Кратчайшие пути в графах

Виктор Васильевич Лепин

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*)

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*)
$$c(P) = \sum_{i=1}^{k} c(v_{i-1}, v_i).$$
2. кратчайшие пути от выделенной вершины $s \in V$ до всех остальных вершин графа.

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*)
$$c(P) = \sum_{i=1}^{k} c(v_{i-1}, v_i).$$
2. кратчайшие пути от выделенной вершины $s \in V$ до всех остальных вершин графа.
3. кратчайшие пути от всех вершин до выделенной вершины $t \in V$.

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*)
$$c(P) = \sum_{i=1}^{k} c(v_{i-1}, v_i).$$
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 $s, t \in V$.

Задачи о кратчайших путях

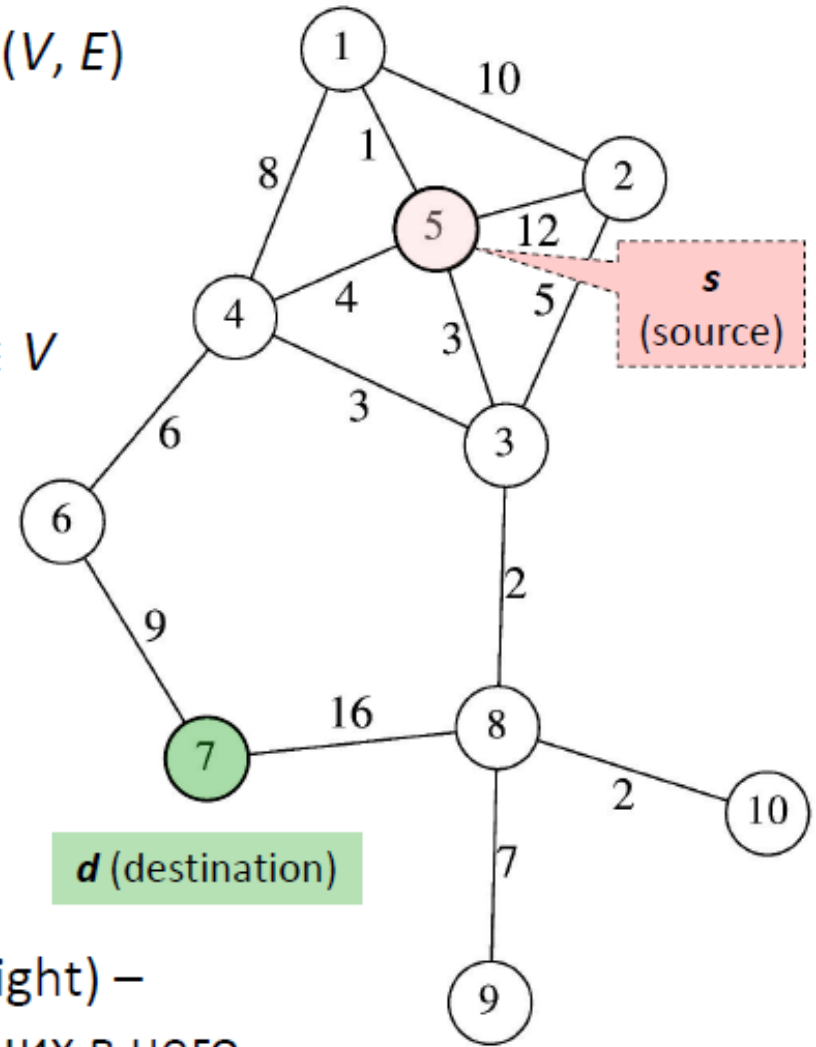
В орграфе $G = (V, E)$, каждой дуге которого приписана *стоимость (длина)* $c(v, w)$, нужно найти

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

Формулировка задачи о кратчайшем пути в графе

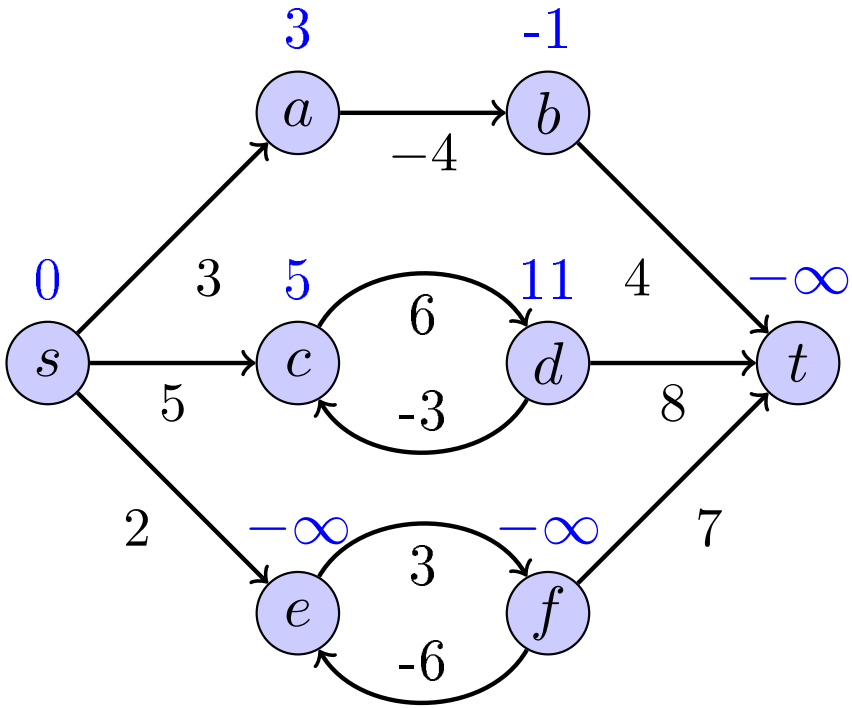
- **Имеется** взвешенный граф $G = (V, E)$
- Каждому ребру $(i, j) \in E$ назначен вес w_{ij}
- Заданы начальная вершина $s \in V$ и конечная $d \in V$
- **Требуется** найти кратчайший путь из вершины s в вершину d (shortest path problem)
- Длина пути (path length, path cost, path weight) – это сумма весов ребер, входящих в него



Алгоритмы

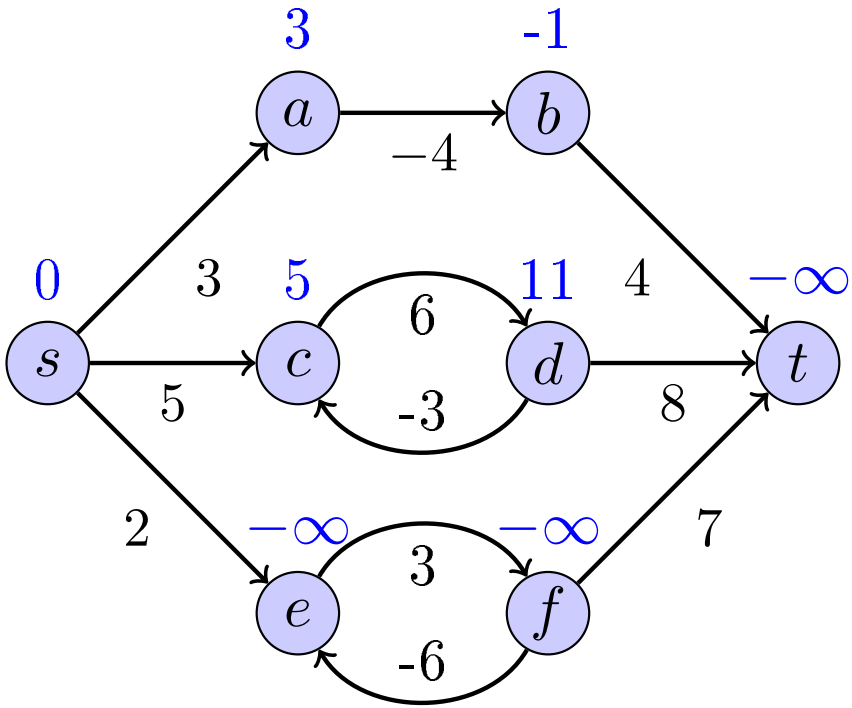
Алгоритм	Применение
Алгоритм Дейкстры	Находит кратчайший путь от одной из вершин графа до всех остальных. Алгоритм работает только для графов без ребер отрицательного веса ($w_{ij} \geq 0$)
Алгоритм Беллмана-Форда	Находит кратчайшие пути от одной вершины графа до всех остальных во взвешенном графе. Вес ребер может быть отрицательным
Алгоритм поиска A* (A star)	Находит путь с наименьшей стоимостью от одной вершины к другой, используя алгоритм поиска по первому наилучшему совпадению на графе
Алгоритм Флойда-Уоршелла	Находит кратчайшие пути между всеми вершинами взвешенного ориентированного графа
Алгоритм Джонсона	Находит кратчайшие пути между всеми парами вершин взвешенного ориентированного графа (должны отсутствовать циклы с отрицательным весом)
Алгоритм Ли (волновой алгоритм)	Находит путь между вершинами s и t графа, содержащий минимальное количество промежуточных вершин (трассировки электрических соединений на кристаллах микросхем и на печатных платах)
Алгоритмы Viterbi, Cherkassky, ...	

Задача ShortestPath и циклы



- Здесь $d(i, j)$ может быть отрицательным; однако не должно быть **циклов отрицательного веса**.

Задача ShortestPath и циклы



- Здесь $d(i, j)$ может быть отрицательным; однако не должно быть **циклов отрицательного веса**.
- Фактически, существование отрицательного цикла означает, что существует путь, имеющий вес $-\infty$. Поскольку e и f образуют цикл с отрицательным весом, достижимый из s, то есть путь, имеющий вес $-\infty$ из s.

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

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.

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,

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_t из s в t.

Уравнения Беллмана

- Стоимость кратчайшего пути в графе G от вершины s до вершины $v \in V$ обозначим через $\sigma(s, v)$

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 $\sigma(s, v) = +\infty$).

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,

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

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:

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:

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

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:

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

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

Приведенные стоимости

- *Функция цен* есть функция $p : V \to \mathbb{R}$.

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

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.

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 $p(v)$

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

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.

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

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 $c_p(P) = c(P)$.

Функция расстояний

- Для покрывающего ордерева T с корнем s *функция расстояний* $d : V \to \mathbb{R}$ определяется рекурсивно следующим образом:

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 $v \in V \setminus s$,

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.

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*,

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

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. $d(v) = \sigma(s, v)$.

Теорема 2

- Пусть все вершины графа $G = (V, E)$ достигаются из вершины $s \in V$.

Теорема 2

- Пусть все вершины графа $G = (V, E)$ достигаются из вершины $s \in V$.
- Граф G имеет дерево кратчайшийх путей тогда и только тогда, когда он не имеет циклов отрицательной стоимости.

Теорема 2

- Пусть все вершины графа $G = (V, E)$ достигаются из вершины $s \in V$.
- Граф G имеет дерево кратчайшийх 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:

Теорема 2

- Пусть все вершины графа $G = (V, E)$ достигаются из вершины $s \in V$.
- Граф G имеет дерево кратчайшийх 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:

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

Следствие 3

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

Алгоритмы поиска кратчайших путей

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{nil\}$, таких, что
  $d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$

Метод последовательной аппроксимации

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

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

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

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{nil\}$, таких, что
  $d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$
  $parent(v) = \mathsf{nil}$, $v \in V$.
- Затем итеративно повторяется следующий шаг:

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{nil\}$, таких, что
  $d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$
  $parent(v) = \mathsf{nil}$, $v \in V$.
- Затем итеративно повторяется следующий шаг:
  - выбрать дугу (v, w) отрицательной приведенной стоимости $c_d(v, w) < 0$,

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{nil\}$, таких, что
  $d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$
  $parent(v) = \mathsf{nil}$, $v \in V$.
- 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 $d(v) + c(v, w)$.

Метод последовательной аппроксимации

- Идея всех алгоритмов поиска кратч. путей одинакова.
- Все они начинают с функций
- $d : V \to \mathbb{R} \cup \{\infty\}$ и $parent : V \to V \cup \{nil\}$, таких, что
  $d(v) = \begin{cases} 0, & v = s, \\ \infty, & v \in V \setminus \{s\}, \end{cases}$
  $parent(v) = \mathsf{nil}$, $v \in V$.
- 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

Свойства метода последовательной аппроксимации

- Если в графе нет циклов отрицательной стоимости, то
- метод заканчивает работу после конечного числа итераций, когда стоимости дуг целочисленны.

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.

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,

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 $parent(v) \neq \mathsf{nil}$, содержит цикл.

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 $parent(v) \neq \mathsf{nil}$, содержит цикл.
- Из описания метода последовательной аппроксимации следует, что стоимость этого цикла отрицательна.

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 $parent(v) \neq \mathsf{nil}$, содержит цикл.
- Из описания метода последовательной аппроксимации следует, что стоимость этого цикла отрицательна.
- Эффективность метода зависит от порядка выбора дуг отрицательной приведенной стоимости.

Алгоритм Беллмана-Форда

- **Вход:** Орграф $G = (V, E)$, ф-ция $c : E \to R$, верш. $s \in V$.

Алгоритм Беллмана-Форда

- **Вход:** Орграф $G = (V, E)$, ф-ция $c : E \to R$, верш. $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, функция $d : V \to \mathbb{R} \cup \{\infty\}$:
  1. если указатели $parent$ представляют дерево, то $d(v)$ — кратчайшее расстояние от s до v;

Алгоритм Беллмана-Форда

- **Вход:** Орграф $G = (V, E)$, ф-ция $c : E \to R$, верш. $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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.

Алгоритм Беллмана-Форда

- **Вход:** Орграф $G = (V, E)$, ф-ция $c : E \to R$, верш. $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.

Алгоритм Беллмана-Форда

- **Вход:** Орграф $G = (V, E)$, ф-ция $c : E \to R$, верш. $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.

Алгоритм Беллмана-Форда

- **Вход:** Орграф $G = (V, E)$, ф-ция $c : E \to R$, верш. $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.
- 3. Для $i = 1, \ldots, n$ выполнить шаги 3.1–3.4:

Алгоритм Беллмана-Форда

- **Вход:** Орграф $G = (V, E)$, ф-ция $c : E \to R$, верш. $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \{s\}$.
- 3. Для $i = 1, \ldots, n$ выполнить шаги 3.1–3.4:
  - 3.1. $Q = \emptyset$, $\bar{d} = d$.

Алгоритм Беллмана-Форда

- **Вход:** Орграф $G = (V, E)$, ф-ция $c : E \to R$, верш. $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.
- 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 $d(w) := \bar{d}(v) + c(v, w)$, $parent(w) := v$, $Q := Q \cup \{w\}$.

Алгоритм Беллмана-Форда

- **Вход:** Орграф $G = (V, E)$, ф-ция $c : E \to R$, верш. $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.
- 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 $S = Q$.

Алгоритм Беллмана-Форда

- **Вход:** Орграф $G = (V, E)$, ф-ция $c : E \to R$, верш. $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.
- 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**.

- Обозначим через $\sigma^i(s, v)$ стоимость кратчайшего пути из s в v среди всех путей, которые имеют ровно i дуг.

- 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 $\sigma^i(s, v) = \emptyset$.

Анализ алгоритма Беллмана-Форда

- 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 $\sigma^i(s, v) = \emptyset$.
- Пусть $d^i(v)$, S^i — соответственно $d(v)$ и список S после итерации i.

Анализ алгоритма Беллмана-Форда

- 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 $\sigma^i(s, v) = \emptyset$.
- 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.

Лемма 4

Справедливы следующие соотношения:

Лемма 4

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

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 $v \in S^i$,

Лемма 5

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

Лемма 6

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

Сложность алгоритма Беллмана-Форда

Теорема 7

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

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 $c(\Gamma) < 0$,

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.

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 Беллмана-Форда может найти отр. цикл,

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 Беллмана-Форда может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.

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 Беллмана-Форда может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,

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 Беллмана-Форда может найти отр. цикл,
- если этот цикл достижим из стартовой вершины.
- Чтобы гарантировать это,
  - добавим к графу G новую вершину 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 Беллмана-Форда 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 $v \in V$.

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 Беллмана-Форда 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^t
  1. каждая вершина достижима из вершины 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 Беллмана-Форда 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^t
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G^t также является циклом в G.

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 Беллмана-Форда 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^t
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G^t также является циклом в G.
- Применим алгоритм Беллмана-Форда к графу G^t с начальной вершиной 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 Беллмана-Форда 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^t
  1. каждая вершина достижима из вершины s
  2. и любой цикл в G^t также является циклом в G.
- Применим алгоритм Беллмана-Форда к графу G^t с начальной вершиной s,
- и, если в G есть отрицательный цикл, то алгоритм найдет его.

Алгоритм Дейкстры

Алгоритм Дейкстры

- **Алгоритм Дейкстры** (Dijkstra's algorithm, 1959) – алгоритм поиска кратчайшего пути в графе из заданной вершины во все остальные (single-source shortest path problem)
- Находит кратчайшее *расстояние* от одной из вершин графа до всех остальных
- Применим только для графов без ребер отрицательного веса и петель ($w_{ij} \geq 0$)

- **Эдсгер Дейкстра** (Edsger Wybe Dijkstra) – нидерландский ученый (структурное программирование, язык Алгол, семафоры, распределенные вычисления)
- Лауреат премии Тьюринга (ACM A.M. Turing Award)

1. Дейкстра Э. Дисциплина программирования = A discipline of programming. — М.: Мир, 1978. — С. 275.
2. Дал У., Дейкстра Э., Хоор К. Структурное программирование = Structured Programming. — М.: Мир, 1975. — С. 247.

Неотрицательные стоимости

- Алгоритм Дейкстры применяется, когда стоимости всех дуг неотрицательные.

Неотрицательные стоимости

- Алгоритм Дейкстры применяется, когда стоимости всех дуг неотрицательные.
- Это также алгоритм последовательной аппроксимации,

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.

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.

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.

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” $d(w)$, добавляет ее к S,

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" $d(w)$, добавляет 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 $c : E \to R$, вершина $s \in V$.

Описание алгоритма Дейкстры

- **Вход:** Орграф $G = (V, E)$, функция $c : E \to R$, вершина $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, задающие дерево кратчайших путей, функция $d : V \to \mathbb{R} \cup \{\infty\}$, где $d(v)$ — *кратчайшее расстояние* от s до v.

Описание алгоритма Дейкстры

- **Вход:** Орграф $G = (V, E)$, функция $c : E \to R$, вершина $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.

Описание алгоритма Дейкстры

- **Вход:** Орграф $G = (V, E)$, функция $c : E \to R$, вершина $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \emptyset$.

Описание алгоритма Дейкстры

- **Вход:** Орграф $G = (V, E)$, функция $c : E \to R$, вершина $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.
- 2. Положить $d(s) = 0$ и $S = \emptyset$.
- 3. Пока $|S| < n - 1$ выполнять шаги 3.1 и 3.2:

Описание алгоритма Дейкстры

- **Вход:** Орграф $G = (V, E)$, функция $c : E \to R$, вершина $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.
- 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 $S := S \cup \{w\}$.

Описание алгоритма Дейкстры

- **Вход:** Орграф $G = (V, E)$, функция $c : E \to R$, вершина $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.
- 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 $c : E \to R$, вершина $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.
- 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 $c : E \to R$, вершина $s \in V$.
- **Выход:** указатели $parent : V \to V \cup \{\mathsf{nil}\}$, 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 $parent(v) = \mathsf{nil}$.
- 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 $parent(v) = w$.

Лемма 8

Алгоритм Дейкстры поддерживает следующие инварианты:

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

Сложность алгоритма Дейкстры

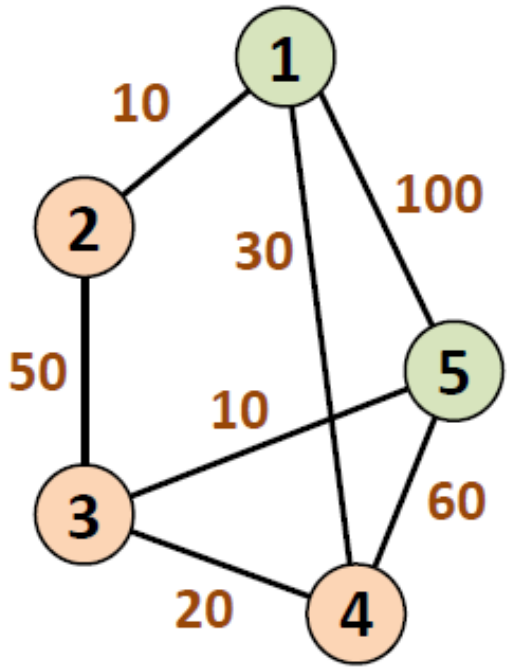
Теорема 9

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

Алгоритм Дейкстры

- **Пример:** найти кратчайший путь из вершины 1 в вершину 5

- Введем обозначения:
  - H – множество посещенных вершин
  - $D[i]$ – текущее известное кратчайшее расстояние от вершины s до вершины i
  - prev[i] – номер вершины, предшествующей i в пути

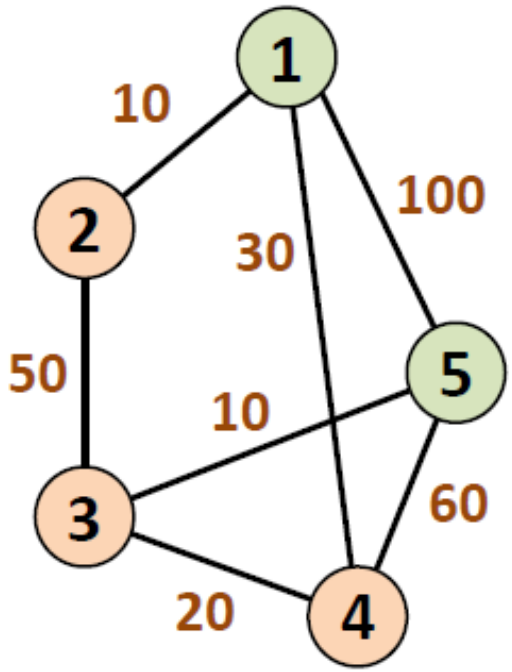


Алгоритм Дейкстры

1. Устанавливаем расстояние $D[i]$ от начальной вершины s до всех остальных в ∞

2. Полагаем $D[s] = 0$

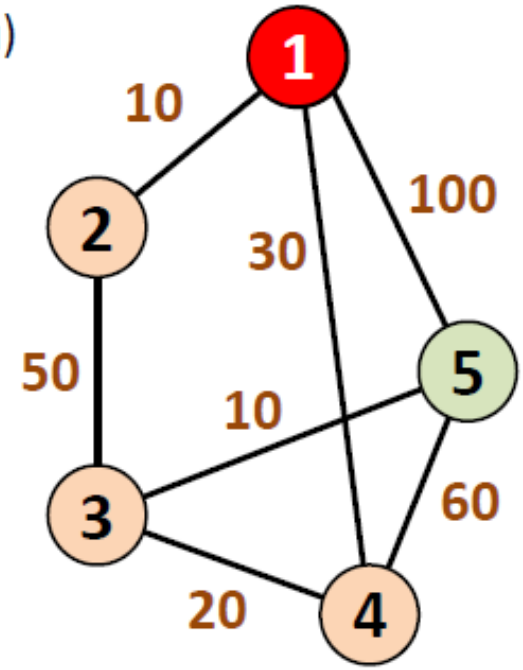
3. Помещаем все вершины в очередь с приоритетом Q (min-heap): приоритет вершины i это значение $D[i]$



	1	2	3	4	5
$D[i]$	0	∞	∞	∞	∞

Алгоритм Дейкстры

4. Запускаем цикл из *n* итераций (по числу вершин)
   1. Извлекаем из очереди *Q* вершину *v* с минимальным приоритетом – ближайшую к *s* вершину
   2. Отмечаем вершину *v* как посещенную (помещаем *v* во множество *H*)
   3. Возможно пути из *s* через вершину *v* стали короче, выполняем проверку: для каждой вершины *u* смежной с вершиной *v* и не включенной в *H* проверяем и корректируем расстояние *D*[*u*]



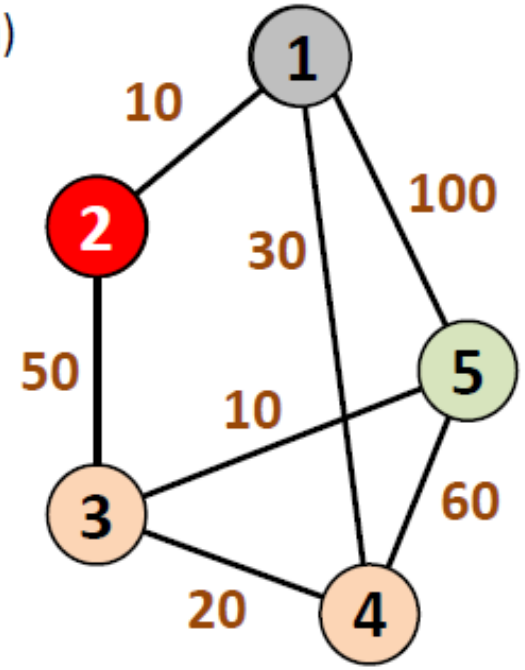

```
if D[v] + w(v, u) < D[u] then
    // Путь из s до u через (v, u) короче
    D[u] = D[v] + w(v, u)
    PrioQueueDecrease(Q, u, D[u])
    prev[u] = v
end if
```

D[2] = 10
D[4] = 30
D[5] = 100

D[i]	0	10	∞	30	100

Алгоритм Дейкстры

4. Запускаем цикл из n итераций (по числу вершин)
   1. Извлекаем из очереди Q вершину v с минимальным приоритетом – ближайшую к s вершину
   2. Отмечаем вершину v как посещенную (помещаем v во множество H)
   3. Возможно пути из s через вершину v стали короче, выполняем проверку: для каждой вершины u смежной с вершиной v и не включенной в H проверяем и корректируем расстояние $D[u]$



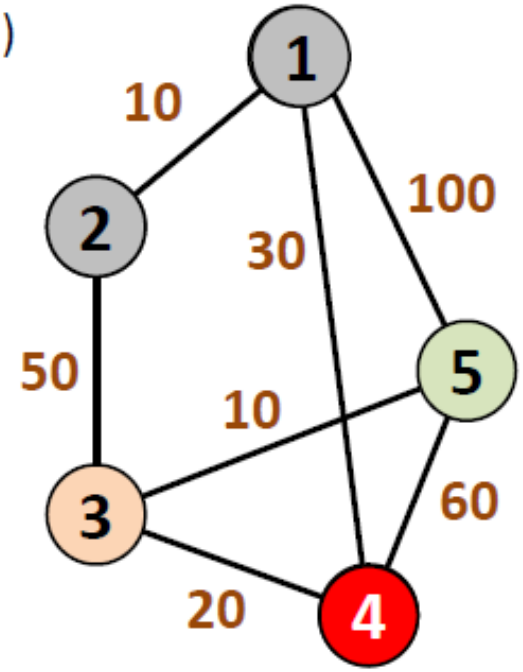
D[3] = 60

```
if D[v] + w(v, u) < D[u] then
    // Путь из s до u через (v, u) короче
    D[u] = D[v] + w(v, u)
    PrioQueueDecrease(Q, u, D[u])
    prev[u] = v
end if
```

D[i]	0	10	60	30	100

Алгоритм Дейкстры

4. Запускаем цикл из n итераций (по числу вершин)
   1. Извлекаем из очереди Q вершину v с минимальным приоритетом – ближайшую к s вершину
   2. Отмечаем вершину v как посещенную (помещаем v во множество H)
   3. Возможно пути из s через вершину v стали короче, выполняем проверку: для каждой вершины u смежной с вершиной v и не включенной в H проверяем и корректируем расстояние $D[u]$



$D[3] = 50$

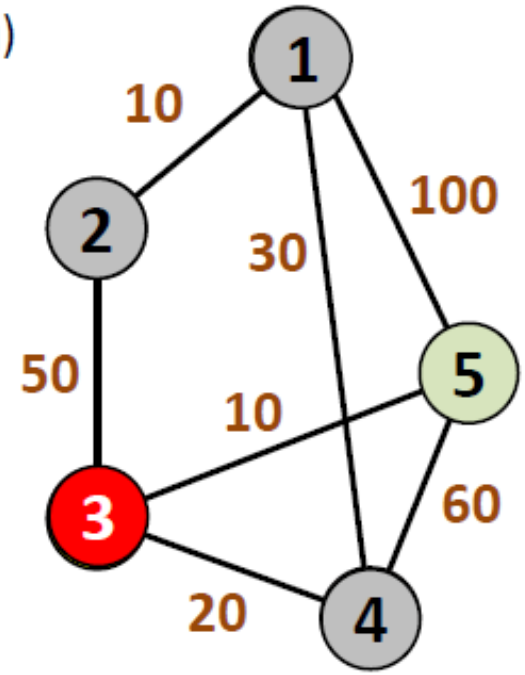
$D[5] = 90$

```
if D[v] + w(v, u) < D[u] then
    // Путь из s до u через (v, u) короче
    D[u] = D[v] + w(v, u)
    PrioQueueDecrease(Q, u, D[u])
    prev[u] = v
end if
```

$D[i]$	0	10	50	30	90

Алгоритм Дейкстры

4. Запускаем цикл из n итераций (по числу вершин)
   1. Извлекаем из очереди Q вершину v с минимальным приоритетом – ближайшую к s вершину
   2. Отмечаем вершину v как посещенную (помещаем v во множество H)
   3. Возможно пути из s через вершину v стали короче, выполняем проверку: для каждой вершины u смежной с вершиной v и не включенной в H проверяем и корректируем расстояние $D[u]$



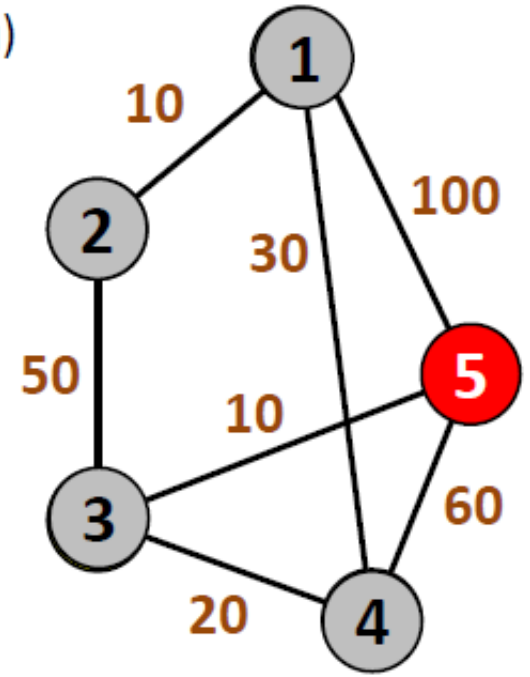
$D[5] = 60$

```
if D[v] + w(v, u) < D[u] then
    // Путь из s до u через (v, u) короче
    D[u] = D[v] + w(v, u)
    PrioQueueDecrease(Q, u, D[u])
    prev[u] = v
end if
```

$D[i]$	0	10	50	30	60

Алгоритм Дейкстры

4. Запускаем цикл из n итераций (по числу вершин)
   1. Извлекаем из очереди Q вершину v с минимальным приоритетом – ближайшую к s вершину
   2. Отмечаем вершину v как посещенную (помещаем v во множество H)
   3. Возможно пути из s через вершину v стали короче, выполняем проверку: для каждой вершины u смежной с вершиной v и не включенной в H проверяем и корректируем расстояние $D[u]$



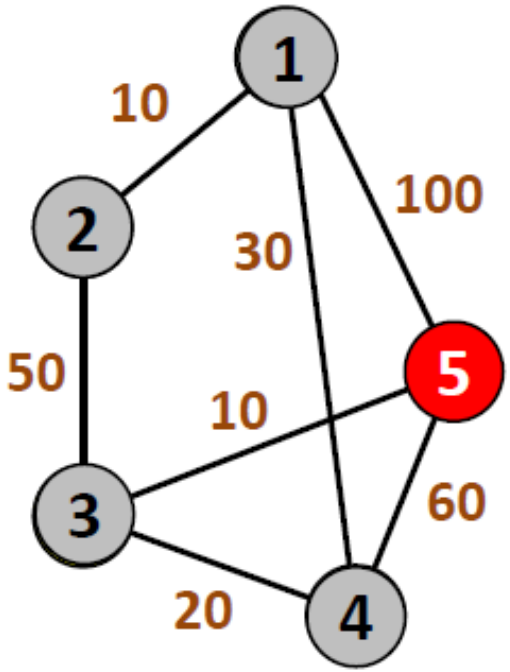

```
if D[v] + w(v, u) < D[u] then
    // Путь из s до u через (v, u) короче
    D[u] = D[v] + w(v, u)
    PrioQueueDecrease(Q, u, D[u])
    prev[u] = v
end if
```

$D[i]$	0	10	50	30	60

Алгоритм Дейкстры

- В массиве D[1:*n*] содержатся длины кратчайших путей из начальной вершины *s* = 1
  - *D*[1] – длина пути из 1 в 1
  - *D*[2] – длина пути из 1 в 2
  - *D*[3] – длина пути из 1 в 3
  - *D*[4] – длина пути из 1 в 4
  - *D*[5] – длина пути из 1 в 5

***D*[*i*]**	**0**	**10**	**50**	**30**	**60**



- **Как определить какие *вершины* входят в кратчайший путь из *s* = 1 в *d* = 5?**
- **Как восстановить путь?**

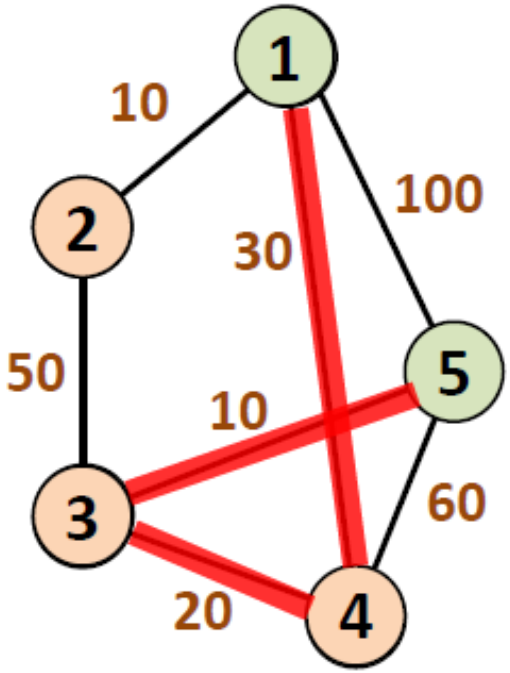
Алгоритм Дейкстры

- **Восстановление кратчайшего пути**
- Массив prev[*i*] содержит номер вершины, предшествующей *i* в пути

	1	2	3	4	5
prev[*i*]	-1	1	4	1	3

- **Восстанавливаем путь с конца**
  - Вершина 5
  - Вершина prev[5] = 3
  - Вершина prev[3] = 4
  - Вершина prev[4] = 1

Кратчайший путь (1, 4, 3, 5)



D[*i*]	0	10	50	30	60

Алгоритм Дейкстры

```
function ShortestPath_Dijkstra(G, src, d, prev)
    // Input: G = (V, E), src, dst
    // Output: d[1:n], prev[1:n]
    //   prev[i] - узел, предшествующий i в пути

    // Помещаем вершины в очередь с приоритетом
    for each i in V \ {src} do
        d[i] = Infinity
        prev[i] = -1
        PriorityQueueInsert(Q, i, d[i])
    end for

    d[src] = 0
    prev[src] = -1
    PriorityQueueInsert(Q, src, d[src])
```

Алгоритм Дейкстры

```
    for i = 0 to n - 1 do
        // Извлекаем узел ближайший к начальному
        v = PriorityQueueRemoveMin(Q)
        // Отмечаем v как посещенный
        H = H + {v}

        // Цикл по смежным вершинам узла v
        for each u in Adj(v) \ H do
            // Путь через u короче текущего пути?
            if d[v] + w(v, u) < d[u] then
                d[u] = d[v] + w(v, u)
                PriorityQueueDecrease(Q, u, d[u])
                prev[u] = v
            end if
        end for
    end for
end function
```

Восстановление кратчайшего пути

```
function SearchShortestPath(G, src, dst)

    ShortestPath_Dijkstra(G, src, d, prev)

    // Восстановление пути из src в dst
    i = dst
    pathlen = 1
    while i != src do
        pathlen = pathlen + 1
        i = prev[i]
    end while

    j = 0
    i = dst
    while i != s do
        path[pathlen - j] = i
        i = prev[i]
        j = j + 1
    end while
    return path[], pathlen
end function
```

$$T = T_{Dijikstra} + O(|V|)$$

Разреженные и насыщенные графы

Вариант реализации алгоритма Дейкстры	**Насыщенный граф** $m = O(n^2)$	**Разреженный граф** $m = O(n)$
Вариант 1 D[i] – это массив (поиск за время $O(n)$)	$T = O(n^2 + m) = \boldsymbol{O(n^2)}$	$T = O(n^2 + m) = \boldsymbol{O(n^2)}$
Вариант 2 D[i] хранятся в бинарной куче	$T = O(n \log n + m \log n) = \boldsymbol{O(n^2 \log n)}$	$T = O(n \log n + m \log n) = \boldsymbol{O(n \log n)}$
Вариант 3 D[i] хранятся в Фибоначчиевой куче	$T = O(m + n \log n) = \boldsymbol{O(n^2)}$	$T = O(n + n \log n) = \boldsymbol{O(n \log n)}$

Кратчайшие пути в ациклических графах

Постановка задачи

- Рассмотрим задачу поиска кратчайших путей в ациклическом орграфе $G = (V, E)$.

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.

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,

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 $l(v) < l(w)$.

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 ($l(v_1) = 1$);

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;

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 ($l(v_2) = 2$);

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.

- Считаем, что $V = \{1, \ldots, n\}$ и $i < j$ для всех $(i, j) \in E$.

Рекуррентные 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 $j = 2, \ldots, n$.

Рекуррентные 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:

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

Рекуррентные 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 рекуррентной формулы:

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

- 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:

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

- 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:

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

Пути максимальной стоимости

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

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:

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

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:

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

- 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:

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

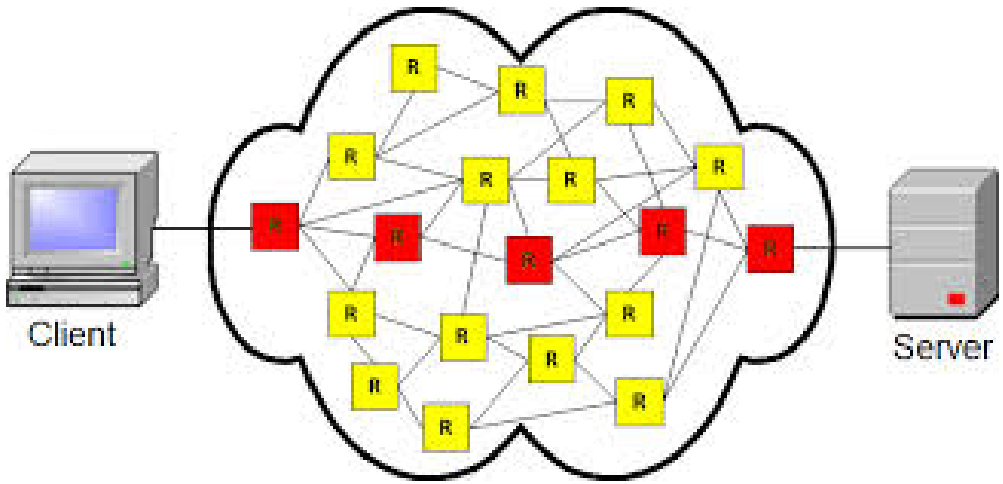
- Максимум по пустому множеству альтернатив равен $-\infty$.

Применение алгоритма Беллмана-Форда: протокол маршрутизатора

Протокол маршрутизации

Задача:

- Каждый узел обозначает маршрутезатор, а вес обозначает **время передачи** пакета от маршрутизатора i до j.
- Цель — разработать протокол для определения самого быстрого маршрута, когда маршрутизатор s хочет отправить пакет в узел t.



Протокол маршрутизации: алгоритм Дейкстры против алгоритма Беллмана-Форда

- Выбор: алгоритм Дейкстры.
- Однако для этого алгоритма необходимо **глобальное знание**, то есть знание всего графа, которое (почти) невозможно получить.
- Напротив, алгоритм Беллмана-Форда **требует только локальной информации**, то есть информации об **окрестностях вершин**, а не **всей сети .**

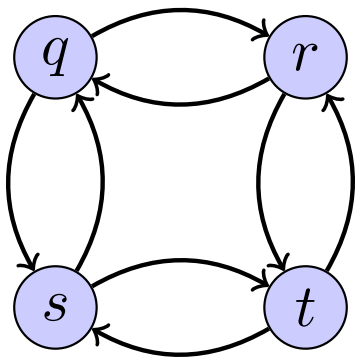
AsynchronousShortestPath(G, t)

1: Инициализация, положить $OPT[t,t] = 0$, и $OPT[v,t] = \infty$;
2: Отметить узел t как “активный”; //Узел v называется “активным” если $OPT[v,t]$ было изменено;
3: **while** существует активный узел **do**
4: Выберите активный узел w произвольно;
5: Удалите для узла w иметку — “активный”;
6: **for all** ребер (v,w) (в произвольном порядке) **do**
7: $OPT[v,t] = \min \begin{cases} OPT[v,t] \\ OPT[w,t] + d(v,w) \end{cases}$
8: **if** $OPT[v,t]$ было изменено **then**
9: Установить для v метку "активный";
10: **end if**
11: **end for**
12: **end while**

Задача: Длинейший путь

Задача: Длинейший путь

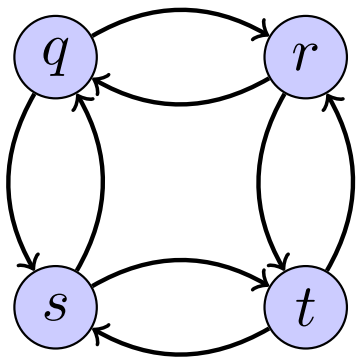
ВХОД:
Ориентированный граф $G =< V, E >$. Каждая дуга (u, v) имеет длину $d(u, v)$. Два узла: s и t.
ВЫХОД:
Самый длинный простой путь из s в t.



Сложность: задача LongestPath является NP-трудной. (Подсказка: очевидно, что задача LongestPath содержит HamiltonianPath в качестве своего особого случая.)

Секрет сложности задачи LongestPath I

- Разбиение на подзадачи: Подзадачи не являются независимыми.
- Рассмотрим задачу — найти путь от q до t. Разбьем ее на две подзадачи: найти путь от q до r и найти путь от r до t.



- Предположим, что мы уже решили подзадачи. Попробуем объединить решения двух подзадач:
  1. $P(q, r) = q \to s \to t \to r$
  2. $P(r, t) = r \to q \to s \to t$

  Мы получим путь $q \to s \to t \to r \to q \to s \to t$, который не является простым.

- Другими словами, использование s в первой подзадаче не позволяет нам использовать s во второй подзадаче. Однако мы не можем получить оптимальное решение второй подзадачи без использования s.

- Напротив, проблема ShortestPath не имеет этой трудности.
- Почему? Решения подзадач **не имеют общего узла**. Предположим, что кратчайшие пути $P(q, r)$ и $P(r, t)$ имеют общий узел $w (w \neq r)$. Тогда будет цикл $w \to \cdots \to r \to \cdots \to w$. Удаление этого цикла приводит к более короткому пути (без отрицательного цикла). Противоречие.
- Это означает, что две подзадачи являются независимыми: решение одной подзадачи не влияет на решение другой подзадачи.

Если все ребра имеют положительный вес, то существует жадный алгоритм, который находит оптимальное решение.

Об этом мы поговорим в следующих лекциях.