов

Лемма 1

- *Пусть* $G = (V, E)$ *есть орграф, на дугах которого определена функция стоимости* c, *а на вершинах функция цен* p.
- *Тогда для пути* P *из вершины* v *в вершину* w *в графе* G *имеет место равенство* $c_p(P) = c(P) + p(v) - p(w)$.
- *В частности, если* P *— цикл, то* $c_p(P) = c(P)$.

Доказательство.

Пусть $P = (v = v_0, v_1, \dots, v_{k-1}, v_k = w)$. Тогда

$$c_p(P) = \sum_{i=1}^k c_p(v_{i-1}, v_i) = \sum_{i=1}^k (c(v_{i-1}, v_i) + p(v_{i-1}) - p(v_i))$$

$$= \sum_{i=1}^k c(v_{i-1}, v_i) + \sum_{i=1}^k p(v_{i-1}) - \sum_{i=1}^k p(v_i)$$

$$= c(P) + p(v) - p(w).$$

□

Приведенные стоимости путей и циклов

Лемма 1

- *Пусть* $G = (V, E)$ *есть орграф, на дугах которого определена функция стоимости* c, *а на вершинах функция цен* p.
- *Тогда для пути* P *из вершины* v *в вершину* w *в графе* G *имеет место равенство* $c_p(P) = c(P) + p(v) - p(w)$.
- *В частности, если* P *— цикл, то* $c_p(P) = c(P)$.

Доказательство.

Пусть $P = (v = v_0, v_1, \dots, v_{k-1}, v_k = w)$. Тогда

$$c_p(P) = \sum_{i=1}^k c_p(v_{i-1}, v_i) = \sum_{i=1}^k (c(v_{i-1}, v_i) + p(v_{i-1}) - p(v_i))$$
$$= \sum_{i=1}^k c(v_{i-1}, v_i) + \sum_{i=1}^k p(v_{i-1}) - \sum_{i=1}^k p(v_i)$$
$$= c(P) + p(v) - p(w).$$

□

Приведенные стоимости путей и циклов

Лемма 1

- *Пусть $G = (V, E)$ есть орграф, на дугах которого определена функция стоимости c, а на вершинах функция цен p.*
- *Тогда для пути P из вершины v в вершину w в графе G имеет место равенство $c_p(P) = c(P) + p(v) - p(w)$.*
- *В частности, если P — цикл, то $c_p(P) = c(P)$.*

Доказательство.

Пусть $P = (v = v_0, v_1, \dots, v_{k-1}, v_k = w)$. Тогда

$$\begin{aligned} c_p(P) &= \sum_{i=1}^k c_p(v_{i-1}, v_i) = \sum_{i=1}^k (c(v_{i-1}, v_i) + p(v_{i-1}) - p(v_i)) \\ &= \sum_{i=1}^k c(v_{i-1}, v_i) + \sum_{i=1}^k p(v_{i-1}) - \sum_{i=1}^k p(v_i) \\ &= c(P) + p(v) - p(w). \end{aligned}$$

□

Функция расстояний

- Для покрывающего ордерева T с корнем s *функция расстояний* $d : V \to \mathbb{R}$ определяется рекурсивно следующим образом:
- $d(s) = 0,\ d(v) = d(parent(v)) + c(parent(v), v)$ для $v \in V \setminus s$,
- где $parent(v)$ есть *отец* вершины v в дереве T.
- Покрывающее ордерево T с корнем s называется *деревом кратчайших путей*,
- если для каждой вершины v единственный путь в дереве T из s в v
- является кратчайшим путем из s в v в графе G, т. е. $d(v) = \sigma(s, v)$.

Функция расстояний

- Для покрывающего ордерева T с корнем s *функция расстояний* $d : V \to \mathbb{R}$ определяется рекурсивно следующим образом:
- $d(s) = 0,\ d(v) = d(parent(v)) + c(parent(v), v)$ для $v \in V \setminus s$,
- где $parent(v)$ есть *отец* вершины v в дереве T.
- Покрывающее ордерево T с корнем s называется *деревом кратчайших путей*,
- если для каждой вершины v единственный путь в дереве T из s в v
- является кратчайшим путем из s в v в графе G, т. е. $d(v) = \sigma(s, v)$.

Функция расстояний

- Для покрывающего ордерева T с корнем s *функция расстояний* $d : V \to \mathbb{R}$ определяется рекурсивно следующим образом:
- $d(s) = 0,\ d(v) = d(parent(v)) + c(parent(v), v)$ для $v \in V \setminus s$,
- где $parent(v)$ есть *отец* вершины v в дереве T.
- Покрывающее ордерево T с корнем s называется *деревом кратчайших путей*,
- если для каждой вершины v единственный путь в дереве T из s в v
- является кратчайшим путем из s в v в графе G, т. е. $d(v) = \sigma(s, v)$.

Функция расстояний

- Для покрывающего ордерева T с корнем s *функция расстояний* $d : V \to \mathbb{R}$ определяется рекурсивно следующим образом:
- $d(s) = 0,\ d(v) = d(parent(v)) + c(parent(v), v)$ для $v \in V \setminus s$,
- где $parent(v)$ есть *отец* вершины v в дереве T.
- Покрывающее ордерево T с корнем s называется *деревом кратчайших путей*,
- если для каждой вершины v единственный путь в дереве T из s в v
- является кратчайшим путем из s в v в графе G, т. е. $d(v) = \sigma(s, v)$.

Функция расстояний

- Для покрывающего ордерева T с корнем s *функция расстояний* $d : V \to \mathbb{R}$ определяется рекурсивно следующим образом:
- $d(s) = 0,\ d(v) = d(parent(v)) + c(parent(v), v)$ для $v \in V \setminus s$,
- где $parent(v)$ есть *отец* вершины v в дереве T.
- Покрывающее ордерево T с корнем s называется *деревом кратчайших путей*,
- если для каждой вершины v единственный путь в дереве T из s в v
- является кратчайшим путем из s в v в графе G, т. е. $d(v) = \sigma(s, v)$.

Функция расстояний

- Для покрывающего ордерева T с корнем s *функция расстояний* $d : V \to \mathbb{R}$ определяется рекурсивно следующим образом:
- $d(s) = 0,\ d(v) = d(parent(v)) + c(parent(v), v) \quad$ для $v \in V \setminus s$,
- где $parent(v)$ есть *отец* вершины v в дереве T.
- Покрывающее ордерево T с корнем s называется *деревом кратчайших путей*,
- если для каждой вершины v единственный путь в дереве T из s в v
- является кратчайшим путем из s в v в графе G, т. е. $d(v) = \sigma(s, v)$.

Функция расстояний

Теорема 2

- *Пусть все вершины графа $G = (V, E)$ достигаются из вершины $s \in V$.*
- *Граф G имеет дерево кратчайших путей тогда и только тогда, когда он не имеет циклов отрицательной стоимости.*
- *Покрывающее ордерево T с корнем s является деревом кратчайших путей тогда и только тогда, когда его функция расстояний d удовлетворяет условию:*

$$c_d(v, w) \geq 0 \text{ для всех } (v, w) \in E.$$

Пропустить доказательство

Функция расстояний

Теорема 2

- *Пусть все вершины графа $G = (V, E)$ достигаются из вершины $s \in V$.*
- *Граф G имеет дерево кратчайших путей тогда и только тогда, когда он не имеет циклов отрицательной стоимости.*
- *Покрывающее ордерево T с корнем s является деревом кратчайших путей тогда и только тогда, когда его функция расстояний d удовлетворяет условию:*

$$c_d(v, w) \geq 0 \text{ для всех } (v, w) \in E.$$

Пропустить доказательство

Функция расстояний

Теорема 2

- *Пусть все вершины графа $G = (V, E)$ достигаются из вершины $s \in V$.*
- *Граф G имеет дерево кратчайших путей тогда и только тогда, когда он не имеет циклов отрицательной стоимости.*
- *Покрывающее ордерево T с корнем s является деревом кратчайших путей тогда и только тогда, когда его функция расстояний d удовлетворяет условию:*

$$c_d(v, w) \geq 0 \text{ для всех } (v, w) \in E.$$

Пропустить доказательство

Функция расстояний

Теорема 2

- *Пусть все вершины графа* $G = (V, E)$ *достигаются из вершины* $s \in V$.
- *Граф* G *имеет дерево кратчайших путей тогда и только тогда, когда он не имеет циклов отрицательной стоимости.*
- *Покрывающее ордерево* T *с корнем* s *является деревом кратчайших путей тогда и только тогда, когда его функция расстояний* d *удовлетворяет условию:*

$$c_d(v, w) \geq 0 \text{ для всех } (v, w) \in E.$$

▸ Пропустить доказательство

Функция расстояний: доказательство теоремы

Доказательство.

- Если G не имеет циклов отрицательной стоимости, то все кратчайшие пути в G являются простыми.
- Пусть D есть объединение дуг этих путей.
- Очевидно, что граф $G' = (V, D)$ содержит ордерево T, которое является деревом кратчайших путей.
- Функция d расстояний дерева кратчайших путей удовлетворяет условию $c_d(v, w) \geq 0$ для всех $(v, w) \in E$.

$\square$

Функция расстояний: доказательство теоремы

Доказательство.

- Если G не имеет циклов отрицательной стоимости, то все кратчайшие пути в G являются простыми.
- Пусть D есть объединение дуг этих путей.
- Очевидно, что граф $G' = (V, D)$ содержит ордерево T, которое является деревом кратчайших путей.
- Функция d расстояний дерева кратчайших путей удовлетворяет условию $c_d(v, w) \geq 0$ для всех $(v, w) \in E$.

$\square$

Функция расстояний: доказательство теоремы

Доказательство.

- Если G не имеет циклов отрицательной стоимости, то все кратчайшие пути в G являются простыми.
- Пусть D есть объединение дуг этих путей.
- Очевидно, что граф $G' = (V, D)$ содержит ордерево T, которое является деревом кратчайших путей.
- Функция d расстояний дерева кратчайших путей удовлетворяет условию $c_d(v, w) \geq 0$ для всех $(v, w) \in E$.

□

Функция расстояний: доказательство теоремы

Доказательство.

- Если G не имеет циклов отрицательной стоимости, то все кратчайшие пути в G являются простыми.
- Пусть D есть объединение дуг этих путей.
- Очевидно, что граф $G' = (V, D)$ содержит ордерево T, которое является деревом кратчайших путей.
- Функция d расстояний дерева кратчайших путей удовлетворяет условию $c_d(v, w) \geq 0$ для всех $(v, w) \in E$.

$\square$

Функция расстояний: доказательство теоремы

Доказательство.

- Докажем обратное. Пусть функция d расстояний дерева T удовлетворяет условию $c_d(v, w) \geq 0$ для всех $(v, w) \in E$.
- Для любого цикла Γ графа G имеем $c(\Gamma) = c_d(\Gamma) \geq 0$.
- Пусть $s = v_0, v_1, \ldots, v_k = v$ есть кратчайший путь из s в v.
- Так как G не имеет циклов отрицательной стоимости, то стоимость кратчайшего пути из s в s равна $0 = d(s)$.
- По индукции допустим, что $d(v_{k-1}) = \sigma(s, v_{k-1})$.
- Так как $d(v) \leq d(v_{k-1}) + c(v_{k-1}, v) = \sigma(s, v) \leq d(v)$,
- то $d(v)$ — также стоимость кратчайшего пути из s в v.

$\square$

Функция расстояний: доказательство теоремы

Доказательство.

- Докажем обратное. Пусть функция d расстояний дерева T удовлетворяет условию $c_d(v,w) \geq 0$ для всех $(v,w) \in E$.
- Для любого цикла Γ графа G имеем $c(\Gamma) = c_d(\Gamma) \geq 0$.
- Пусть $s = v_0, v_1, \ldots, v_k = v$ есть кратчайший путь из s в v.
- Так как G не имеет циклов отрицательной стоимости, то стоимость кратчайшего пути из s в s равна $0 = d(s)$.
- По индукции допустим, что $d(v_{k-1}) = \sigma(s, v_{k-1})$.
- Так как $d(v) \leq d(v_{k-1}) + c(v_{k-1}, v) = \sigma(s,v) \leq d(v)$,
- то $d(v)$ — также стоимость кратчайшего пути из s в v.

□

Функция расстояний: доказательство теоремы

Доказательство.

- Докажем обратное. Пусть функция d расстояний дерева T удовлетворяет условию $c_d(v, w) \geq 0$ для всех $(v, w) \in E$.
- Для любого цикла Γ графа G имеем $c(\Gamma) = c_d(\Gamma) \geq 0$.
- Пусть $s = v_0, v_1, \ldots, v_k = v$ есть кратчайший путь из s в v.
- Так как G не имеет циклов отрицательной стоимости, то стоимость кратчайшего пути из s в s равна $0 = d(s)$.
- По индукции допустим, что $d(v_{k-1}) = \sigma(s, v_{k-1})$.
- Так как $d(v) \leq d(v_{k-1}) + c(v_{k-1}, v) = \sigma(s, v) \leq d(v)$,
- то $d(v)$ — также стоимость кратчайшего пути из s в v.

□

Функция расстояний: доказательство теоремы

Доказательство.

- Докажем обратное. Пусть функция d расстояний дерева T удовлетворяет условию $c_d(v,w) \geq 0$ для всех $(v,w) \in E$.
- Для любого цикла Γ графа G имеем $c(\Gamma) = c_d(\Gamma) \geq 0$.
- Пусть $s = v_0, v_1, \ldots, v_k = v$ есть кратчайший путь из s в v.
- Так как G не имеет циклов отрицательной стоимости, то стоимость кратчайшего пути из s в s равна $0 = d(s)$.
- По индукции допустим, что $d(v_{k-1}) = \sigma(s, v_{k-1})$.
- Так как $d(v) \leq d(v_{k-1}) + c(v_{k-1}, v) = \sigma(s,v) \leq d(v)$,
- то $d(v)$ — также стоимость кратчайшего пути из s в v.

$\square$

Функция расстояний: доказательство теоремы

Доказательство.

- Докажем обратное. Пусть функция d расстояний дерева T удовлетворяет условию $c_d(v,w) \geq 0$ для всех $(v,w) \in E$.
- Для любого цикла Γ графа G имеем $c(\Gamma) = c_d(\Gamma) \geq 0$.
- Пусть $s = v_0, v_1, \ldots, v_k = v$ есть кратчайший путь из s в v.
- Так как G не имеет циклов отрицательной стоимости, то стоимость кратчайшего пути из s в s равна $0 = d(s)$.
- По индукции допустим, что $d(v_{k-1}) = \sigma(s, v_{k-1})$.
- Так как $d(v) \leq d(v_{k-1}) + c(v_{k-1}, v) = \sigma(s,v) \leq d(v)$,
- то $d(v)$ — также стоимость кратчайшего пути из s в v.

$\square$

Функция расстояний: доказательство теоремы

Доказательство.

- Докажем обратное. Пусть функция d расстояний дерева T удовлетворяет условию $c_d(v, w) \geq 0$ для всех $(v, w) \in E$.
- Для любого цикла Γ графа G имеем $c(\Gamma) = c_d(\Gamma) \geq 0$.
- Пусть $s = v_0, v_1, \ldots, v_k = v$ есть кратчайший путь из s в v.
- Так как G не имеет циклов отрицательной стоимости, то стоимость кратчайшего пути из s в s равна $0 = d(s)$.
- По индукции допустим, что $d(v_{k-1}) = \sigma(s, v_{k-1})$.
- Так как $d(v) \leq d(v_{k-1}) + c(v_{k-1}, v) = \sigma(s, v) \leq d(v)$,
- то $d(v)$ — также стоимость кратчайшего пути из s в v.

□

Критерий отсутствия отрицательных циклов

Следствие 3

Орграф $G = (V, E)$ не имеет циклов отрицательной стоимости тогда и только тогда, когда существует функция цен $p : V \to \mathbb{R}$, что $c_p(v, w) \geq 0$ для всех $(v, w) \in E$.

Доказательство.

- Достаточность: $c(\Gamma) = c_p(\Gamma) \geq 0$ для любого цикла Γ.
- Необходимость. Построим вспомогательный орграф G_{aux},
- добавляя к G новую вершину s и множество дуг, которые выходят из s во все остальные вершины.
- Стоимости новых дуг определим равными нулю.
- Если G не имеет отр. циклов, то их нет и в G_{aux}.
- G_{aux} имеет дерево кратч. путей, функция p расстояний которого удовл. условию $c_p(v, w) \geq 0$ для всех $(v, w) \in E$.

Критерий отсутствия отрицательных циклов

Следствие 3

Орграф $G = (V, E)$ не имеет циклов отрицательной стоимости тогда и только тогда, когда существует функция цен $p : V \to \mathbb{R}$, что $c_p(v, w) \geq 0$ для всех $(v, w) \in E$.

Доказательство.

- Достаточность: $c(\Gamma) = c_p(\Gamma) \geq 0$ для любого цикла Γ.
- Необходимость. Построим вспомогательный орграф G_{aux},
- добавляя к G новую вершину s и множество дуг, которые выходят из s во все остальные вершины.
- Стоимости новых дуг определим равными нулю.
- Если G не имеет отр. циклов, то их нет и в G_{aux}.
- G_{aux} имеет дерево кратч. путей, функция p расстояний которого удовл. условию $c_p(v, w) \geq 0$ для всех $(v, w) \in E$.

Критерий отсутствия отрицательных циклов

Следствие 3

Орграф $G = (V, E)$ не имеет циклов отрицательной стоимости тогда и только тогда, когда существует функция цен $p : V \to \mathbb{R}$, что $c_p(v, w) \geq 0$ для всех $(v, w) \in E$.

Доказательство.

- Достаточность: $c(\Gamma) = c_p(\Gamma) \geq 0$ для любого цикла Γ.
- Необходимость. Построим вспомогательный орграф G_{aux},
- добавляя к G новую вершину s и множество дуг, которые выходят из s во все остальные вершины.
- Стоимости новых дуг определим равными нулю.
- Если G не имеет отр. циклов, то их нет и в G_{aux}.
- G_{aux} имеет дерево кратч. путей, функция p расстояний которого удовл. условию $c_p(v, w) \geq 0$ для всех $(v, w) \in E$.

Критерий отсутствия отрицательных циклов

Следствие 3

Орграф $G = (V, E)$ не имеет циклов отрицательной стоимости тогда и только тогда, когда существует функция цен $p : V \to \mathbb{R}$, что $c_p(v, w) \geq 0$ для всех $(v, w) \in E$.

Доказательство.

- Достаточность: $c(\Gamma) = c_p(\Gamma) \geq 0$ для любого цикла Γ.
- Необходимость. Построим вспомогательный орграф G_{aux},
- добавляя к G новую вершину s и множество дуг, которые выходят из s во все остальные вершины.
- Стоимости новых дуг определим равными нулю.
- Если G не имеет отр. циклов, то их нет и в G_{aux}.
- G_{aux} имеет дерево кратч. путей, функция p расстояний которого удовл. условию $c_p(v, w) \geq 0$ для всех $(v, w) \in E$.

Критерий отсутствия отрицательных циклов

Следствие 3

Орграф $G = (V, E)$ не имеет циклов отрицательной стоимости тогда и только тогда, когда существует функция цен $p : V \to \mathbb{R}$, что $c_p(v, w) \geq 0$ для всех $(v, w) \in E$.

Доказательство.

- Достаточность: $c(\Gamma) = c_p(\Gamma) \geq 0$ для любого цикла Γ.
- Необходимость. Построим вспомогательный орграф G_{aux},
- добавляя к G новую вершину s и множество дуг, которые выходят из s во все остальные вершины.
- Стоимости новых дуг определим равными нулю.
- Если G не имеет отр. циклов, то их нет и в G_{aux}.
- G_{aux} имеет дерево кратч. путей, функция p расстояний которого удовл. условию $c_p(v, w) \geq 0$ для всех $(v, w) \in E$.

Критерий отсутствия отрицательных циклов

Следствие 3

Орграф $G = (V, E)$ не имеет циклов отрицательной стоимости тогда и только тогда, когда существует функция цен $p : V \to \mathbb{R}$, что $c_p(v, w) \geq 0$ для всех $(v, w) \in E$.

Доказательство.

- Достаточность: $c(\Gamma) = c_p(\Gamma) \geq 0$ для любого цикла Γ.
- Необходимость. Построим вспомогательный орграф G_{aux},
- добавляя к G новую вершину s и множество дуг, которые выходят из s во все остальные вершины.
- Стоимости новых дуг определим равными нулю.
- Если G не имеет отр. циклов, то их нет и в G_{aux}.
- G_{aux} имеет дерево кратч. путей, функция p расстояний которого удовл. условию $c_p(v, w) \geq 0$ для всех $(v, w) \in E$.

План лекции

1. Дерево кратчайших путей
   - Принцип оптимальности
   - Критерий существования дерева кратчайших путей
2. Алгоритмы поиска кратчайших путей
   - Алгоритм Форда — Беллмана
   - Алгоритм Дейкстры
   - Кратчайшие пути в ациклических графах

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{\mathbf{nil}\}$, таких, что

$$d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$$

$$parent(v) = \mathbf{nil}, \quad v \in V.$$

- Затем итеративно повторяется следующий шаг:
  - выбрать дугу (v, w) отрицательной приведенной стоимости $c_d(v, w) < 0$,
  - положить $parent(w) = v$ и заменить $d(w)$ на $d(v) + c(v, w)$.
- Это есть *метод последовательной аппроксимации.*

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{\mathbf{nil}\}$, таких, что

$$d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$$

$$parent(v) = \mathbf{nil}, \quad v \in V.$$

- Затем итеративно повторяется следующий шаг:
  - выбрать дугу (v, w) отрицательной приведенной стоимости $c_d(v, w) < 0$,
  - положить $parent(w) = v$ и заменить $d(w)$ на $d(v) + c(v, w)$.
- Это есть *метод последовательной аппроксимации*.

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{\mathbf{nil}\}$, таких, что

$$d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$$

$$parent(v) = \mathbf{nil}, \quad v \in V.$$

- Затем итеративно повторяется следующий шаг:
  - выбрать дугу (v, w) отрицательной приведенной стоимости $c_d(v, w) < 0$,
  - положить $parent(w) = v$ и заменить $d(w)$ на $d(v) + c(v, w)$.
- Это есть *метод последовательной аппроксимации.*

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{\mathbf{nil}\}$, таких, что

$$d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$$

$$parent(v) = \mathbf{nil}, \quad v \in V.$$

- Затем итеративно повторяется следующий шаг:
  - выбрать дугу (v, w) отрицательной приведенной стоимости $c_d(v, w) < 0$,
  - положить $parent(w) = v$ и заменить $d(w)$ на $d(v) + c(v, w)$.
- Это есть *метод последовательной аппроксимации*.

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{\mathbf{nil}\}$, таких, что

$$d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$$

$$parent(v) = \mathbf{nil}, \quad v \in V.$$

- Затем итеративно повторяется следующий шаг:
  - выбрать дугу (v, w) отрицательной приведенной стоимости $c_d(v, w) < 0$,
  - положить $parent(w) = v$ и заменить $d(w)$ на $d(v) + c(v, w)$.
- Это есть *метод последовательной аппроксимации.*

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{\mathbf{nil}\}$, таких, что

$$d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$$
$$parent(v) = \mathbf{nil}, \quad v \in V.$$

- Затем итеративно повторяется следующий шаг:
  - выбрать дугу (v, w) отрицательной приведенной стоимости $c_d(v, w) < 0$,
  - положить $parent(w) = v$ и заменить $d(w)$ на $d(v) + c(v, w)$.
- Это есть *метод последовательной аппроксимации.*

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{\mathbf{nil}\}$, таких, что

$$d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$$
$$parent(v) = \mathbf{nil}, \quad v \in V.$$

- Затем итеративно повторяется следующий шаг:
  - выбрать дугу (v, w) отрицательной приведенной стоимости $c_d(v, w) < 0$,
  - положить $parent(w) = v$ и заменить $d(w)$ на $d(v) + c(v, w)$.
- Это есть *метод последовательной аппроксимации.*

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{\mathbf{nil}\}$, таких, что

$$d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$$
$$parent(v) = \mathbf{nil}, \quad v \in V.$$

- Затем итеративно повторяется следующий шаг:
  - выбрать дугу (v, w) отрицательной приведенной стоимости $c_d(v, w) < 0$,
  - положить $parent(w) = v$ и заменить $d(w)$ на $d(v) + c(v, w)$.
- Это есть *метод последовательной аппроксимации.*

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{\mathbf{nil}\}$, таких, что

$$d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$$
$$parent(v) = \mathbf{nil}, \quad v \in V.$$

- Затем итеративно повторяется следующий шаг:
  - выбрать дугу (v, w) отрицательной приведенной стоимости $c_d(v, w) < 0$,
  - положить $parent(w) = v$ и заменить $d(w)$ на $d(v) + c(v, w)$.
- Это есть *метод последовательной аппроксимации.*

Свойства метода последовательной аппроксимации

- Если в графе нет циклов отрицательной стоимости, то
- метод заканчивает работу после конечного числа итераций, когда стоимости дуг целочисленны.
- При этом $d(v) = \sigma(s, v)$ для всех $v \in V$, а указатели *parent* задают дерево кратчайших путей.
- Если в графе есть цикл отрицательной стоимости, достижимый из вершины s,
- то метод должен остановиться, как только граф, составленый из дуг $(parent(v), v)$ с $parent(v) \neq \mathbf{nil}$, содержит цикл.
- Из описания метода последовательной аппроксимации следует, что стоимость этого цикла отрицательна.
- Эффективность метода зависит от порядка выбора дуг отрицательной приведенной стоимости.

Свойства метода последовательной аппроксимации

- Если в графе нет циклов отрицательной стоимости, то
- метод заканчивает работу после конечного числа итераций, когда стоимости дуг целочисленны.
- При этом $d(v) = \sigma(s, v)$ для всех $v \in V$, а указатели *parent* задают дерево кратчайших путей.
- Если в графе есть цикл отрицательной стоимости, достижимый из вершины s,
- то метод должен остановиться, как только граф, составленый из дуг $(parent(v), v)$ с $parent(v) \neq \mathbf{nil}$, содержит цикл.
- Из описания метода последовательной аппроксимации следует, что стоимость этого цикла отрицательна.
- Эффективность метода зависит от порядка выбора дуг отрицательной приведенной стоимости.

Свойства метода последовательной аппроксимации

- Если в графе нет циклов отрицательной стоимости, то
- метод заканчивает работу после конечного числа итераций, когда стоимости дуг целочисленны.
- При этом $d(v) = \sigma(s, v)$ для всех $v \in V$, а указатели *parent* задают дерево кратчайших путей.
- Если в графе есть цикл отрицательной стоимости, достижимый из вершины s,
- то метод должен остановиться, как только граф, составленый из дуг $(parent(v), v)$ с $parent(v) \neq \textbf{nil}$, содержит цикл.
- Из описания метода последовательной аппроксимации следует, что стоимость этого цикла отрицательна.
- Эффективность метода зависит от порядка выбора дуг отрицательной приведенной стоимости.

Свойства метода последовательной аппроксимации

- Если в графе нет циклов отрицательной стоимости, то
- метод заканчивает работу после конечного числа итераций, когда стоимости дуг целочисленны.
- При этом $d(v) = \sigma(s, v)$ для всех $v \in V$, а указатели *parent* задают дерево кратчайших путей.
- Если в графе есть цикл отрицательной стоимости, достижимый из вершины s,
- то метод должен остановиться, как только граф, составленый из дуг $(parent(v), v)$ с $parent(v) \neq \mathbf{nil}$, содержит цикл.
- Из описания метода последовательной аппроксимации следует, что стоимость этого цикла отрицательна.
- Эффективность метода зависит от порядка выбора дуг отрицательной приведенной стоимости.

Свойства метода последовательной аппроксимации

- Если в графе нет циклов отрицательной стоимости, то
- метод заканчивает работу после конечного числа итераций, когда стоимости дуг целочисленны.
- При этом $d(v) = \sigma(s, v)$ для всех $v \in V$, а указатели *parent* задают дерево кратчайших путей.
- Если в графе есть цикл отрицательной стоимости, достижимый из вершины s,
- то метод должен остановиться, как только граф, составленый из дуг $(parent(v), v)$ с $parent(v) \neq \textbf{nil}$, содержит цикл.
- Из описания метода последовательной аппроксимации следует, что стоимость этого цикла отрицательна.
- Эффективность метода зависит от порядка выбора дуг отрицательной приведенной стоимости.

Свойства метода последовательной аппроксимации

- Если в графе нет циклов отрицательной стоимости, то
- метод заканчивает работу после конечного числа итераций, когда стоимости дуг целочисленны.
- При этом $d(v) = \sigma(s, v)$ для всех $v \in V$, а указатели *parent* задают дерево кратчайших путей.
- Если в графе есть цикл отрицательной стоимости, достижимый из вершины s,
- то метод должен остановиться, как только граф, составленый из дуг $(parent(v), v)$ с $parent(v) \neq \textbf{nil}$, содержит цикл.
- Из описания метода последовательной аппроксимации следует, что стоимость этого цикла отрицательна.
- Эффективность метода зависит от порядка выбора дуг отрицательной приведенной стоимости.

Свойства метода последовательной аппроксимации

- Если в графе нет циклов отрицательной стоимости, то
- метод заканчивает работу после конечного числа итераций, когда стоимости дуг целочисленны.
- При этом $d(v) = \sigma(s, v)$ для всех $v \in V$, а указатели *parent* задают дерево кратчайших путей.
- Если в графе есть цикл отрицательной стоимости, достижимый из вершины s,
- то метод должен остановиться, как только граф, составленый из дуг $(parent(v), v)$ с $parent(v) \neq \mathbf{nil}$, содержит цикл.
- Из описания метода последовательной аппроксимации следует, что стоимость этого цикла отрицательна.
- Эффективность метода зависит от порядка выбора дуг отрицательной приведенной стоимости.

Описание алгоритма Форда - Беллмана

- **Вход**: Орграф $G = (V, E)$, ф-ция $c : E \to \mathbb{R}$, верш. $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, функция $d : V \to \mathbb{R} \cup \{\infty\}$:
  1. если указатели $parent$ представляют дерево, то $d(v)$ — *кратчайшее расстояние* от s до v;
  2. в противном случае, любой цикл в графе, опред. указателями $parent$, является *отрицательным циклом*.
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) =$**nil**.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.
- 3. Для $i = 1, \dots, n$ выполнить шаги 3.1–3.4:
  - 3.1. $Q = \emptyset$, $\bar{d} = d$.
  - 3.2. Для $(v, w) \in E(S, V)$, таких, что $d(w) > \bar{d}(v) + c(v, w)$, положить $d(w) := \bar{d}(v) + c(v, w)$, $parent(w) = v$, $Q := Q \cup \{w\}$.
  - 3.3. Если $Q = \emptyset$, вернуть **истина**; иначе положить $S = Q$.
- 4. Вернуть **ложь**.

Описание алгоритма Форда - Беллмана

- **Вход**: Орграф $G = (V, E)$, ф-ция $c : E \to \mathbb{R}$, верш. $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, функция $d : V \to \mathbb{R} \cup \{\infty\}$:
  1. если указатели *parent* представляют дерево, то $d(v)$ — *кратчайшее расстояние* от s до v;
  2. в противном случае, любой цикл в графе, опред. указателями *parent*, является *отрицательным циклом.*
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) =$**nil**.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.
- 3. Для $i = 1, \dots, n$ выполнить шаги 3.1–3.4:
  - 3.1. $Q = \emptyset$, $\bar{d} = d$.
  - 3.2. Для $(v, w) \in E(S, V)$, таких, что $d(w) > \bar{d}(v) + c(v, w)$, положить $d(w) := \bar{d}(v) + c(v, w)$, $parent(w) = v$, $Q := Q \cup \{w\}$.
  - 3.3. Если $Q = \emptyset$, вернуть **истина**; иначе положить $S = Q$.
- 4. Вернуть **ложь**.

Описание алгоритма Форда - Беллмана

- **Вход**: Орграф $G = (V, E)$, ф-ция $c : E \to \mathbb{R}$, верш. $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, функция $d : V \to \mathbb{R} \cup \{\infty\}$:
  1. если указатели $parent$ представляют дерево, то $d(v)$ — *кратчайшее расстояние* от s до v;
  2. в противном случае, любой цикл в графе, опред. указателями $parent$, является *отрицательным циклом.*
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) = $**nil**.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.
- 3. Для $i = 1, \dots, n$ выполнить шаги 3.1–3.4:
  - 3.1. $Q = \emptyset$, $\bar{d} = d$.
  - 3.2. Для $(v, w) \in E(S, V)$, таких, что $d(w) > \bar{d}(v) + c(v, w)$, положить $d(w) := \bar{d}(v) + c(v, w)$, $parent(w) = v$, $Q := Q \cup \{w\}$.
  - 3.3. Если $Q = \emptyset$, вернуть **истина**; иначе положить $S = Q$.
- 4. Вернуть **ложь**.

Описание алгоритма Форда - Беллмана

- **Вход**: Орграф $G = (V, E)$, ф-ция $c : E \to \mathbb{R}$, верш. $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, функция $d : V \to \mathbb{R} \cup \{\infty\}$:
  1. если указатели $parent$ представляют дерево, то $d(v)$ — *кратчайшее расстояние* от s до v;
  2. в противном случае, любой цикл в графе, опред. указателями $parent$, является *отрицательным циклом*.
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) = \mathbf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.
- 3. Для $i = 1, \dots, n$ выполнить шаги 3.1–3.4:
  - 3.1. $Q = \emptyset$, $\bar{d} = d$.
  - 3.2. Для $(v, w) \in E(S, V)$, таких, что $d(w) > \bar{d}(v) + c(v, w)$, положить $d(w) := \bar{d}(v) + c(v, w)$, $parent(w) = v$, $Q := Q \cup \{w\}$.
  - 3.3. Если $Q = \emptyset$, вернуть **истина**; иначе положить $S = Q$.
- 4. Вернуть **ложь**.

Описание алгоритма Форда - Беллмана

- **Вход**: Орграф $G = (V, E)$, ф-ция $c : E \to \mathbb{R}$, верш. $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, функция $d : V \to \mathbb{R} \cup \{\infty\}$:
  1. если указатели $parent$ представляют дерево, то $d(v)$ — *кратчайшее расстояние* от s до v;
  2. в противном случае, любой цикл в графе, опред. указателями $parent$, является *отрицательным циклом*.
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) =$**nil**.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.
- 3. Для $i = 1, \dots, n$ выполнить шаги 3.1–3.4:
  - 3.1. $Q = \emptyset$, $\bar{d} = d$.
  - 3.2. Для $(v, w) \in E(S, V)$, таких, что $d(w) > \bar{d}(v) + c(v, w)$, положить $d(w) := \bar{d}(v) + c(v, w)$, $parent(w) = v$, $Q := Q \cup \{w\}$.
  - 3.3. Если $Q = \emptyset$, вернуть **истина**; иначе положить $S = Q$.
- 4. Вернуть **ложь**.

Описание алгоритма Форда - Беллмана

- **Вход**: Орграф $G = (V, E)$, ф-ция $c : E \to \mathbb{R}$, верш. $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, функция $d : V \to \mathbb{R} \cup \{\infty\}$:
  1. если указатели *parent* представляют дерево, то $d(v)$ — *кратчайшее расстояние* от s до v;
  2. в противном случае, любой цикл в графе, опред. указателями *parent*, является *отрицательным циклом*.
- 1\. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) =$**nil**.
- 2\. Положить $d(s) = 0$ и $S = \{s\}$.
- 3\. Для $i = 1, \ldots, n$ выполнить шаги 3.1–3.4:
  - 3.1. $Q = \emptyset$, $\bar{d} = d$.
  - 3.2. Для $(v, w) \in E(S, V)$, таких, что $d(w) > \bar{d}(v) + c(v, w)$, положить $d(w) := \bar{d}(v) + c(v, w)$, $parent(w) = v$, $Q := Q \cup \{w\}$.
  - 3.3. Если $Q = \emptyset$, вернуть **истина**; иначе положить $S = Q$.
- 4\. Вернуть **ложь**.

Описание алгоритма Форда - Беллмана

- **Вход**: Орграф $G = (V, E)$, ф-ция $c : E \to \mathbb{R}$, верш. $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, функция $d : V \to \mathbb{R} \cup \{\infty\}$:
  1. если указатели $parent$ представляют дерево, то $d(v)$ — *кратчайшее расстояние* от s до v;
  2. в противном случае, любой цикл в графе, опред. указателями $parent$, является *отрицательным циклом*.
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) =$**nil**.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.
- 3. Для $i = 1, \ldots, n$ выполнить шаги 3.1–3.4:
  - 3.1. $Q = \emptyset$, $\bar{d} = d$.
  - 3.2. Для $(v, w) \in E(S, V)$, таких, что $d(w) > \bar{d}(v) + c(v, w)$, положить $d(w) := \bar{d}(v) + c(v, w)$, $parent(w) = v$, $Q := Q \cup \{w\}$.
  - 3.3. Если $Q = \emptyset$, вернуть **истина**; иначе положить $S = Q$.
- 4. Вернуть **ложь**.

Описание алгоритма Форда - Беллмана

- **Вход**: Орграф $G = (V, E)$, ф-ция $c : E \to \mathbb{R}$, верш. $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, функция $d : V \to \mathbb{R} \cup \{\infty\}$:
  1. если указатели $parent$ представляют дерево, то $d(v)$ — *кратчайшее расстояние* от s до v;
  2. в противном случае, любой цикл в графе, опред. указателями $parent$, является *отрицательным циклом*.
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) =$**nil**.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.
- 3. Для $i = 1, \dots, n$ выполнить шаги 3.1–3.4:
  - 3.1. $Q = \emptyset$, $\bar{d} = d$.
  - 3.2. Для $(v, w) \in E(S, V)$, таких, что $d(w) > \bar{d}(v) + c(v, w)$, положить $d(w) := \bar{d}(v) + c(v, w)$, $parent(w) = v$, $Q := Q \cup \{w\}$.
  - 3.3. Если $Q = \emptyset$, вернуть **истина**; иначе положить $S = Q$.
- 4. Вернуть **ложь**.

Описание алгоритма Форда - Беллмана

- **Вход**: Орграф $G = (V, E)$, ф-ция $c : E \to \mathbb{R}$, верш. $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, функция $d : V \to \mathbb{R} \cup \{\infty\}$:
  1. если указатели *parent* представляют дерево, то $d(v)$ — *кратчайшее расстояние* от s до v;
  2. в противном случае, любой цикл в графе, опред. указателями *parent*, является *отрицательным циклом*.
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) =$**nil**.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.
- 3. Для $i = 1, \dots, n$ выполнить шаги 3.1–3.4:
  - 3.1. $Q = \emptyset$, $\bar{d} = d$.
  - 3.2. Для $(v, w) \in E(S, V)$, таких, что $d(w) > \bar{d}(v) + c(v, w)$, положить $d(w) := \bar{d}(v) + c(v, w)$, $parent(w) = v$, $Q := Q \cup \{w\}$.
  - 3.3. Если $Q = \emptyset$, вернуть **истина**; иначе положить $S = Q$.
- 4. Вернуть **ложь**.

Описание алгоритма Форда - Беллмана

- **Вход**: Орграф $G = (V, E)$, ф-ция $c : E \to \mathbb{R}$, верш. $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, функция $d : V \to \mathbb{R} \cup \{\infty\}$:
  1. если указатели $parent$ представляют дерево, то $d(v)$ — *кратчайшее расстояние* от s до v;
  2. в противном случае, любой цикл в графе, опред. указателями $parent$, является *отрицательным циклом*.
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) =$**nil**.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.
- 3. Для $i = 1, \dots, n$ выполнить шаги 3.1–3.4:
  - 3.1. $Q = \emptyset$, $\bar{d} = d$.
  - 3.2. Для $(v, w) \in E(S, V)$, таких, что $d(w) > \bar{d}(v) + c(v, w)$, положить $d(w) := \bar{d}(v) + c(v, w)$, $parent(w) = v$, $Q := Q \cup \{w\}$.
  - 3.3. Если $Q = \emptyset$, вернуть **истина**; иначе положить $S = Q$.
- 4. Вернуть **ложь**.

Описание алгоритма Форда - Беллмана

- **Вход**: Орграф $G = (V, E)$, ф-ция $c : E \to \mathbb{R}$, верш. $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, функция $d : V \to \mathbb{R} \cup \{\infty\}$:
  1. если указатели *parent* представляют дерево, то $d(v)$ — *кратчайшее расстояние* от s до v;
  2. в противном случае, любой цикл в графе, опред. указателями *parent*, является *отрицательным циклом*.
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) =$**nil**.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.
- 3. Для $i = 1, \ldots, n$ выполнить шаги 3.1–3.4:
  - 3.1. $Q = \emptyset$, $\bar{d} = d$.
  - 3.2. Для $(v, w) \in E(S, V)$, таких, что $d(w) > \bar{d}(v) + c(v, w)$, положить $d(w) := \bar{d}(v) + c(v, w)$, $parent(w) = v$, $Q := Q \cup \{w\}$.
  - 3.3. Если $Q = \emptyset$, вернуть **истина**; иначе положить $S = Q$.
- 4. Вернуть **ложь**.

Анализ алгоритма Форда - Беллмана

- Обозначим через $\sigma^i(s, v)$ стоимость кратчайшего пути из s в v среди всех путей, которые имеют ровно i дуг.
- Если такого пути не существует, то $\sigma^i(s, v) = \infty$.
- Пусть $d^i(v)$, S^i — соответственно $d(v)$ и список S после итерации i.
- Этап инициализации (шаги 1 и 2) называем 0-й итерацией.

Анализ алгоритма Форда - Беллмана

- Обозначим через $\sigma^i(s,v)$ стоимость кратчайшего пути из s в v среди всех путей, которые имеют ровно i дуг.
- Если такого пути не существует, то $\sigma^i(s,v) = \infty$.
- Пусть $d^i(v)$, S^i — соответственно $d(v)$ и список S после итерации i.
- Этап инициализации (шаги 1 и 2) называем 0-й итерацией.

Анализ алгоритма Форда - Беллмана

- Обозначим через $\sigma^i(s, v)$ стоимость кратчайшего пути из s в v среди всех путей, которые имеют ровно i дуг.
- Если такого пути не существует, то $\sigma^i(s, v) = \infty$.
- Пусть $d^i(v)$, S^i — соответственно $d(v)$ и список S после итерации i.
- Этап инициализации (шаги 1 и 2) называем 0-й итерацией.

Анализ алгоритма Форда - Беллмана

- Обозначим через $\sigma^i(s,v)$ стоимость кратчайшего пути из s в v среди всех путей, которые имеют ровно i дуг.
- Если такого пути не существует, то $\sigma^i(s,v) = \infty$.
- Пусть $d^i(v)$, S^i — соответственно $d(v)$ и список S после итерации i.
- Этап инициализации (шаги 1 и 2) называем 0-й итерацией.

Свойства функции расстояний

Лемма 4

Справедливы следующие соотношения:

(i) $d^i(v) = \min\limits_{0 \le k \le i} \sigma^k(s, v)$ *для всех* $v \in V$,

(ii) $d^i(v) = \sigma^i(s, v) < \sigma^{i-1}(s, v)$ *для всех* $v \in S^i$,

Доказательство.

- Утверждение леммы справедливо после 0-й итерации.
- Допустим, что после завершения $(i - 1)$-й (перед началом i-й) итерации алгоритма

$$d^{i-1}(v) = \min_{0 \le k \le i-1} \sigma^k(s, v) \text{ для всех } v \in V,$$

$$d^{i-1}(v) = \sigma^{i-1}(s, v) \text{ для всех } v \in S^{i-1}.$$

□

Свойства функции расстояний

Лемма 4

Справедливы следующие соотношения:

(i) $d^i(v) = \min_{0\le k\le i} \sigma^k(s,v)$ *для всех* $v \in V$,

(ii) $d^i(v) = \sigma^i(s,v) < \sigma^{i-1}(s,v)$ *для всех* $v \in S^i$,

Доказательство.

- Утверждение леммы справедливо после 0-й итерации.
- Допустим, что после завершения $(i-1)$-й (перед началом i-й) итерации алгоритма

$$d^{i-1}(v) = \min_{0\le k\le i-1} \sigma^k(s,v) \text{ для всех } v \in V,$$

$$d^{i-1}(v) = \sigma^{i-1}(s,v) \text{ для всех } v \in S^{i-1}.$$

$\square$

Свойства функции расстояний

Лемма 4

Справедливы следующие соотношения:

(i) $d^i(v) = \min_{0 \le k \le i} \sigma^k(s, v)$ *для всех* $v \in V$,

(ii) $d^i(v) = \sigma^i(s, v) < \sigma^{i-1}(s, v)$ *для всех* $v \in S^i$,

Доказательство.

- Утверждение леммы справедливо после 0-й итерации.
- Допустим, что после завершения $(i-1)$-й (перед началом i-й) итерации алгоритма

$$d^{i-1}(v) = \min_{0 \le k \le i-1} \sigma^k(s, v) \text{ для всех } v \in V,$$

$$d^{i-1}(v) = \sigma^{i-1}(s, v) \text{ для всех } v \in S^{i-1}.$$

□

Свойства функции расстояний

Лемма 4

Справедливы следующие соотношения:

(i) $d^i(v) = \min\limits_{0 \le k \le i} \sigma^k(s,v)$ *для всех* $v \in V$,

(ii) $d^i(v) = \sigma^i(s,v) < \sigma^{i-1}(s,v)$ *для всех* $v \in S^i$,

Доказательство.

- Утверждение леммы справедливо после 0-й итерации.
- Допустим, что после завершения $(i-1)$-й (перед началом i-й) итерации алгоритма

$$d^{i-1}(v) = \min_{0 \le k \le i-1} \sigma^k(s,v) \text{ для всех } v \in V,$$

$$d^{i-1}(v) = \sigma^{i-1}(s,v) \text{ для всех } v \in S^{i-1}.$$

□

Свойства функции расстояний

Лемма 4

Справедливы следующие соотношения:

(i) $d^i(v) = \min\limits_{0 \le k \le i} \sigma^k(s, v)$ *для всех* $v \in V$,

(ii) $d^i(v) = \sigma^i(s, v) < \sigma^{i-1}(s, v)$ *для всех* $v \in S^i$,

Доказательство.

- Утверждение леммы справедливо после 0-й итерации.
- Допустим, что после завершения $(i-1)$-й (перед началом i-й) итерации алгоритма

$$d^{i-1}(v) = \min_{0 \le k \le i-1} \sigma^k(s, v) \text{ для всех } v \in V,$$

$$d^{i-1}(v) = \sigma^{i-1}(s, v) \text{ для всех } v \in S^{i-1}.$$

□

Свойства функции расстояний

Лемма 4

Справедливы следующие соотношения:

(i) $d^i(v) = \min\limits_{0 \le k \le i} \sigma^k(s, v)$ *для всех* $v \in V$,

(ii) $d^i(v) = \sigma^i(s, v) < \sigma^{i-1}(s, v)$ *для всех* $v \in S^i$,

Доказательство.

- Утверждение леммы справедливо после 0-й итерации.
- Допустим, что после завершения $(i-1)$-й (перед началом i-й) итерации алгоритма

$$d^{i-1}(v) = \min_{0 \le k \le i-1} \sigma^k(s, v) \text{ для всех } v \in V,$$

$$d^{i-1}(v) = \sigma^{i-1}(s, v) \text{ для всех } v \in S^{i-1}.$$

□

Свойства функции расстояний

Лемма 4

Справедливы следующие соотношения:

(i) $d^i(v) = \min\limits_{0 \le k \le i} \sigma^k(s,v)$ *для всех* $v \in V$,

(ii) $d^i(v) = \sigma^i(s,v) < \sigma^{i-1}(s,v)$ *для всех* $v \in S^i$,

Доказательство.

- Утверждение леммы справедливо после 0-й итерации.
- Допустим, что после завершения $(i-1)$-й (перед началом i-й) итерации алгоритма

$$d^{i-1}(v) = \min_{0 \le k \le i-1} \sigma^k(s,v) \text{ для всех } v \in V,$$

$$d^{i-1}(v) = \sigma^{i-1}(s,v) \text{ для всех } v \in S^{i-1}.$$

□

Свойства функции расстояний

Лемма 4

Справедливы следующие соотношения:

(i) $d^i(v) = \min_{0 \le k \le i} \sigma^k(s, v)$ *для всех* $v \in V$,

(ii) $d^i(v) = \sigma^i(s, v) < \sigma^{i-1}(s, v)$ *для всех* $v \in S^i$,

Доказательство.

- Пусть $s = v_0, v_1, \dots, v_{i-1}, v_i = w$ есть путь стоимости $\sigma^i(s, w)$ в графе G.
- По допущению $d^{i-1}(v_{i-1}) = \sigma^{i-1}(v_{i-1})$.
- Поэтому $v_{i-1} \in S^{i-1}$ и, если $d^{i-1}(w) > d^{i-1}(v_{i-1}) + c(v_{i-1}, w) = \sigma^i(s, w)$,
- то после i-й итерации $d^i(w) = \sigma^i(s, w)$, а вершина w будет включена в S^i.

$\square$

Свойства функции расстояний

Лемма 4

Справедливы следующие соотношения:

(i) $d^i(v) = \min_{0 \le k \le i} \sigma^k(s,v)$ *для всех* $v \in V$,

(ii) $d^i(v) = \sigma^i(s,v) < \sigma^{i-1}(s,v)$ *для всех* $v \in S^i$,

Доказательство.

- Пусть $s = v_0, v_1, \dots, v_{i-1}, v_i = w$ есть путь стоимости $\sigma^i(s,w)$ в графе G.
- По допущению $d^{i-1}(v_{i-1}) = \sigma^{i-1}(v_{i-1})$.
- Поэтому $v_{i-1} \in S^{i-1}$ и, если $d^{i-1}(w) > d^{i-1}(v_{i-1}) + c(v_{i-1}, w) = \sigma^i(s,w)$,
- то после i-й итерации $d^i(w) = \sigma^i(s,w)$, а вершина w будет включена в S^i.

□

Свойства функции расстояний

Лемма 4

Справедливы следующие соотношения:

(i) $d^i(v) = \min_{0 \le k \le i} \sigma^k(s,v)$ *для всех* $v \in V$,

(ii) $d^i(v) = \sigma^i(s,v) < \sigma^{i-1}(s,v)$ *для всех* $v \in S^i$,

Доказательство.

- Пусть $s = v_0, v_1, \dots, v_{i-1}, v_i = w$ есть путь стоимости $\sigma^i(s,w)$ в графе G.
- По допущению $d^{i-1}(v_{i-1}) = \sigma^{i-1}(v_{i-1})$.
- Поэтому $v_{i-1} \in S^{i-1}$ и, если $d^{i-1}(w) > d^{i-1}(v_{i-1}) + c(v_{i-1}, w) = \sigma^i(s,w)$,
- то после i-й итерации $d^i(w) = \sigma^i(s,w)$, а вершина w будет включена в S^i.

□

Свойства функции расстояний

Лемма 4

Справедливы следующие соотношения:

(i) $d^i(v) = \min\limits_{0 \le k \le i} \sigma^k(s,v)$ *для всех* $v \in V$,

(ii) $d^i(v) = \sigma^i(s,v) < \sigma^{i-1}(s,v)$ *для всех* $v \in S^i$,

Доказательство.

- Пусть $s = v_0, v_1, \ldots, v_{i-1}, v_i = w$ есть путь стоимости $\sigma^i(s,w)$ в графе G.
- По допущению $d^{i-1}(v_{i-1}) = \sigma^{i-1}(v_{i-1})$.
- Поэтому $v_{i-1} \in S^{i-1}$ и, если $d^{i-1}(w) > d^{i-1}(v_{i-1}) + c(v_{i-1},w) = \sigma^i(s,w)$,
- то после i-й итерации $d^i(w) = \sigma^i(s,w)$, а вершина w будет включена в S^i.

$\square$

Свойства функции расстояний

Лемма 5

Если после завершения алгоритма Форда — Беллмана $S = \emptyset$, то функция расстояний d удовлетворяет условию $c_d(v,w) \geq 0$ для всех $(v,w) \in E$.

Доказательство.

- Заметим, что $d^i(v) \geq d^{i+1}(v)$ для всех $v \in V$.
- Пусть $(v,w) \in E$.
- Так как $S = \emptyset$, то $d(v) = d^i(v)$ для некоторого $1 \leq i \leq n-1$.
- Поэтому

$$d(w) \leq d^{i+1}(w) \leq d^i(v) + c(v,w) = d(v) + c(v,w).$$

$\square$

Свойства функции расстояний

Лемма 5

Если после завершения алгоритма Форда — Беллмана $S = \emptyset$, то функция расстояний d удовлетворяет условию $c_d(v, w) \geq 0$ для всех $(v, w) \in E$.

Доказательство.

- Заметим, что $d^i(v) \geq d^{i+1}(v)$ для всех $v \in V$.
- Пусть $(v, w) \in E$.
- Так как $S = \emptyset$, то $d(v) = d^i(v)$ для некоторого $1 \leq i \leq n - 1$.
- Поэтому

$$d(w) \leq d^{i+1}(w) \leq d^i(v) + c(v, w) = d(v) + c(v, w).$$

□

Свойства функции расстояний

Лемма 5

Если после завершения алгоритма Форда — Беллмана $S = \emptyset$, то функция расстояний d удовлетворяет условию $c_d(v, w) \geq 0$ для всех $(v, w) \in E$.

Доказательство.

- Заметим, что $d^i(v) \geq d^{i+1}(v)$ для всех $v \in V$.
- Пусть $(v, w) \in E$.
- Так как $S = \emptyset$, то $d(v) = d^i(v)$ для некоторого $1 \leq i \leq n - 1$.
- Поэтому

$$d(w) \leq d^{i+1}(w) \leq d^i(v) + c(v, w) = d(v) + c(v, w).$$

$\square$

Свойства функции расстояний

Лемма 5

Если после завершения алгоритма Форда — Беллмана $S = \emptyset$, то функция расстояний d удовлетворяет условию $c_d(v, w) \geq 0$ для всех $(v, w) \in E$.

Доказательство.

- Заметим, что $d^i(v) \geq d^{i+1}(v)$ для всех $v \in V$.
- Пусть $(v, w) \in E$.
- Так как $S = \emptyset$, то $d(v) = d^i(v)$ для некоторого $1 \leq i \leq n - 1$.
- Поэтому

$$d(w) \leq d^{i+1}(w) \leq d^i(v) + c(v, w) = d(v) + c(v, w).$$

□

Свойства функции расстояний

Лемма 5

Если после завершения алгоритма Форда — Беллмана $S = \emptyset$, то функция расстояний d удовлетворяет условию $c_d(v, w) \geq 0$ для всех $(v, w) \in E$.

Доказательство.

- Заметим, что $d^i(v) \geq d^{i+1}(v)$ для всех $v \in V$.
- Пусть $(v, w) \in E$.
- Так как $S = \emptyset$, то $d(v) = d^i(v)$ для некоторого $1 \leq i \leq n - 1$.
- Поэтому

$$d(w) \leq d^{i+1}(w) \leq d^i(v) + c(v, w) = d(v) + c(v, w).$$

□

Свойства функции расстояний

Лемма 5

Если после завершения алгоритма Форда — Беллмана $S = \emptyset$, то функция расстояний d удовлетворяет условию $c_d(v, w) \geq 0$ для всех $(v, w) \in E$.

Доказательство.

- Заметим, что $d^i(v) \geq d^{i+1}(v)$ для всех $v \in V$.
- Пусть $(v, w) \in E$.
- Так как $S = \emptyset$, то $d(v) = d^i(v)$ для некоторого $1 \leq i \leq n - 1$.
- Поэтому

$$d(w) \leq d^{i+1}(w) \leq d^i(v) + c(v, w) = d(v) + c(v, w).$$

□

Отрицательные циклы

Лемма 6

Граф G имеет цикл отрицательной стоимости тогда и только тогда, когда после завершения алгоритма Форда — Беллмана множество S не пустое.

Доказательство.

- Если $S = \emptyset$, то по теореме об отсутствии отр. циклов и предшествующей лемме граф G не имеет циклов отрицательной стоимости.
- Если $S \neq \emptyset$, то для $v \in S$

$$d^n(v) = \sigma^n(s,v) < \sigma^i(s,v) \text{ для всех } 0 \leq i < n.$$

- Каждый путь из s в v длины n и стоимости $\sigma^n(s,v)$ обходит цикл отрицательной стоимости.

$\square$

Отрицательные циклы

Лемма 6

Граф G имеет цикл отрицательной стоимости тогда и только тогда, когда после завершения алгоритма Форда — Беллмана множество S не пустое.

Доказательство.

- Если $S = \emptyset$, то по теореме об отсутствии отр. циклов и предшествующей лемме граф G не имеет циклов отрицательной стоимости.
- Если $S \neq \emptyset$, то для $v \in S$

$$d^n(v) = \sigma^n(s, v) < \sigma^i(s, v) \text{ для всех } 0 \leq i < n.$$

- Каждый путь из s в v длины n и стоимости $\sigma^n(s, v)$ обходит цикл отрицательной стоимости.

$\square$

Отрицательные циклы

Лемма 6

Граф G имеет цикл отрицательной стоимости тогда и только тогда, когда после завершения алгоритма Форда — Беллмана множество S не пустое.

Доказательство.

- Если $S = \emptyset$, то по теореме об отсутствии отр. циклов и предшествующей лемме граф G не имеет циклов отрицательной стоимости.
- Если $S \neq \emptyset$, то для $v \in S$
$$d^n(v) = \sigma^n(s, v) < \sigma^i(s, v) \text{ для всех } 0 \leq i < n.$$
- Каждый путь из s в v длины n и стоимости $\sigma^n(s, v)$ обходит цикл отрицательной стоимости.

$\square$

Отрицательные циклы

Лемма 6

Граф G имеет цикл отрицательной стоимости тогда и только тогда, когда после завершения алгоритма Форда — Беллмана множество S не пустое.

Доказательство.

- Если $S = \emptyset$, то по теореме об отсутствии отр. циклов и предшествующей лемме граф G не имеет циклов отрицательной стоимости.
- Если $S \neq \emptyset$, то для $v \in S$

$$d^n(v) = \sigma^n(s, v) < \sigma^i(s, v) \text{ для всех } 0 \le i < n.$$

- Каждый путь из s в v длины n и стоимости $\sigma^n(s, v)$ обходит цикл отрицательной стоимости.

$\square$

Отрицательные циклы

Лемма 6

Граф G имеет цикл отрицательной стоимости тогда и только тогда, когда после завершения алгоритма Форда — Беллмана множество S не пустое.

Доказательство.

- Если $S = \emptyset$, то по теореме об отсутствии отр. циклов и предшествующей лемме граф G не имеет циклов отрицательной стоимости.
- Если $S \neq \emptyset$, то для $v \in S$

$$d^n(v) = \sigma^n(s,v) < \sigma^i(s,v) \text{ для всех } 0 \leq i < n.$$

- Каждый путь из s в v длины n и стоимости $\sigma^n(s,v)$ обходит цикл отрицательной стоимости.

□

Сложность алгоритма Форда — Беллмана

Теорема 7

За время $O(nm)$ алгоритм Форда — Беллмана или строит дерево кратчайших путей, или находит цикл отрицательной стоимости.

Доказательство.

- Корректность алгоритма вытекает из ранее доказанных лемм.
- Так как сложность одной итерации алгоритма Форда — Беллмана не превосходит $O(m)$,
- а количество итераций не больше n,
- то общая сложность алгоритма есть $O(nm)$.

□

Сложность алгоритма Форда — Беллмана

Теорема 7

За время $O(nm)$ алгоритм Форда — Беллмана или строит дерево кратчайших путей, или находит цикл отрицательной стоимости.

Доказательство.

- Корректность алгоритма вытекает из ранее доказанных лемм.
- Так как сложность одной итерации алгоритма Форда — Беллмана не превосходит $O(m)$,
- а количество итераций не больше n,
- то общая сложность алгоритма есть $O(nm)$.

□

Сложность алгоритма Форда — Беллмана

Теорема 7

За время $O(nm)$ алгоритм Форда — Беллмана или строит дерево кратчайших путей, или находит цикл отрицательной стоимости.

Доказательство.

- Корректность алгоритма вытекает из ранее доказанных лемм.
- Так как сложность одной итерации алгоритма Форда — Беллмана не превосходит $O(m)$,
- а количество итераций не больше n,
- то общая сложность алгоритма есть $O(nm)$.

□

Сложность алгоритма Форда — Беллмана

Теорема 7

За время $O(nm)$ алгоритм Форда — Беллмана или строит дерево кратчайшийх путей, или находит цикл отрицательной стоимости.

Доказательство.

- Корректность алгоритма вытекает из ранее доказанных лемм.
- Так как сложность одной итерации алгоритма Форда — Беллмана не превосходит $O(m)$,
- а количество итераций не больше n,
- то общая сложность алгоритма есть $O(nm)$.

□

Сложность алгоритма Форда — Беллмана

Теорема 7

За время $O(nm)$ алгоритм Форда — Беллмана или строит дерево кратчайших путей, или находит цикл отрицательной стоимости.

Доказательство.

- Корректность алгоритма вытекает из ранее доказанных лемм.
- Так как сложность одной итерации алгоритма Форда — Беллмана не превосходит $O(m)$,
- а количество итераций не больше n,
- то общая сложность алгоритма есть $O(nm)$.

□

Сложность алгоритма Форда — Беллмана

Теорема 7

За время $O(nm)$ алгоритм Форда — Беллмана или строит дерево кратчайших путей, или находит цикл отрицательной стоимости.

Доказательство.

- Корректность алгоритма вытекает из ранее доказанных лемм.
- Так как сложность одной итерации алгоритма Форда — Беллмана не превосходит $O(m)$,
- а количество итераций не больше n,
- то общая сложность алгоритма есть $O(nm)$.

$\square$

Сложность алгоритма Форда — Беллмана

Теорема 7

За время $O(nm)$ алгоритм Форда — Беллмана или строит дерево кратчайших путей, или находит цикл отрицательной стоимости.

Доказательство.

- Корректность алгоритма вытекает из ранее доказанных лемм.
- Так как сложность одной итерации алгоритма Форда — Беллмана не превосходит $O(m)$,
- а количество итераций не больше n,
- то общая сложность алгоритма есть $O(nm)$.

$\square$

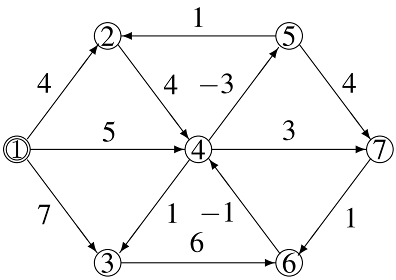
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{\quad\}$
3. $S = \{\quad\}$
4. $S = \{\quad\}$
5. $S = \{\}$

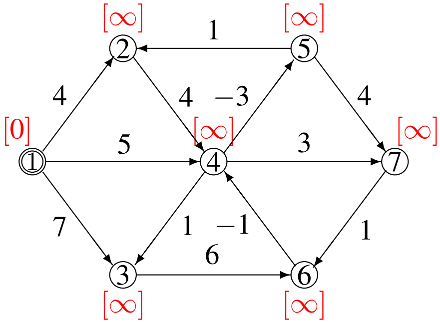


Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.



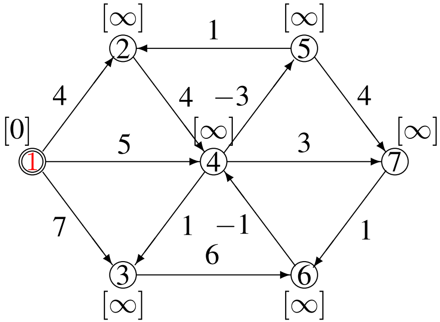
1. Инициализация: $S = \{1\}$.
2. $S = \{ \quad \}$
3. $S = \{ \quad \}$
4. $S = \{ \quad \}$
5. $S = \{\}$

Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.



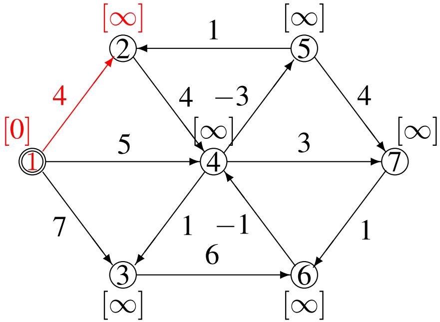
1. Инициализация: $S = \{1\}$.
2. $S = \{\quad\}$
3. $S = \{\quad\}$
4. $S = \{\quad\}$
5. $S = \{\}$

Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.



1. Инициализация: $S = \{1\}$.
2. $S = \{\quad\}$
3. $S = \{\quad\}$
4. $S = \{\quad\}$
5. $S = \{\}$

$d(1) + c(1,2) = 0 + 4 = 4 < \infty = d(2) \;\Rightarrow\; d(2) = 4.$

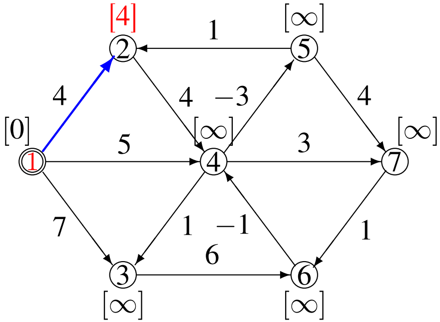
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2 \quad \}$
3. $S = \{ \quad \}$
4. $S = \{ \quad \}$
5. $S = \{\}$



$d(1) + c(1,2) = 0 + 4 = 4 < \infty = d(2) \;\Rightarrow\; d(2) = 4.$

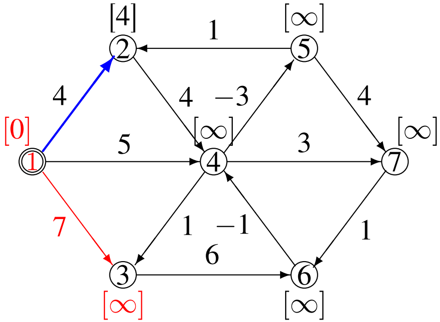
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2 \quad \}$
3. $S = \{ \quad \}$
4. $S = \{ \quad \}$
5. $S = \{\}$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(1) + c(1,3) = 0 + 7 = 7 < \infty = d(3) \Rightarrow d(3) = 7.$

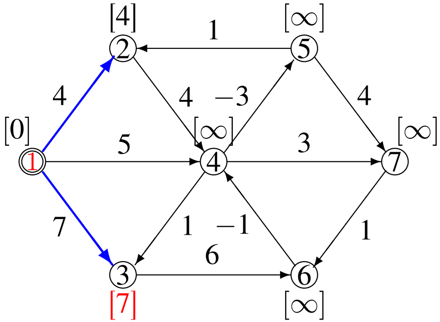
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3 \quad \}$
3. $S = \{ \qquad \}$
4. $S = \{ \qquad \}$
5. $S = \{\}$



$d(1) + c(1,3) = 0 + 7 = 7 < \infty = d(3) \Rightarrow d(3) = 7.$

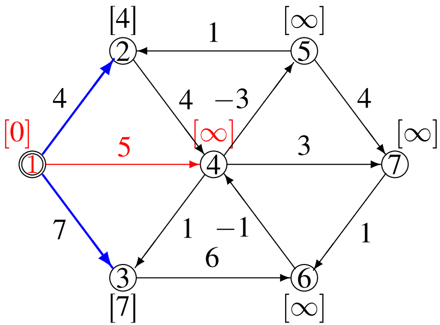
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3\}$
3. $S = \{\}$
4. $S = \{\}$
5. $S = \{\}$



$d(1) + c(1,4) = 0 + 5 = 5 < \infty = d(4) \Rightarrow d(4) = 5.$

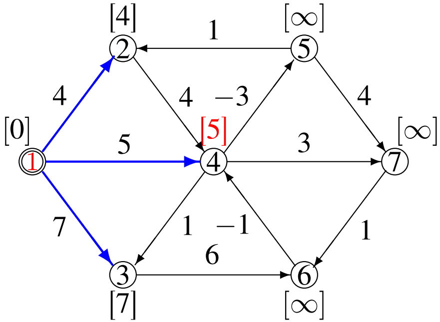
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{ \quad \}$
4. $S = \{ \quad \}$
5. $S = \{\}$



$d(1) + c(1,4) = 0 + 5 = 5 < \infty = d(4) \Rightarrow d(4) = 5.$

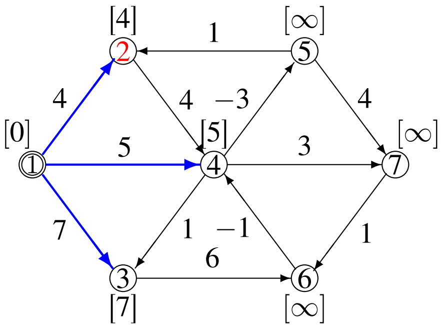
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{ \quad \}$
4. $S = \{ \quad \}$
5. $S = \{\}$



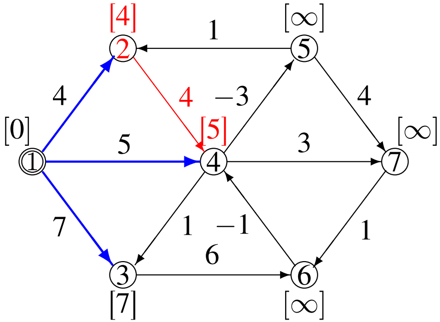
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{ \qquad \}$
4. $S = \{ \qquad \}$
5. $S = \{\}$



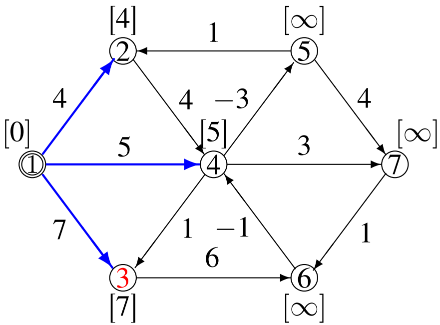
$d(2) + c(2, 4) = 4 + 4 = 8 > 5 = d(4)$.

Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.



1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{ \quad \}$
4. $S = \{ \quad \}$
5. $S = \{\}$

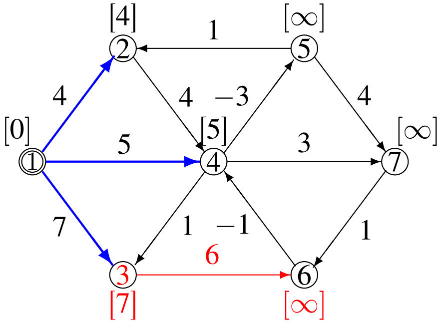
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{\qquad\qquad\}$
4. $S = \{\qquad\quad\}$
5. $S = \{\}$



$d(3) + c(3,6) = 7 + 6 = 13 < \infty = d(6) \;\Rightarrow\; d(6) = 13.$

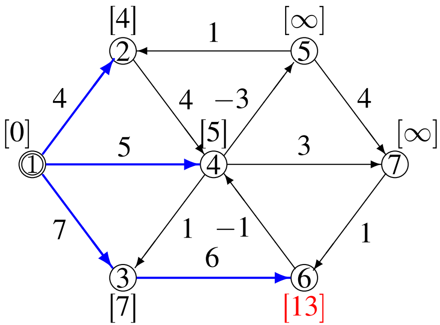
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6 \quad\quad \}$
4. $S = \{ \quad\quad \}$
5. $S = \{\}$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(3) + c(3,6) = 7 + 6 = 13 < \infty = d(6) \Rightarrow d(6) = 13.$

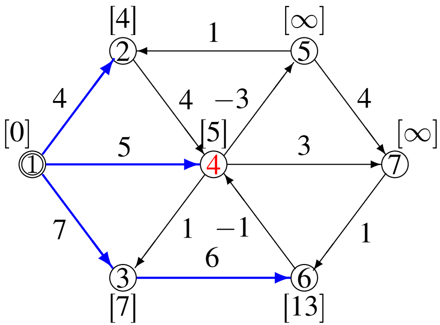
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6 \quad\quad \}$
4. $S = \{ \quad\quad \}$
5. $S = \{\}$

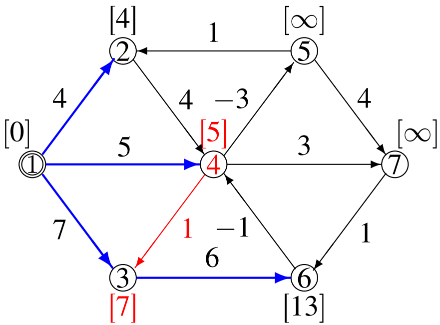


Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.



1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6 \quad \}$
4. $S = \{ \quad \}$
5. $S = \{\}$

$d(4) + c(4,3) = 5 + 1 = 6 < 7 = d(3) \Rightarrow d(3) = 6.$

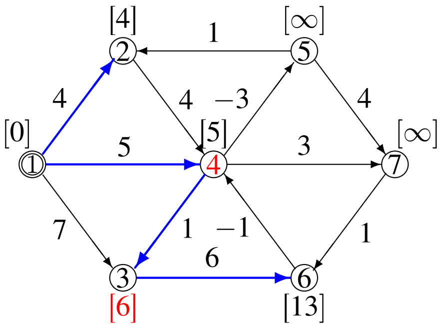
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3 \quad \}$
4. $S = \{ \quad \}$
5. $S = \{\}$



$d(4) + c(4, 3) = 5 + 1 = 6 < 7 = d(3) \Rightarrow d(3) = 6.$

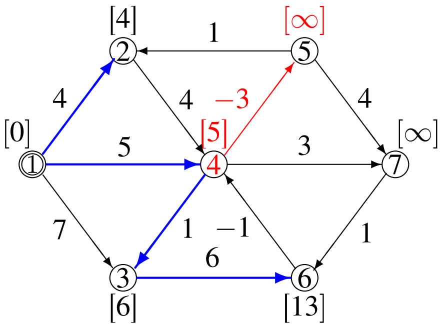
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3 \quad\quad \}$
4. $S = \{ \quad\quad \}$
5. $S = \{\}$



$d(4) + c(4, 5) = 5 + (-3) = 2 < \infty = d(5) \;\Rightarrow\; d(5) = 2.$

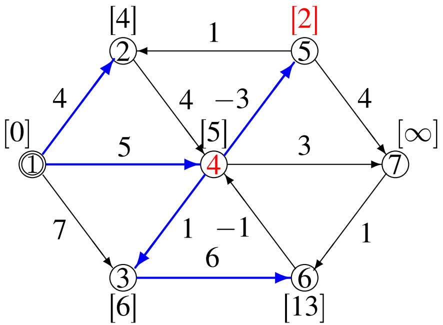
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5\}$
4. $S = \{\}$
5. $S = \{\}$



$d(4) + c(4,5) = 5 + (-3) = 2 < \infty = d(5) \Rightarrow d(5) = 2.$

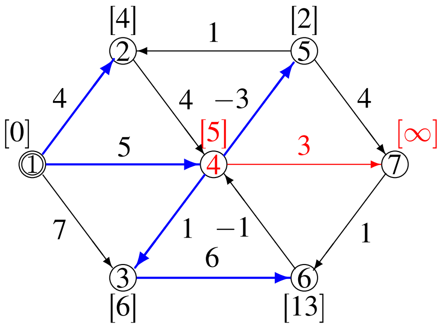
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5 \quad \}$
4. $S = \{ \quad \}$
5. $S = \{\}$

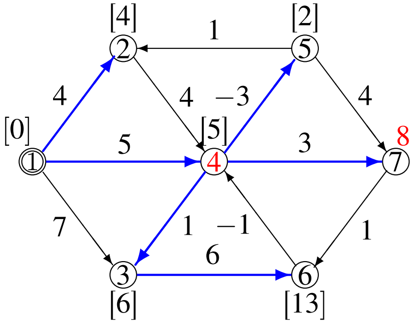
Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(4) + c(4, 7) = 5 + 3 = 8 < \infty = d(7) \Rightarrow d(7) = 8.$

Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



Обозначения:

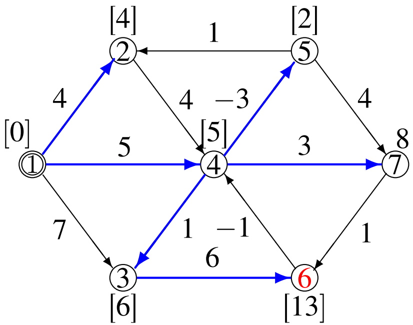
- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{\quad\}$
5. $S = \{\}$

$d(4) + c(4, 7) = 5 + 3 = 8 < \infty = d(7) \Rightarrow d(7) = 8.$

Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{\quad\}$
5. $S = \{\}$

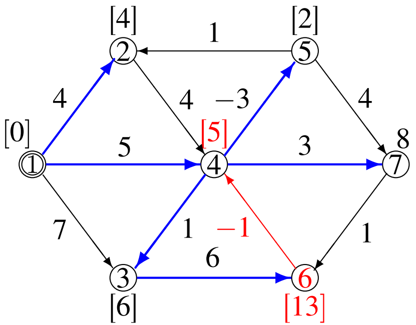
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{\quad\quad\}$
5. $S = \{\}$



$d(6) + c(6, 4) = 13 + (-1) = 12 > 5 = d(4)$.

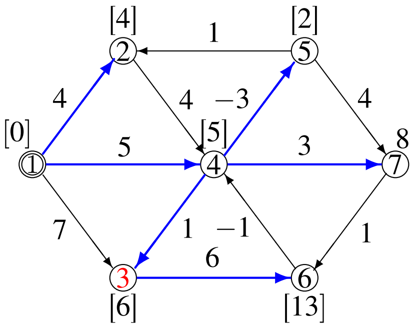
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{\quad\quad\}$
5. $S = \{\}$



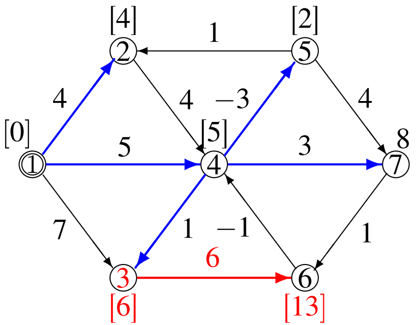
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{ \quad \}$
5. $S = \{\}$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(3) + c(3,6) = 6 + 6 = 12 < 13 = d(6) \;\Rightarrow\; d(6) = 12.$

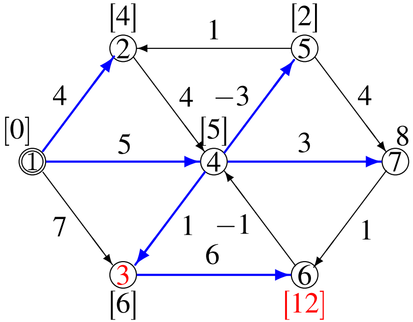
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6 \quad \}$
5. $S = \{\}$

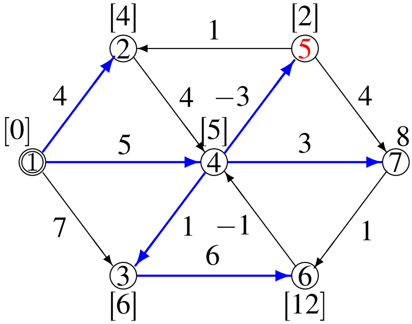
Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(3) + c(3, 6) = 6 + 6 = 12 < 13 = d(6) \Rightarrow d(6) = 12.$

Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6 \quad \}$
5. $S = \{\}$

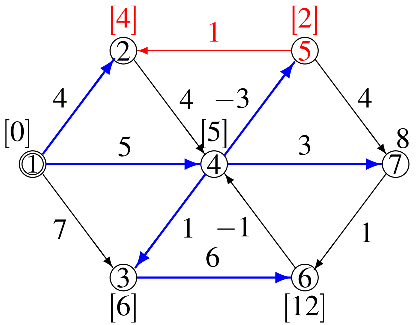
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6 \quad \}$
5. $S = \{\}$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(5) + c(5, 2) = 2 + 1 = 3 < 4 = d(2) \Rightarrow d(2) = 3.$

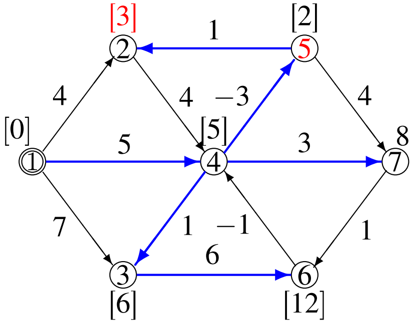
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2 \quad\}$
5. $S = \{\}$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(5) + c(5,2) = 2 + 1 = 3 < 4 = d(2) \Rightarrow d(2) = 3$.

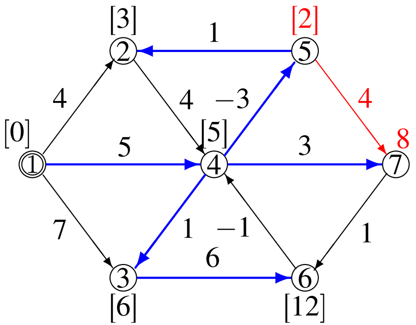
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2 \quad \}$
5. $S = \{\}$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(5) + c(5,7) = 2 + 4 = 6 < 8 = d(7) \Rightarrow d(7) = 6.$

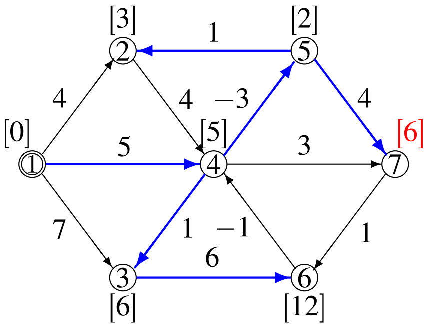
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2, 7\}$
5. $S = \{\}$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



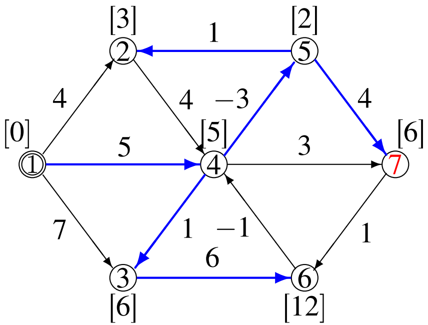
$d(5) + c(5,7) = 2 + 4 = 6 < 8 = d(7) \Rightarrow d(7) = 6.$

Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.



1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2, 7\}$
5. $S = \{\}$

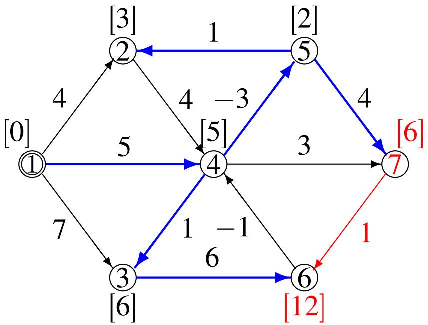
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2, 7\}$
5. $S = \{\}$



$d(7) + c(7,6) = 6 + 1 = 7 < 12 = d(6) \Rightarrow d(6) = 7.$

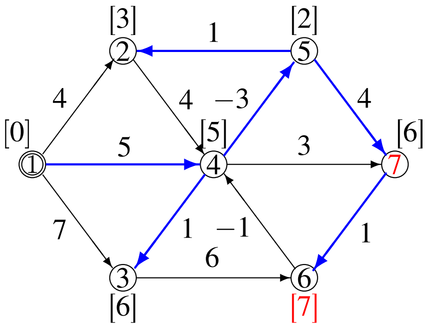
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2, 7\}$
5. $S = \{\}$



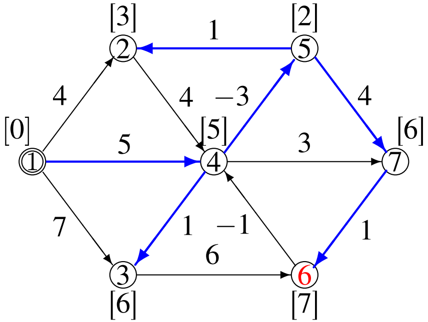
$d(7) + c(7,6) = 6 + 1 = 7 < 12 = d(6)$ $\Rightarrow$ $d(6) = 7$.

Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.



1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2, 7\}$
5. $S = \{\}$

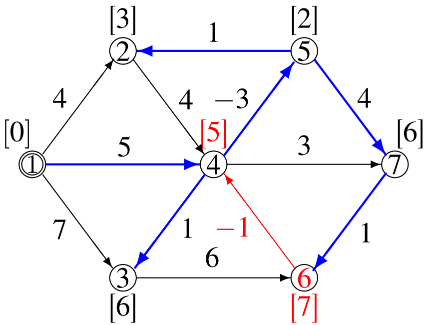
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2, 7\}$
5. $S = \{\}$



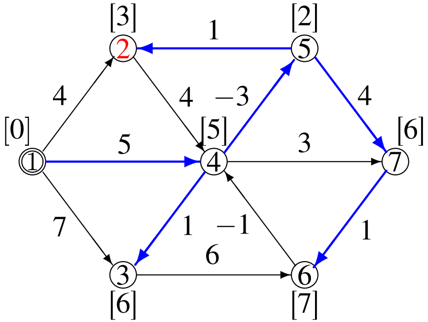
$d(6) + c(6,4) = 7 + (-1) = 6 > 5 = d(4)$.

Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.



1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2, 7\}$
5. $S = \{\}$

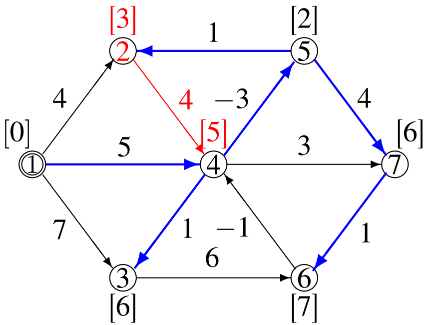
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2, 7\}$
5. $S = \{\}$



$d(2) + c(2, 4) = 3 + 4 = 7 > 5 = d(4)$.

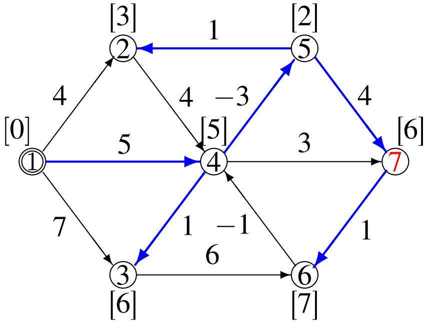
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2, 7\}$
5. $S = \{\}$



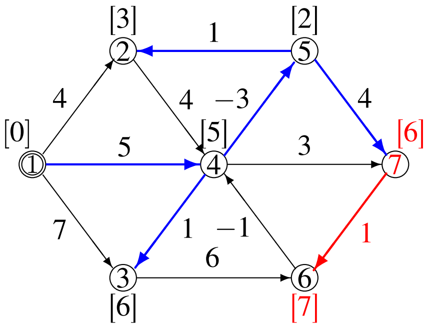
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2, 7\}$
5. $S = \{\}$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(7) + c(7, 6) = 6 + 1 = 7 = d(6)$.

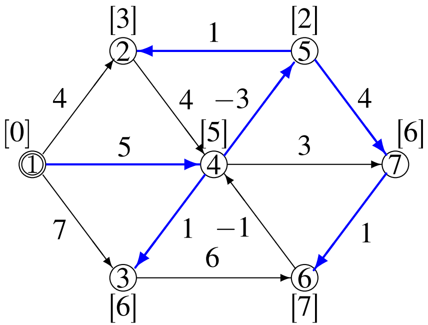
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. Инициализация: $S = \{1\}$.
2. $S = \{2, 3, 4\}$
3. $S = \{6, 3, 5, 7\}$
4. $S = \{6, 2, 7\}$
5. $S = \{\}$



Поиск отрицательных циклов

- Задача поиска в орграфе $G = (V, E)$, дугам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$, цикла Γ отрицательной стоимости $c(\Gamma) < 0$,
- имеет самостоятельный интерес.
- Алгоритм Форда — Беллмана может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
  - добавим к графу G новую вершину s
  - и соединим ее дугой (s, v) нулевой стоимости с каждой вершиной $v \in V$.
- В расширенном графе G'
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G' также является циклом в G.
- Применим алгоритм Форда — Беллмана к графу G' с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Поиск отрицательных циклов

- Задача поиска в орграфе $G = (V, E)$, дугам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$, цикла Γ отрицательной стоимости $c(\Gamma) < 0$,
- имеет самостоятельный интерес.
- Алгоритм Форда — Беллмана может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
  - добавим к графу G новую вершину s
  - и соединим ее дугой (s, v) нулевой стоимости с каждой вершиной $v \in V$.
- В расширенном графе G'
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G' также является циклом в G.
- Применим алгоритм Форда — Беллмана к графу G' с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Поиск отрицательных циклов

- Задача поиска в орграфе $G = (V, E)$, дугам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$, цикла Γ отрицательной стоимости $c(\Gamma) < 0$,
- имеет самостоятельный интерес.
- Алгоритм Форда — Беллмана может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
  - добавим к графу G новую вершину s
  - и соединим ее дугой (s, v) нулевой стоимости с каждой вершиной $v \in V$.
- В расширенном графе G'
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G' также является циклом в G.
- Применим алгоритм Форда — Беллмана к графу G' с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Поиск отрицательных циклов

- Задача поиска в орграфе $G = (V, E)$, дугам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$, цикла Γ отрицательной стоимости $c(\Gamma) < 0$,
- имеет самостоятельный интерес.
- Алгоритм Форда — Беллмана может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
  - добавим к графу G новую вершину s
  - и соединим ее дугой (s, v) нулевой стоимости с каждой вершиной $v \in V$.
- В расширенном графе G'
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G' также является циклом в G.
- Применим алгоритм Форда — Беллмана к графу G' с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Поиск отрицательных циклов

- Задача поиска в орграфе $G = (V, E)$, дугам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$, цикла Γ отрицательной стоимости $c(\Gamma) < 0$,
- имеет самостоятельный интерес.
- Алгоритм Форда — Беллмана может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
  - добавим к графу G новую вершину s
  - и соединим ее дугой (s, v) нулевой стоимости с каждой вершиной $v \in V$.
- В расширенном графе G'
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G' также является циклом в G.
- Применим алгоритм Форда — Беллмана к графу G' с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Поиск отрицательных циклов

- Задача поиска в орграфе $G = (V, E)$, дугам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$, цикла Γ отрицательной стоимости $c(\Gamma) < 0$,
- имеет самостоятельный интерес.
- Алгоритм Форда — Беллмана может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
  - добавим к графу G новую вершину s
  - и соединим ее дугой (s, v) нулевой стоимости с каждой вершиной $v \in V$.
- В расширенном графе G'
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G' также является циклом в G.
- Применим алгоритм Форда — Беллмана к графу G' с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Поиск отрицательных циклов

- Задача поиска в орграфе $G = (V, E)$, дугам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$, цикла Γ отрицательной стоимости $c(\Gamma) < 0$,
- имеет самостоятельный интерес.
- Алгоритм Форда — Беллмана может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
  - добавим к графу G новую вершину s
  - и соединим ее дугой (s, v) нулевой стоимости с каждой вершиной $v \in V$.
- В расширенном графе G'
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G' также является циклом в G.
- Применим алгоритм Форда — Беллмана к графу G' с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Поиск отрицательных циклов

- Задача поиска в орграфе $G = (V, E)$, дугам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$, цикла Γ отрицательной стоимости $c(\Gamma) < 0$,
- имеет самостоятельный интерес.
- Алгоритм Форда — Беллмана может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
  - добавим к графу G новую вершину s
  - и соединим ее дугой (s, v) нулевой стоимости с каждой вершиной $v \in V$.
- В расширенном графе G'
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G' также является циклом в G.
- Применим алгоритм Форда — Беллмана к графу G' с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Поиск отрицательных циклов

- Задача поиска в орграфе $G = (V, E)$, дугам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$, цикла Γ отрицательной стоимости $c(\Gamma) < 0$,
- имеет самостоятельный интерес.
- Алгоритм Форда — Беллмана может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
  - добавим к графу G новую вершину s
  - и соединим ее дугой (s, v) нулевой стоимости с каждой вершиной $v \in V$.
- В расширенном графе G'
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G' также является циклом в G.
- Применим алгоритм Форда — Беллмана к графу G' с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Поиск отрицательных циклов

- Задача поиска в орграфе $G = (V, E)$, дугам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$, цикла Γ отрицательной стоимости $c(\Gamma) < 0$,
- имеет самостоятельный интерес.
- Алгоритм Форда — Беллмана может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
  - добавим к графу G новую вершину s
  - и соединим ее дугой (s, v) нулевой стоимости с каждой вершиной $v \in V$.
- В расширенном графе G'
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G' также является циклом в G.
- Применим алгоритм Форда — Беллмана к графу G' с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Поиск отрицательных циклов

- Задача поиска в орграфе $G = (V, E)$, дугам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$, цикла Γ отрицательной стоимости $c(\Gamma) < 0$,
- имеет самостоятельный интерес.
- Алгоритм Форда — Беллмана может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
    - добавим к графу G новую вершину s
    - и соединим ее дугой (s, v) нулевой стоимости с каждой вершиной $v \in V$.
- В расширенном графе G'
    1. каждая вершина достижима из вершины s
    2. и любой цикл в G' также является циклом в G.
- Применим алгоритм Форда — Беллмана к графу G' с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Поиск отрицательных циклов

- Задача поиска в орграфе $G = (V, E)$, дугам $(v, w) \in E$ которого приписаны стоимости $c(v, w)$, цикла Γ отрицательной стоимости $c(\Gamma) < 0$,
- имеет самостоятельный интерес.
- Алгоритм Форда — Беллмана может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
  - добавим к графу G новую вершину s
  - и соединим ее дугой (s, v) нулевой стоимости с каждой вершиной $v \in V$.
- В расширенном графе G'
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G' также является циклом в G.
- Применим алгоритм Форда — Беллмана к графу G' с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Арбитраж на валютном рынке

- Все валюты, представленные на некотором валютном рынке, соответствуют вершинам $v \in V$ графа $G = (V, E)$.
- Дуга $(v, w) \in E$ представляет транзакцию валюты v в валюту w: ед. валюты v меняют на $\gamma(v, w)$ ед. валюты w.
- Цикл $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в графе G называется *арбитражем на валютном рынке*, если
- произведение обменных курсов на дугах этого цикла больше единицы: $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.
- Задача поиска арбитража на валютном рынке сводится к задаче поиска цикла отр. стоимости в графе G,
- если дугам приписать стоимости $c(v, w) = -\ln(\gamma(v, w))$.
- Стоимость цикла $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в G равна

$$c(\Gamma) = -\sum_{i=1}^{k} \ln(\gamma(v_{i-1}, v_i)) = -\ln\left(\prod_{i=1}^{k} \gamma(v_{i-1}, v_i)\right).$$

- Поэтому из $c(\Gamma) < 0$ следует, что $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.

Арбитраж на валютном рынке

- Все валюты, представленные на некотором валютном рынке, соответствуют вершинам $v \in V$ графа $G = (V, E)$.
- Дуга $(v, w) \in E$ представляет транзакцию валюты v в валюту w: ед. валюты v меняют на $\gamma(v, w)$ ед. валюты w.
- Цикл $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в графе G называется *арбитражем на валютном рынке*, если
- произведение обменных курсов на дугах этого цикла больше единицы: $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.
- Задача поиска арбитража на валютном рынке сводится к задаче поиска цикла отр. стоимости в графе G,
- если дугам приписать стоимости $c(v, w) = -\ln(\gamma(v, w))$.
- Стоимость цикла $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в G равна

$$c(\Gamma) = -\sum_{i=1}^{k} \ln(\gamma(v_{i-1}, v_i)) = -\ln\left(\prod_{i=1}^{k} \gamma(v_{i-1}, v_i)\right).$$

- Поэтому из $c(\Gamma) < 0$ следует, что $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.

Арбитраж на валютном рынке

- Все валюты, представленные на некотором валютном рынке, соответствуют вершинам $v \in V$ графа $G = (V, E)$.
- Дуга $(v, w) \in E$ представляет транзакцию валюты v в валюту w: ед. валюты v меняют на $\gamma(v, w)$ ед. валюты w.
- Цикл $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в графе G называется *арбитражем на валютном рынке*, если
- произведение обменных курсов на дугах этого цикла больше единицы: $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.
- Задача поиска арбитража на валютном рынке сводится к задаче поиска цикла отр. стоимости в графе G,
- если дугам приписать стоимости $c(v, w) = -\ln(\gamma(v, w))$.
- Стоимость цикла $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в G равна

$$c(\Gamma) = -\sum_{i=1}^{k} \ln(\gamma(v_{i-1}, v_i)) = -\ln\left(\prod_{i=1}^{k} \gamma(v_{i-1}, v_i)\right).$$

- Поэтому из $c(\Gamma) < 0$ следует, что $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.

Арбитраж на валютном рынке

- Все валюты, представленные на некотором валютном рынке, соответствуют вершинам $v \in V$ графа $G = (V, E)$.
- Дуга $(v, w) \in E$ представляет транзакцию валюты v в валюту w: ед. валюты v меняют на $\gamma(v, w)$ ед. валюты w.
- Цикл $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в графе G называется *арбитражем на валютном рынке*, если
- произведение обменных курсов на дугах этого цикла больше единицы: $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.
- Задача поиска арбитража на валютном рынке сводится к задаче поиска цикла отр. стоимости в графе G,
- если дугам приписать стоимости $c(v, w) = -\ln(\gamma(v, w))$.
- Стоимость цикла $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в G равна

$$c(\Gamma) = -\sum_{i=1}^{k} \ln(\gamma(v_{i-1}, v_i)) = -\ln\left(\prod_{i=1}^{k} \gamma(v_{i-1}, v_i)\right).$$

- Поэтому из $c(\Gamma) < 0$ следует, что $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.

Арбитраж на валютном рынке

- Все валюты, представленные на некотором валютном рынке, соответствуют вершинам $v \in V$ графа $G = (V, E)$.
- Дуга $(v, w) \in E$ представляет транзакцию валюты v в валюту w: ед. валюты v меняют на $\gamma(v, w)$ ед. валюты w.
- Цикл $\Gamma = (v_0, v_1, \dots, v_k = v_0)$ в графе G называется *арбитражем на валютном рынке*, если
- произведение обменных курсов на дугах этого цикла больше единицы: $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.
- Задача поиска арбитража на валютном рынке сводится к задаче поиска цикла отр. стоимости в графе G,
- если дугам приписать стоимости $c(v, w) = -\ln(\gamma(v, w))$.
- Стоимость цикла $\Gamma = (v_0, v_1, \dots, v_k = v_0)$ в G равна

$$c(\Gamma) = -\sum_{i=1}^{k} \ln(\gamma(v_{i-1}, v_i)) = -\ln\left(\prod_{i=1}^{k} \gamma(v_{i-1}, v_i)\right).$$

- Поэтому из $c(\Gamma) < 0$ следует, что $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.

Арбитраж на валютном рынке

- Все валюты, представленные на некотором валютном рынке, соответствуют вершинам $v \in V$ графа $G = (V, E)$.
- Дуга $(v, w) \in E$ представляет транзакцию валюты v в валюту w: ед. валюты v меняют на $\gamma(v, w)$ ед. валюты w.
- Цикл $\Gamma = (v_0, v_1, \dots, v_k = v_0)$ в графе G называется *арбитражем на валютном рынке*, если
- произведение обменных курсов на дугах этого цикла больше единицы: $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.
- Задача поиска арбитража на валютном рынке сводится к задаче поиска цикла отр. стоимости в графе G,
- если дугам приписать стоимости $c(v, w) = -\ln(\gamma(v, w))$.
- Стоимость цикла $\Gamma = (v_0, v_1, \dots, v_k = v_0)$ в G равна

$$c(\Gamma) = -\sum_{i=1}^{k} \ln(\gamma(v_{i-1}, v_i)) = -\ln\left(\prod_{i=1}^{k} \gamma(v_{i-1}, v_i)\right).$$

- Поэтому из $c(\Gamma) < 0$ следует, что $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.

Арбитраж на валютном рынке

- Все валюты, представленные на некотором валютном рынке, соответствуют вершинам $v \in V$ графа $G = (V, E)$.
- Дуга $(v, w) \in E$ представляет транзакцию валюты v в валюту w: ед. валюты v меняют на $\gamma(v, w)$ ед. валюты w.
- Цикл $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в графе G называется *арбитражем на валютном рынке*, если
- произведение обменных курсов на дугах этого цикла больше единицы: $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.
- Задача поиска арбитража на валютном рынке сводится к задаче поиска цикла отр. стоимости в графе G,
- если дугам приписать стоимости $c(v, w) = -\ln(\gamma(v, w))$.
- Стоимость цикла $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в G равна
$$c(\Gamma) = -\sum_{i=1}^{k} \ln(\gamma(v_{i-1}, v_i)) = -\ln\left(\prod_{i=1}^{k} \gamma(v_{i-1}, v_i)\right).$$
- Поэтому из $c(\Gamma) < 0$ следует, что $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.

Арбитраж на валютном рынке

- Все валюты, представленные на некотором валютном рынке, соответствуют вершинам $v \in V$ графа $G = (V, E)$.
- Дуга $(v, w) \in E$ представляет транзакцию валюты v в валюту w: ед. валюты v меняют на $\gamma(v, w)$ ед. валюты w.
- Цикл $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в графе G называется *арбитражем на валютном рынке*, если
- произведение обменных курсов на дугах этого цикла больше единицы: $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.
- Задача поиска арбитража на валютном рынке сводится к задаче поиска цикла отр. стоимости в графе G,
- если дугам приписать стоимости $c(v, w) = -\ln(\gamma(v, w))$.
- Стоимость цикла $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в G равна

$$c(\Gamma) = -\sum_{i=1}^{k} \ln(\gamma(v_{i-1}, v_i)) = -\ln\left(\prod_{i=1}^{k} \gamma(v_{i-1}, v_i)\right).$$

- Поэтому из $c(\Gamma) < 0$ следует, что $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.

Арбитраж на валютном рынке

- Все валюты, представленные на некотором валютном рынке, соответствуют вершинам $v \in V$ графа $G = (V, E)$.
- Дуга $(v, w) \in E$ представляет транзакцию валюты v в валюту w: ед. валюты v меняют на $\gamma(v, w)$ ед. валюты w.
- Цикл $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в графе G называется *арбитражем на валютном рынке*, если
- произведение обменных курсов на дугах этого цикла больше единицы: $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.
- Задача поиска арбитража на валютном рынке сводится к задаче поиска цикла отр. стоимости в графе G,
- если дугам приписать стоимости $c(v, w) = -\ln(\gamma(v, w))$.
- Стоимость цикла $\Gamma = (v_0, v_1, \ldots, v_k = v_0)$ в G равна

$$c(\Gamma) = -\sum_{i=1}^{k} \ln(\gamma(v_{i-1}, v_i)) = -\ln\left(\prod_{i=1}^{k} \gamma(v_{i-1}, v_i)\right).$$

- Поэтому из $c(\Gamma) < 0$ следует, что $\prod_{i=1}^{k} \gamma(v_{i-1}, v_i) > 1$.

План лекции

1. Дерево кратчайших путей
   - Принцип оптимальности
   - Критерий существования дерева кратчайших путей
2. Алгоритмы поиска кратчайших путей
   - Алгоритм Форда — Беллмана
   - Алгоритм Дейкстры
   - Кратчайшие пути в ациклических графах

Неотрицательные стоимости

- Алгоритм Дейкстры применяется, когда стоимости всех дуг неотрицательные.
- Это также алгоритм последовательной аппроксимации,
- который на каждой итерации уточняет верхние оценки $d(v)$ длин кратчайших путей от источника s.
- Алгоритм поддерживает подмножество вершин S, до которых кратчайший путь уже найден.
- Если $v \in S$, то $d(v)$ есть длина кратч. пути от s до v.
- На очередной итерации алгоритм выбирает вершину $w \in V \setminus S$ с минимальной «меткой» $d(w)$, добавляе ее к S,
- и для всех дуг $(w,v) \in E(w, V \setminus S)$ перевычисляет метки их конечных вершин по правилу:

$$d(v) = \min\{d(v), d(w) + c(w,v)\}.$$

Неотрицательные стоимости

- Алгоритм Дейкстры применяется, когда стоимости всех дуг неотрицательные.
- Это также алгоритм последовательной аппроксимации,
- который на каждой итерации уточняет верхние оценки $d(v)$ длин кратчайших путей от источника s.
- Алгоритм поддерживает подмножество вершин S, до которых кратчайший путь уже найден.
- Если $v \in S$, то $d(v)$ есть длина кратч. пути от s до v.
- На очередной итерации алгоритм выбирает вершину $w \in V \setminus S$ с минимальной «меткой» $d(w)$, добавляе ее к S,
- и для всех дуг $(w, v) \in E(w, V \setminus S)$ перевычисляет метки их конечных вершин по правилу:

$$d(v) = \min\{d(v), d(w) + c(w, v)\}.$$

Неотрицательные стоимости

- Алгоритм Дейкстры применяется, когда стоимости всех дуг неотрицательные.
- Это также алгоритм последовательной аппроксимации,
- который на каждой итерации уточняет верхние оценки $d(v)$ длин кратчайших путей от источника s.
- Алгоритм поддерживает подмножество вершин S, до которых кратчайший путь уже найден.
- Если $v \in S$, то $d(v)$ есть длина кратч. пути от s до v.
- На очередной итерации алгоритм выбирает вершину $w \in V \setminus S$ с минимальной «меткой» $d(w)$, добавляе ее к S,
- и для всех дуг $(w, v) \in E(w, V \setminus S)$ перевычисляет метки их конечных вершин по правилу:

$$d(v) = \min\{d(v), d(w) + c(w, v)\}.$$

Неотрицательные стоимости

- Алгоритм Дейкстры применяется, когда стоимости всех дуг неотрицательные.
- Это также алгоритм последовательной аппроксимации,
- который на каждой итерации уточняет верхние оценки $d(v)$ длин кратчайших путей от источника s.
- Алгоритм поддерживает подмножество вершин S, до которых кратчайший путь уже найден.
- Если $v \in S$, то $d(v)$ есть длина кратч. пути от s до v.
- На очередной итерации алгоритм выбирает вершину $w \in V \setminus S$ с минимальной «меткой» $d(w)$, добавляе ее к S,
- и для всех дуг $(w, v) \in E(w, V \setminus S)$ перевычисляет метки их конечных вершин по правилу:

$$d(v) = \min\{d(v), d(w) + c(w, v)\}.$$

Неотрицательные стоимости

- Алгоритм Дейкстры применяется, когда стоимости всех дуг неотрицательные.
- Это также алгоритм последовательной аппроксимации,
- который на каждой итерации уточняет верхние оценки $d(v)$ длин кратчайших путей от источника s.
- Алгоритм поддерживает подмножество вершин S, до которых кратчайший путь уже найден.
- Если $v \in S$, то $d(v)$ есть длина кратч. пути от s до v.
- На очередной итерации алгоритм выбирает вершину $w \in V \setminus S$ с минимальной «меткой» $d(w)$, добавляе ее к S,
- и для всех дуг $(w, v) \in E(w, V \setminus S)$ перевычисляет метки их конечных вершин по правилу:

$$d(v) = \min\{d(v), d(w) + c(w, v)\}.$$

Неотрицательные стоимости

- Алгоритм Дейкстры применяется, когда стоимости всех дуг неотрицательные.
- Это также алгоритм последовательной аппроксимации,
- который на каждой итерации уточняет верхние оценки $d(v)$ длин кратчайших путей от источника s.
- Алгоритм поддерживает подмножество вершин S, до которых кратчайший путь уже найден.
- Если $v \in S$, то $d(v)$ есть длина кратч. пути от s до v.
- На очередной итерации алгоритм выбирает вершину $w \in V \setminus S$ с минимальной «меткой» $d(w)$, добавляе ее к S,
- и для всех дуг $(w, v) \in E(w, V \setminus S)$ перевычисляет метки их конечных вершин по правилу:

$$d(v) = \min\{d(v), d(w) + c(w, v)\}.$$

Неотрицательные стоимости

- Алгоритм Дейкстры применяется, когда стоимости всех дуг неотрицательные.
- Это также алгоритм последовательной аппроксимации,
- который на каждой итерации уточняет верхние оценки $d(v)$ длин кратчайших путей от источника s.
- Алгоритм поддерживает подмножество вершин S, до которых кратчайший путь уже найден.
- Если $v \in S$, то $d(v)$ есть длина кратч. пути от s до v.
- На очередной итерации алгоритм выбирает вершину $w \in V \setminus S$ с минимальной «меткой» $d(w)$, добавляе ее к S,
- и для всех дуг $(w, v) \in E(w, V \setminus S)$ перевычисляет метки их конечных вершин по правилу:

$$d(v) = \min\{d(v), d(w) + c(w, v)\}.$$

Описание алгоритма Дейкстры

- **Вход**: Орграф $G = (V, E)$, функция $c : E \to \mathbb{R}$, вершина $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\textbf{nil}\}$, задающие дерево кратчайших путей, функция $d : V \to \mathbb{R} \cup \{\infty\}$, где $d(v)$ — *кратчайшее расстояние* от s до v.
- 1. Для всех $v \in V \setminus \{s\}$
  положить $d(v) = \infty$ и $parent(v) = \textbf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \emptyset$.
- 3. Пока $|S| < n - 1$ выполнять шаги 3.1 и 3.2:
  - 3.1. Выбрать $w \in \arg\min\{d(v) : v \in V \setminus S\}$
    и положить $S := S \cup \{w\}$.
  - 3.2. Для всех $(w, v) \in E(w, V \setminus S)$, что $d(v) > d(w) + c(w, v)$,
    положить $d(v) = d(w) + c(w, v)$ и $parent(v) = w$.

Описание алгоритма Дейкстры

- **Вход**: Орграф $G = (V, E)$, функция $c : E \to \mathbb{R}$, вершина $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, задающие дерево кратчайших путей, функция $d : V \to \mathbb{R} \cup \{\infty\}$, где $d(v)$ — *кратчайшее расстояние* от s до v.
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) = \mathbf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \emptyset$.
- 3. Пока $|S| < n - 1$ выполнять шаги 3.1 и 3.2:
  - 3.1. Выбрать $w \in \arg\min\{d(v) : v \in V \setminus S\}$ и положить $S := S \cup \{w\}$.
  - 3.2. Для всех $(w, v) \in E(w, V \setminus S)$, что $d(v) > d(w) + c(w, v)$, положить $d(v) = d(w) + c(w, v)$ и $parent(v) = w$.

Описание алгоритма Дейкстры

- **Вход**: Орграф $G = (V, E)$, функция $c : E \to \mathbb{R}$, вершина $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, задающие дерево кратчайших путей, функция $d : V \to \mathbb{R} \cup \{\infty\}$, где $d(v)$ — *кратчайшее расстояние* от s до v.
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) = \mathbf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \emptyset$.
- 3. Пока $|S| < n - 1$ выполнять шаги 3.1 и 3.2:
  - 3.1. Выбрать $w \in \arg\min\{d(v) : v \in V \setminus S\}$ и положить $S := S \cup \{w\}$.
  - 3.2. Для всех $(w, v) \in E(w, V \setminus S)$, что $d(v) > d(w) + c(w, v)$, положить $d(v) = d(w) + c(w, v)$ и $parent(v) = w$.

Описание алгоритма Дейкстры

- **Вход**: Орграф $G = (V, E)$, функция $c : E \to \mathbb{R}$, вершина $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, задающие дерево кратчайших путей, функция $d : V \to \mathbb{R} \cup \{\infty\}$, где $d(v)$ — *кратчайшее расстояние* от s до v.
- 1. Для всех $v \in V \setminus \{s\}$
  положить $d(v) = \infty$ и $parent(v) = \mathbf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \emptyset$.
- 3. Пока $|S| < n - 1$ выполнять шаги 3.1 и 3.2:
  - 3.1. Выбрать $w \in \arg\min\{d(v) : v \in V \setminus S\}$
    и положить $S := S \cup \{w\}$.
  - 3.2. Для всех $(w, v) \in E(w, V \setminus S)$, что $d(v) > d(w) + c(w, v)$,
    положить $d(v) = d(w) + c(w, v)$ и $parent(v) = w$.

Описание алгоритма Дейкстры

- **Вход**: Орграф $G = (V, E)$, функция $c : E \to \mathbb{R}$, вершина $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, задающие дерево кратчайших путей, функция $d : V \to \mathbb{R} \cup \{\infty\}$, где $d(v)$ — *кратчайшее расстояние* от s до v.
- 1. Для всех $v \in V \setminus \{s\}$
  положить $d(v) = \infty$ и $parent(v) = \mathbf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \emptyset$.
- 3. Пока $|S| < n - 1$ выполнять шаги 3.1 и 3.2:
  - 3.1. Выбрать $w \in \arg\min\{d(v) : v \in V \setminus S\}$
    и положить $S := S \cup \{w\}$.
  - 3.2. Для всех $(w, v) \in E(w, V \setminus S)$, что $d(v) > d(w) + c(w, v)$,
    положить $d(v) = d(w) + c(w, v)$ и $parent(v) = w$.

Описание алгоритма Дейкстры

- **Вход**: Орграф $G = (V, E)$, функция $c : E \to \mathbb{R}$, вершина $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, задающие дерево кратчайших путей, функция $d : V \to \mathbb{R} \cup \{\infty\}$, где $d(v)$ — *кратчайшее расстояние* от s до v.
- 1. Для всех $v \in V \setminus \{s\}$ положить $d(v) = \infty$ и $parent(v) = \mathbf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \emptyset$.
- 3. Пока $|S| < n - 1$ выполнять шаги 3.1 и 3.2:
  - 3.1. Выбрать $w \in \arg\min\{d(v) : v \in V \setminus S\}$ и положить $S := S \cup \{w\}$.
  - 3.2. Для всех $(w, v) \in E(w, V \setminus S)$, что $d(v) > d(w) + c(w, v)$, положить $d(v) = d(w) + c(w, v)$ и $parent(v) = w$.

Описание алгоритма Дейкстры

- **Вход**: Орграф $G = (V, E)$, функция $c : E \to \mathbb{R}$, вершина $s \in V$.
- **Выход**: указатели $parent : V \to V \cup \{\mathbf{nil}\}$, задающие дерево кратчайших путей, функция $d : V \to \mathbb{R} \cup \{\infty\}$, где $d(v)$ — *кратчайшее расстояние* от s до v.
- 1. Для всех $v \in V \setminus \{s\}$
  положить $d(v) = \infty$ и $parent(v) = \mathbf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \emptyset$.
- 3. Пока $|S| < n - 1$ выполнять шаги 3.1 и 3.2:
  - 3.1. Выбрать $w \in \arg\min\{d(v) : v \in V \setminus S\}$
    и положить $S := S \cup \{w\}$.
  - 3.2. Для всех $(w, v) \in E(w, V \setminus S)$, что $d(v) > d(w) + c(w, v)$,
    положить $d(v) = d(w) + c(w, v)$ и $parent(v) = w$.

Свойства алгоритма Дейкстры

Лемма 8

Алгоритм Дейкстры поддерживает следующие инварианты:

1. $d(v) \le d(w)$ *для всех* $v \in S$, $w \in V \setminus S$;
2. $d(v) + c(v,w) \ge d(w)$ *для всех* $(v,w) \in E(V,S)$.

Доказательство.

Индукцией по $|S|$. □

Свойства алгоритма Дейкстры

Лемма 8

Алгоритм Дейкстры поддерживает следующие инварианты:

1. $d(v) \le d(w)$ *для всех* $v \in S$, $w \in V \setminus S$;
2. $d(v) + c(v,w) \ge d(w)$ *для всех* $(v,w) \in E(V,S)$.

Доказательство.

Индукцией по $|S|$. □

Свойства алгоритма Дейкстры

Лемма 8

Алгоритм Дейкстры поддерживает следующие инварианты:

1. $d(v) \le d(w)$ *для всех* $v \in S$, $w \in V \setminus S$;
2. $d(v) + c(v,w) \ge d(w)$ *для всех* $(v,w) \in E(V,S)$.

Доказательство.

Индукцией по $|S|$. □

Свойства алгоритма Дейкстры

Лемма 8

Алгоритм Дейкстры поддерживает следующие инварианты:

1. $d(v) \le d(w)$ *для всех* $v \in S$, $w \in V \setminus S$;
2. $d(v) + c(v,w) \ge d(w)$ *для всех* $(v,w) \in E(V,S)$.

Доказательство.

Индукцией по $|S|$. □

Сложность алгоритма Дейкстры

Теорема 9

Если стоимости всех дуг неотрицательны, то за время $O(n^2)$ алгоритм Дейкстры строит дерево кратчайших путей.

Доказательство.

- Корректность алгоритма следует из предшествующей леммы.
- Оценим сложность алгоритма.
- На этапе инициализации требуется $O(n)$ операций.
- Сложность одной итерации $O(n)$.
- А так как количество всех итераций равно $n-1$, то сложность всего алгоритма — $O(n^2)$.

□

Сложность алгоритма Дейкстры

Теорема 9

Если стоимости всех дуг неотрицательны, то за время $O(n^2)$ алгоритм Дейкстры строит дерево кратчайших путей.

Доказательство.

- Корректность алгоритма следует из предшествующей леммы.
- Оценим сложность алгоритма.
- На этапе инициализации требуется $O(n)$ операций.
- Сложность одной итерации $O(n)$.
- А так как количество всех итераций равно $n-1$, то сложность всего алгоритма — $O(n^2)$.

$\square$

Сложность алгоритма Дейкстры

Теорема 9

Если стоимости всех дуг неотрицательны, то за время $O(n^2)$ алгоритм Дейкстры строит дерево кратчайших путей.

Доказательство.

- Корректность алгоритма следует из предшествующей леммы.
- Оценим сложность алгоритма.
- На этапе инициализации требуется $O(n)$ операций.
- Сложность одной итерации $O(n)$.
- А так как количество всех итераций равно $n-1$, то сложность всего алгоритма — $O(n^2)$.

□

Сложность алгоритма Дейкстры

Теорема 9

Если стоимости всех дуг неотрицательны, то за время $O(n^2)$ алгоритм Дейкстры строит дерево кратчайших путей.

Доказательство.

- Корректность алгоритма следует из предшествующей леммы.
- Оценим сложность алгоритма.
- На этапе инициализации требуется $O(n)$ операций.
- Сложность одной итерации $O(n)$.
- А так как количество всех итераций равно $n-1$, то сложность всего алгоритма — $O(n^2)$.

□

Сложность алгоритма Дейкстры

Теорема 9

Если стоимости всех дуг неотрицательны, то за время $O(n^2)$ алгоритм Дейкстры строит дерево кратчайших путей.

Доказательство.

- Корректность алгоритма следует из предшествующей леммы.
- Оценим сложность алгоритма.
- На этапе инициализации требуется $O(n)$ операций.
- Сложность одной итерации $O(n)$.
- А так как количество всех итераций равно $n-1$, то сложность всего алгоритма — $O(n^2)$.

□

Сложность алгоритма Дейкстры

Теорема 9

Если стоимости всех дуг неотрицательны, то за время $O(n^2)$ алгоритм Дейкстры строит дерево кратчайших путей.

Доказательство.

- Корректность алгоритма следует из предшествующей леммы.
- Оценим сложность алгоритма.
- На этапе инициализации требуется $O(n)$ операций.
- Сложность одной итерации $O(n)$.
- А так как количество всех итераций равно $n-1$, то сложность всего алгоритма — $O(n^2)$.

□

Сложность алгоритма Дейкстры

Теорема 9

Если стоимости всех дуг неотрицательны, то за время $O(n^2)$ алгоритм Дейкстры строит дерево кратчайших путей.

Доказательство.

- Корректность алгоритма следует из предшествующей леммы.
- Оценим сложность алгоритма.
- На этапе инициализации требуется $O(n)$ операций.
- Сложность одной итерации $O(n)$.
- А так как количество всех итераций равно $n-1$, то сложность всего алгоритма — $O(n^2)$.

□

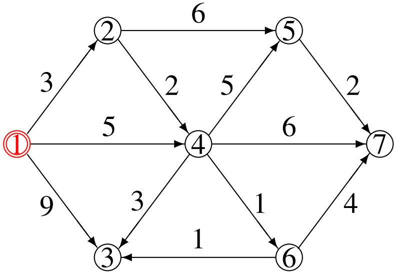
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



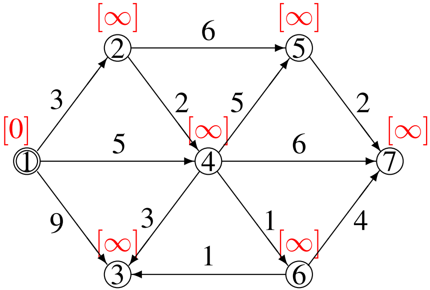
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



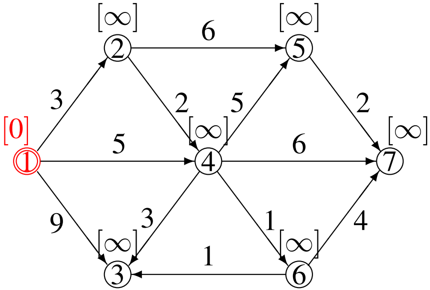
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



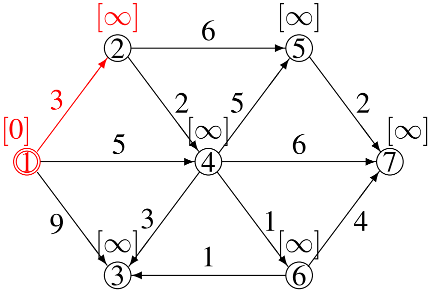
Пример

Найти кратчайшие пути от вершины 1 до всех остальных вершин.

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$



$d(1) + c(1,2) = 0 + 3 = 3 < \infty = d(2) \Rightarrow d(2) = 3.$

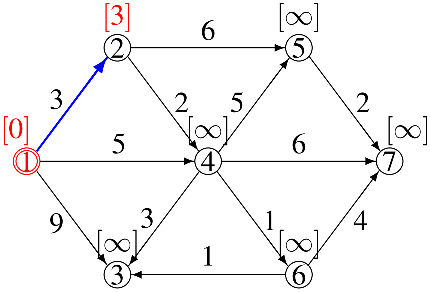
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(1) + c(1,2) = 0 + 3 = 3 < \infty = d(2) \;\Rightarrow\; d(2) = 3.$

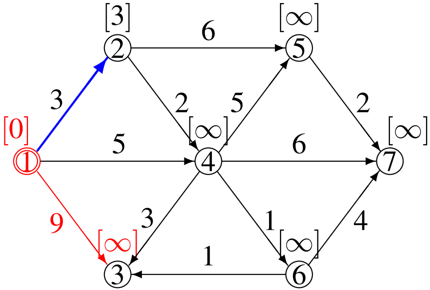
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1.$
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(1) + c(1,3) = 0 + 9 = 9 < \infty = d(3) \;\Rightarrow\; d(3) = 9.$

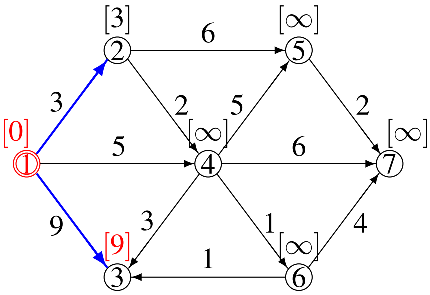
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(1) + c(1,3) = 0 + 9 = 9 < \infty = d(3) \;\Rightarrow\; d(3) = 9.$

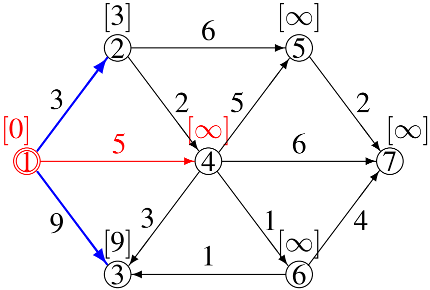
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1.$
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(1) + c(1,4) = 0 + 5 = 5 < \infty = d(4) \Rightarrow d(4) = 5.$

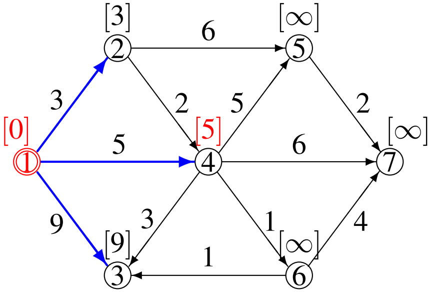
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(1) + c(1,4) = 0 + 5 = 5 < \infty = d(4) \;\Rightarrow\; d(4) = 5.$

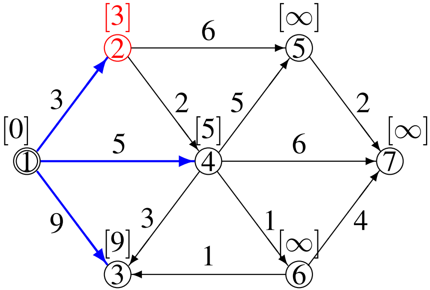
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



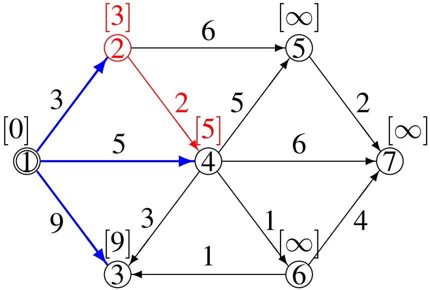
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(2)+c(2,4) = 3+2 = 5 = d(4)$.

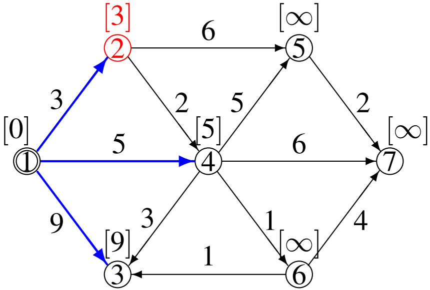
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



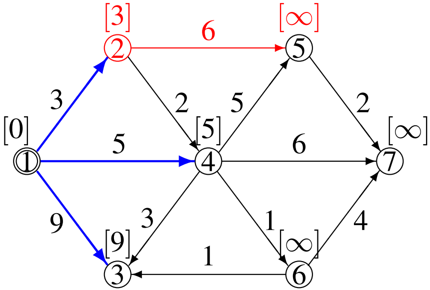
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(2) + c(2,5) = 3 + 6 = 9 < \infty = d(5) \;\Rightarrow\; d(5) = 9.$

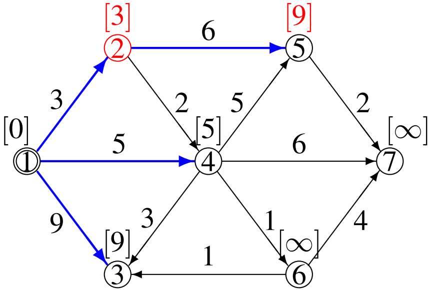
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(2) + c(2,5) = 3 + 6 = 9 < \infty = d(5)$ $\Rightarrow$ $d(5) = 9$.

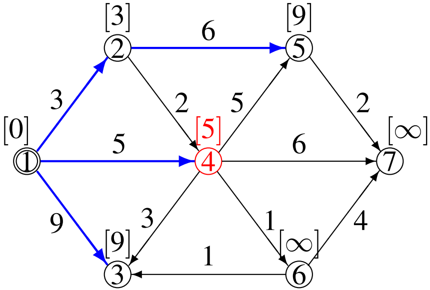
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1.$
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



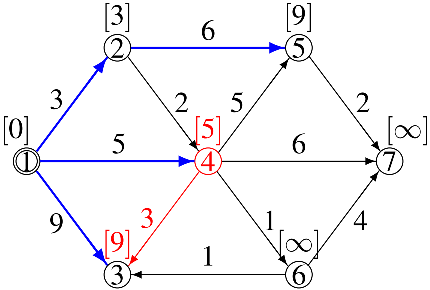
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(4) + c(4,3) = 5 + 3 = 8 < 9 = d(3) \;\Rightarrow\; d(3) = 8.$

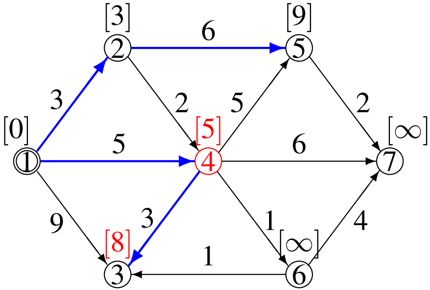
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(4) + c(4, 3) = 5 + 3 = 8 < 9 = d(3) \Rightarrow d(3) = 8$.

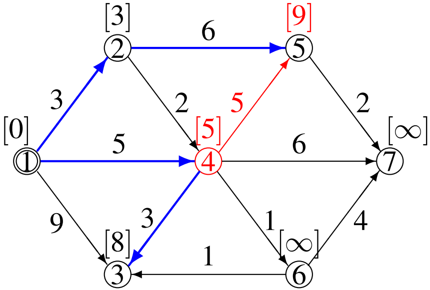
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(4) + c(4,5) = 5 + 5 = 10 > 9 = d(5)$.

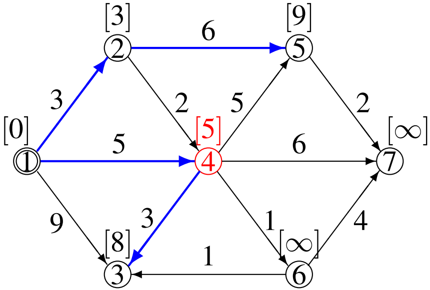
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(4) + c(4,5) = 5 + 5 = 10 >$
$9 = d(5)$.

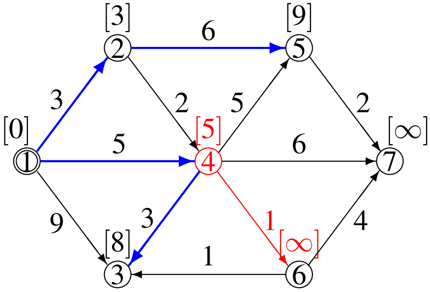
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(4) + c(4,6) = 5 + 1 = 6 < \infty = d(6) \Rightarrow d(6) = 6.$

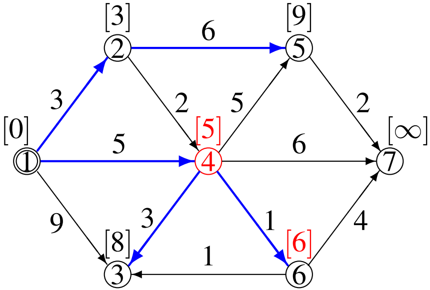
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(4) + c(4,6) = 5 + 1 = 6 < \infty = d(6) \Rightarrow d(6) = 6$.

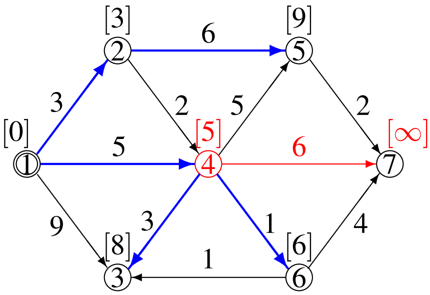
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(4) + c(4, 7) = 5 + 6 = 11 < \infty = d(7) \Rightarrow d(7) = 11.$

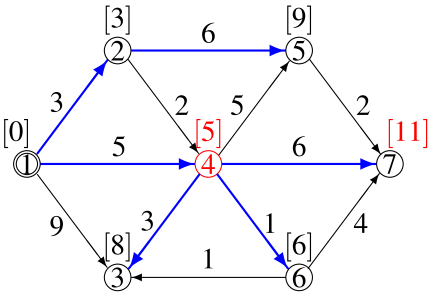
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(4) + c(4,7) = 5 + 6 = 11 < \infty = d(7) \;\Rightarrow\; d(7) = 11.$

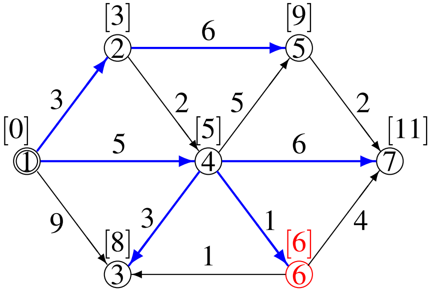
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1.$
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



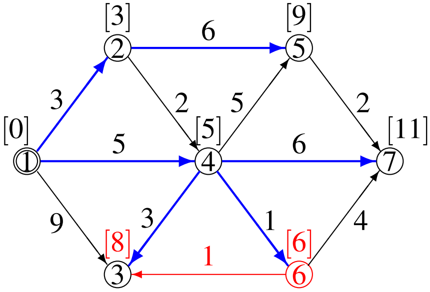
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(6) + c(6,3) = 6 + 1 = 7 < 8 = d(3) \Rightarrow d(3) = 7.$

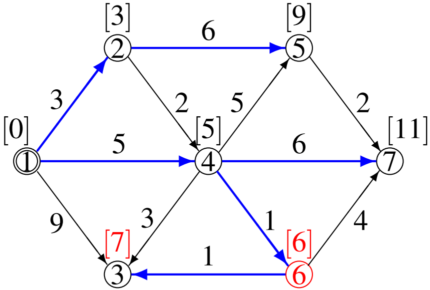
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(6) + c(6,3) = 6 + 1 = 7 < 8 = d(3) \Rightarrow d(3) = 7.$

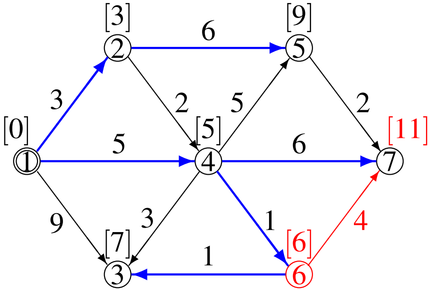
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(6) + c(6,7) = 6 + 4 = 10 < 11 = d(7) \;\Rightarrow\; d(7) = 10.$

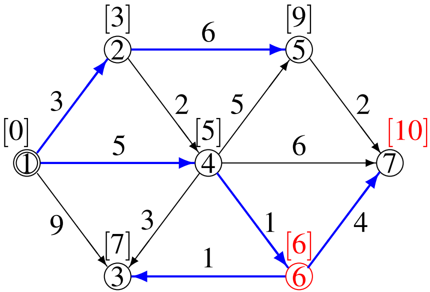
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1.$
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(6) + c(6,7) = 6 + 4 = 10 < 11 = d(7)$ $\Rightarrow$ $d(7) = 10.$

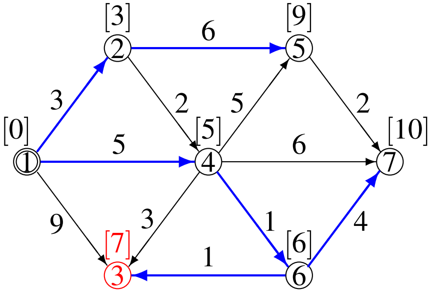
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



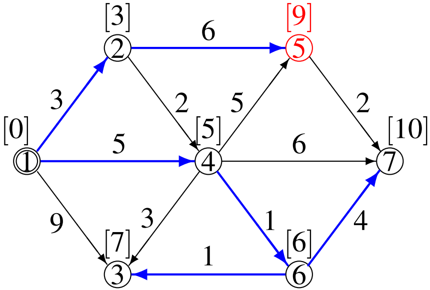
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



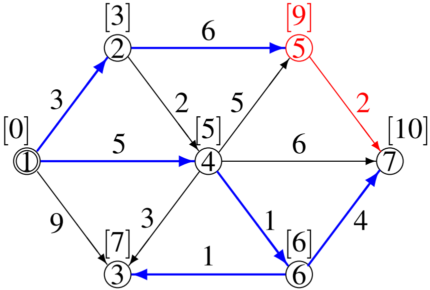
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(5) + c(5,7) = 9 + 2 = 11 > 10 = d(7)$.

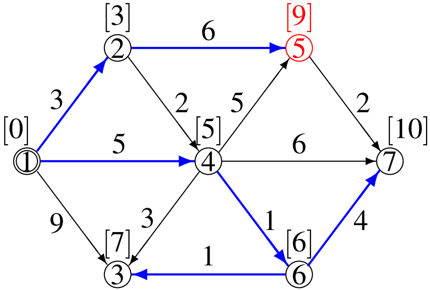
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1.$
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



$d(5) + c(5,7) = 9 + 2 = 11 > 10 = d(7).$

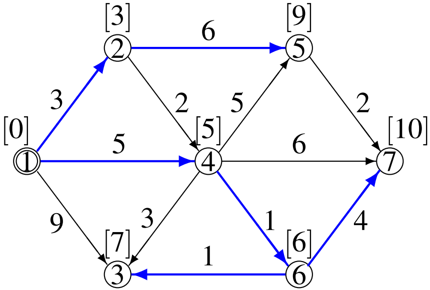
Пример

Обозначения:

- числа в квадр. скобках рядом с вершинами — текущие значения $d(v)$;
- дуги синего цвета — это дуги $(parent(v), v)$.

1. $w = 1$.
2. $w = 2$
3. $w = 4$
4. $w = 6$
5. $w = 3$
6. $w = 5$

Найти кратчайшие пути от вершины 1 до всех остальных вершин.



Ответ: Синие дуги образуют дерево кратчайших путей.

План лекции

1. Дерево кратчайших путей
   - Принцип оптимальности
   - Критерий существования дерева кратчайших путей
2. Алгоритмы поиска кратчайших путей
   - Алгоритм Форда — Беллмана
   - Алгоритм Дейкстры
   - Кратчайшие пути в ациклических графах

Постановка задачи

- Рассмотрим задачу поиска кратчайших путей в ациклическом орграфе $G = (V, E)$.
- Стоимости дуг $c(v, w)$ ($(v, w) \in E$) могут быть произвольные: как положительные, так и отрицательные.
- Так как в G нет отрицательных циклов, то все кратчайшие пути простые.

Постановка задачи

- Рассмотрим задачу поиска кратчайших путей в ациклическом орграфе $G = (V, E)$.
- Стоимости дуг $c(v, w)$ $((v, w) \in E)$ могут быть произвольные: как положительные, так и отрицательные.
- Так как в G нет отрицательных циклов, то все кратчайшие пути простые.

Постановка задачи

- Рассмотрим задачу поиска кратчайших путей в ациклическом орграфе $G = (V, E)$.
- Стоимости дуг $c(v, w)$ $((v, w) \in E)$ могут быть произвольные: как положительные, так и отрицательные.
- Так как в G нет отрицательных циклов, то все кратчайшие пути простые.

Топологическая сортировка

- Если граф G не имеет ориентированных циклов, то его можно топологически отсортировать,
- т. е. найти такую нумерацию $l : V \to \{1, \dots, n\}$ его вершин, для которой для каждой дуги $(v, w) \in E$ выполняется неравенство $l(v) < l(w)$.
- Сначала в графе G находим вершину v_1, в которую не входит ни одна дуга, и приписываем ей номер 1 ($l(v_1) = 1$);
- затем удаляем из графа вершину v_1 и все инцидентные ей дуги;
- в оставшемся подграфе снова находим вершину v_2, в которую не входит ни одна дуга, и приписываем ей номер 2 ($l(v_2) = 2$);
- так продолжаем до тех пор, пока все вешины не будут занумерованы.

Топологическая сортировка

- Если граф G не имеет ориентированных циклов, то его можно топологически отсортировать,
- т. е. найти такую нумерацию $l : V \to \{1, \dots, n\}$ его вершин, для которой для каждой дуги $(v, w) \in E$ выполняется неравенство $l(v) < l(w)$.
- Сначала в графе G находим вершину v_1, в которую не входит ни одна дуга, и приписываем ей номер 1 ($l(v_1) = 1$);
- затем удаляем из графа вершину v_1 и все инцидентные ей дуги;
- в оставшемся подграфе снова находим вершину v_2, в которую не входит ни одна дуга, и приписываем ей номер 2 ($l(v_2) = 2$);
- так продолжаем до тех пор, пока все вешины не будут занумерованы.

Топологическая сортировка

- Если граф G не имеет ориентированных циклов, то его можно топологически отсортировать,
- т. е. найти такую нумерацию $l : V \to \{1, \dots, n\}$ его вершин, для которой для каждой дуги $(v, w) \in E$ выполняется неравенство $l(v) < l(w)$.
- Сначала в графе G находим вершину v_1, в которую не входит ни одна дуга, и приписываем ей номер 1 $(l(v_1) = 1)$;
- затем удаляем из графа вершину v_1 и все инцидентные ей дуги;
- в оставшемся подграфе снова находим вершину v_2, в которую не входит ни одна дуга, и приписываем ей номер 2 $(l(v_2) = 2)$;
- так продолжаем до тех пор, пока все вешины не будут занумерованы.

Топологическая сортировка

- Если граф G не имеет ориентированных циклов, то его можно топологически отсортировать,
- т. е. найти такую нумерацию $l : V \to \{1, \ldots, n\}$ его вершин, для которой для каждой дуги $(v, w) \in E$ выполняется неравенство $l(v) < l(w)$.
- Сначала в графе G находим вершину v_1, в которую не входит ни одна дуга, и приписываем ей номер 1 ($l(v_1) = 1$);
- затем удаляем из графа вершину v_1 и все инцидентные ей дуги;
- в оставшемся подграфе снова находим вершину v_2, в которую не входит ни одна дуга, и приписываем ей номер 2 ($l(v_2) = 2$);
- так продолжаем до тех пор, пока все вешины не будут занумерованы.

Топологическая сортировка

- Если граф G не имеет ориентированных циклов, то его можно топологически отсортировать,
- т. е. найти такую нумерацию $l : V \to \{1, \ldots, n\}$ его вершин, для которой для каждой дуги $(v, w) \in E$ выполняется неравенство $l(v) < l(w)$.
- Сначала в графе G находим вершину v_1, в которую не входит ни одна дуга, и приписываем ей номер 1 $(l(v_1) = 1)$;
- затем удаляем из графа вершину v_1 и все инцидентные ей дуги;
- в оставшемся подграфе снова находим вершину v_2, в которую не входит ни одна дуга, и приписываем ей номер 2 $(l(v_2) = 2)$;
- так продолжаем до тех пор, пока все вешины не будут занумерованы.

Топологическая сортировка

- Если граф G не имеет ориентированных циклов, то его можно топологически отсортировать,
- т. е. найти такую нумерацию $l : V \to \{1, \dots, n\}$ его вершин, для которой для каждой дуги $(v, w) \in E$ выполняется неравенство $l(v) < l(w)$.
- Сначала в графе G находим вершину v_1, в которую не входит ни одна дуга, и приписываем ей номер 1 ($l(v_1) = 1$);
- затем удаляем из графа вершину v_1 и все инцидентные ей дуги;
- в оставшемся подграфе снова находим вершину v_2, в которую не входит ни одна дуга, и приписываем ей номер 2 ($l(v_2) = 2$);
- так продолжаем до тех пор, пока все вешины не будут занумерованы.

Рекурентные формулы

- Считаем, что $V = \{1,\dots,n\}$ и $i < j$ для всех $(i,j) \in E$.
- Наша цель — вычислить длины $d(j)$ кратчайший путей в графе G от вершины 1 до всех остальных вершин $j = 2,\dots,n$.
- Из принципа оптимальности вытекает справедливость следующей рекурентной формулы:

$$
\begin{aligned}
d(1) &= 0,\\
d(j) &= \min_{(i,j)\in E} \left(d(i) + c(i,j)\right), \quad j = 2,\dots,n.
\end{aligned}
$$

- Как обычно, минимум по пустому множеству альтернатив равен $+\infty$.

Рекурентные формулы

- Считаем, что $V = \{1, \dots, n\}$ и $i < j$ для всех $(i,j) \in E$.
- Наша цель — вычислить длины $d(j)$ кратчайший путей в графе G от вершины 1 до всех остальных вершин $j = 2, \dots, n$.
- Из принципа оптимальности вытекает справедливость следующей рекурентной формулы:

$$d(1) = 0,$$

$$d(j) = \min_{(i,j)\in E} \left(d(i) + c(i,j)\right), \quad j = 2, \dots, n.$$

- Как обычно, минимум по пустому множеству альтернатив равен $+\infty$.

Рекурентные формулы

- Считаем, что $V = \{1, \ldots, n\}$ и $i < j$ для всех $(i,j) \in E$.
- Наша цель — вычислить длины $d(j)$ кратчайший путей в графе G от вершины 1 до всех остальных вершин $j = 2, \ldots, n$.
- Из принципа оптимальности вытекает справедливость следующей рекурентной формулы:

$$d(1) = 0,$$

$$d(j) = \min_{(i,j)\in E} \left(d(i) + c(i,j)\right), \quad j = 2, \ldots, n.$$

- Как обычно, минимум по пустому множеству альтернатив равен $+\infty$.

Рекурентные формулы

- Считаем, что $V = \{1, \dots, n\}$ и $i < j$ для всех $(i,j) \in E$.
- Наша цель — вычислить длины $d(j)$ кратчайший путей в графе G от вершины 1 до всех остальных вершин $j = 2, \dots, n$.
- Из принципа оптимальности вытекает справедливость следующей рекурентной формулы:

$$d(1) = 0,$$

$$d(j) = \min_{(i,j)\in E} \left(d(i) + c(i,j)\right), \quad j = 2, \dots, n.$$

- Как обычно, минимум по пустому множеству альтернатив равен $+\infty$.

Рекурентные формулы

- Считаем, что $V = \{1, \dots, n\}$ и $i < j$ для всех $(i,j) \in E$.
- Наша цель — вычислить длины $d(j)$ кратчайший путей в графе G от вершины 1 до всех остальных вершин $j = 2, \dots, n$.
- Из принципа оптимальности вытекает справедливость следующей рекурентной формулы:

$$
\begin{aligned}
&d(1) = 0,\\
&d(j) = \min_{(i,j)\in E} \left(d(i) + c(i,j)\right), \quad j = 2, \dots, n.
\end{aligned}
$$

- Как обычно, минимум по пустому множеству альтернатив равен $+\infty$.

Рекурентные формулы

- Считаем, что $V = \{1, \ldots, n\}$ и $i < j$ для всех $(i,j) \in E$.
- Наша цель — вычислить длины $d(j)$ кратчайший путей в графе G от вершины 1 до всех остальных вершин $j = 2, \ldots, n$.
- Из принципа оптимальности вытекает справедливость следующей рекурентной формулы:

$$d(1) = 0,$$
$$d(j) = \min_{(i,j)\in E} \left(d(i) + c(i,j)\right), \quad j = 2, \ldots, n.$$

- Как обычно, минимум по пустому множеству альтернатив равен $+\infty$.

Обратный ход

Зная значения $d(j)$, мы можем можем определить дерево кратчайших путей, выполнив *обратный ход*:

$$parent(j) \in \{i : \ (i,j) \in E \text{ и } d(j) = d(i) + c(i,j)\}, \quad j = n, n-1, \ldots, 2,$$

$$parent(1) = \text{nil}.$$

Обратный ход

Знаая значения $d(j)$, мы можем можем определить дерево кратчайших путей, выполнив *обратный ход*:

$$parent(j) \in \{i: \ (i,j) \in E \text{ и } d(j) = d(i) + c(i,j)\}, \quad j = n, n-1, \ldots, 2,$$

$$parent(1) = \text{nil}.$$

Обратный ход

Зная значения $d(j)$, мы можем можем определить дерево кратчайших путей, выполнив *обратный ход*:

$$parent(j) \in \{i : \ (i,j) \in E \text{ и } d(j) = d(i) + c(i,j)\}, \quad j = n, n-1, \dots, 2,$$

$$parent(1) = \text{nil}.$$

Пути максимальной стоимости

- В дальнейшем мы также будем рассматривать задачи, решение которых сводится к поиску путей максимальной стоимости (длины) в ациклическом графе.
- Справедлива следующая рекурентная формула:

$$d(1) = 0,$$

$$d(j) = \max_{(i,j)\in E} \left(d(i) + c(i,j)\right), \quad j = 2, \dots, n.$$

- Максимум по пустому множеству альтернатив равен $-\infty$.

Пути максимальной стоимости

- В дальнейшем мы также будем рассматривать задачи, решение которых сводится к поиску путей максимальной стоимости (длины) в ациклическом графе.
- Справедлива следующая рекурентная формула:

$$d(1) = 0,$$
$$d(j) = \max_{(i,j)\in E} \left(d(i) + c(i,j)\right), \quad j = 2,\dots,n.$$

- Максимум по пустому множеству альтернатив равен $-\infty$.

Пути максимальной стоимости

- В дальнейшем мы также будем рассматривать задачи, решение которых сводится к поиску путей максимальной стоимости (длины) в ациклическом графе.
- Справедлива следующая рекурентная формула:

$$d(1) = 0,$$
$$d(j) = \max_{(i,j)\in E} \left(d(i) + c(i,j)\right), \quad j = 2, \ldots, n.$$

- Максимум по пустому множеству альтернатив равен $-\infty$.

Пути максимальной стоимости

- В дальнейшем мы также будем рассматривать задачи, решение которых сводится к поиску путей максимальной стоимости (длины) в ациклическом графе.
- Справедлива следующая рекурентная формула:

$$d(1) = 0,$$
$$d(j) = \max_{(i,j)\in E} \left(d(i) + c(i,j)\right), \quad j = 2, \dots, n.$$

- Максимум по пустому множеству альтернатив равен $-\infty$.

Пути максимальной стоимости

- В дальнейшем мы также будем рассматривать задачи, решение которых сводится к поиску путей максимальной стоимости (длины) в ациклическом графе.
- Справедлива следующая рекурентная формула:

$$d(1) = 0,$$
$$d(j) = \max_{(i,j)\in E} \left(d(i) + c(i,j)\right), \quad j = 2, \dots, n.$$

- Максимум по пустому множеству альтернатив равен $-\infty$.