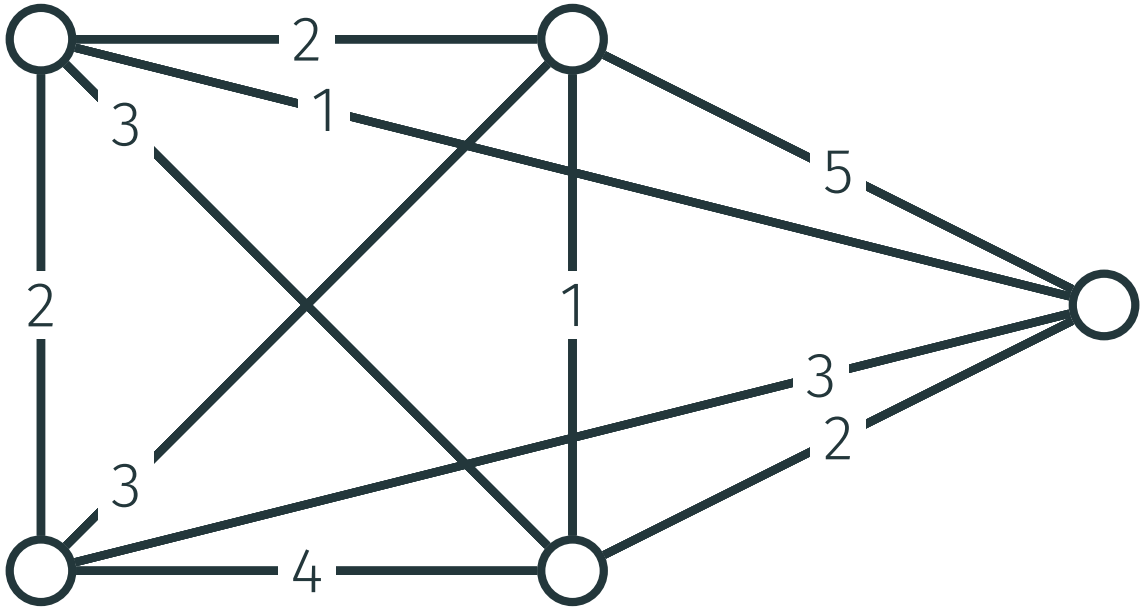
Алгоритмы для задачи коммивояжёра

Александр Куликов

https://compsciclub.ru/courses/tsp/2017-autumn/

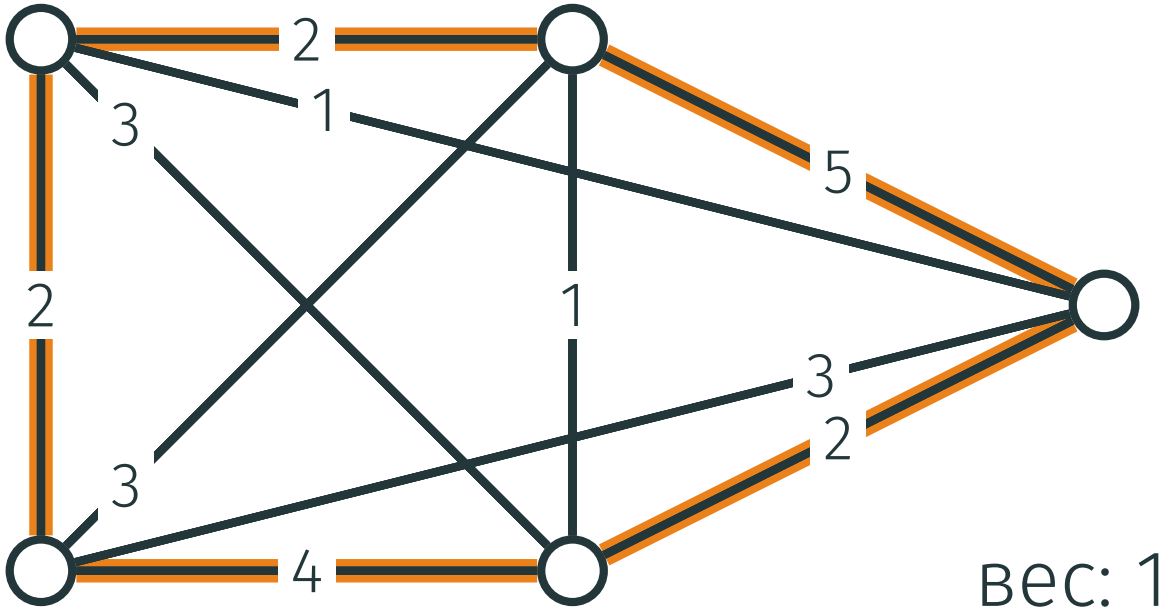
Задача коммивояжёра

Найти в данном полном ориентированном взвешенном графе цикл (или путь) минимального веса, проходящий по всем вершинам ровно по разу



Задача коммивояжёра

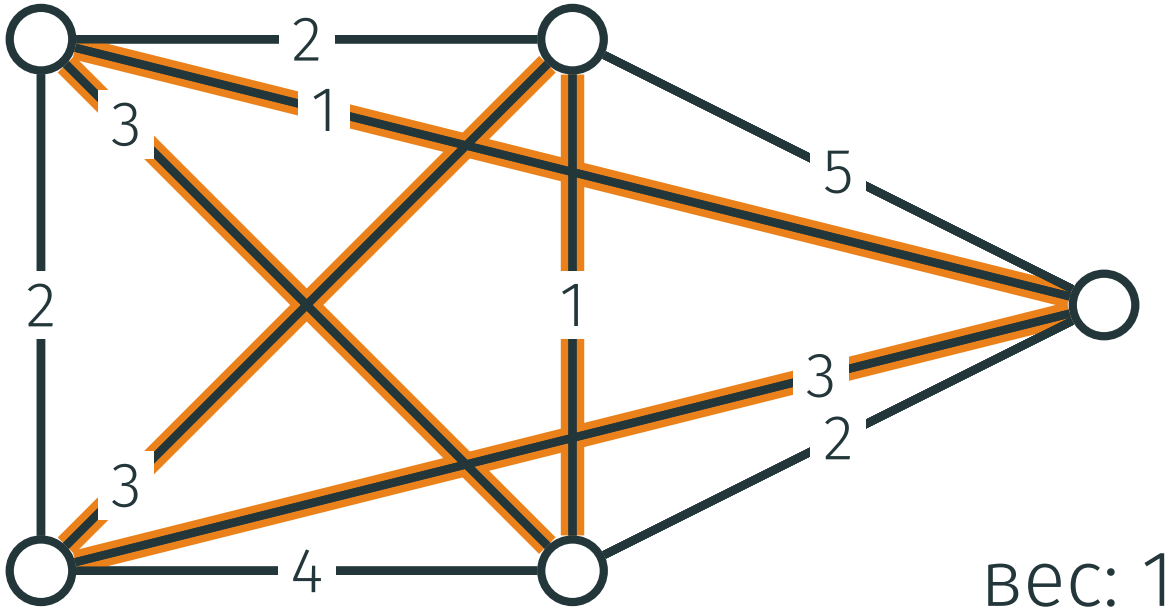
Найти в данном полном ориентированном взвешенном графе цикл (или путь) минимального веса, проходящий по всем вершинам ровно по разу



вес: 15

Задача коммивояжёра

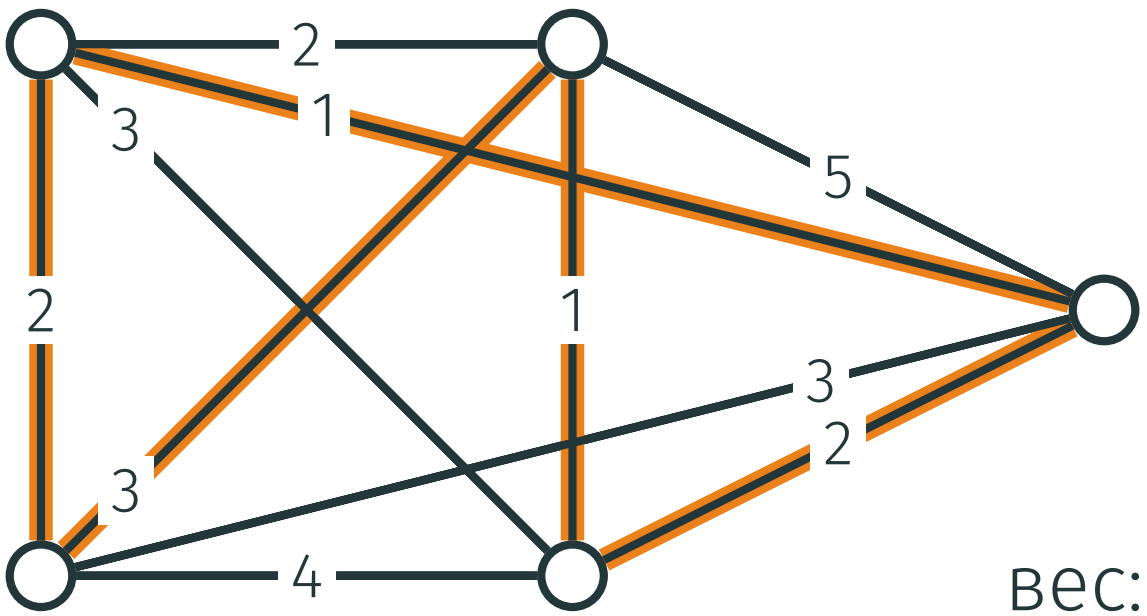
Найти в данном полном ориентированном взвешенном графе цикл (или путь) минимального веса, проходящий по всем вершинам ровно по разу



вес: 11

Задача коммивояжёра

Найти в данном полном ориентированном взвешенном графе цикл (или путь) минимального веса, проходящий по всем вершинам ровно по разу



вес: 9

Статус

- Классическая задача комбинаторной оптимизации с огромным количеством приложений

Статус

- Классическая задача комбинаторной оптимизации с огромным количеством приложений
- Полиномиальных алгоритмов до сих пор не придумано

Приложения

Доставка

Нужно посетить несколько точек. Как сделать это как можно быстрее?

SAP Transportation Management

Consolidate orders and maximize the return on your transportation spend – with our transportation management system (TMS). Accurately forecast demand and shipment volumes to fine-tune transportation planning. Enhance freight, fleet, and logistics management. And gain real-time visibility into global and domestic shipping across all transportation modes and industries.

Яндекс Маршрутизация

Для кого | Возможности | Для интернет-магазинов | Преимущества

Платформа для решения логистических задач в городской среде

Предсказуемая последняя миля.
Без пробок и перекрытий.
С точностью до минуты.

Оставить заявку

VeeRoute

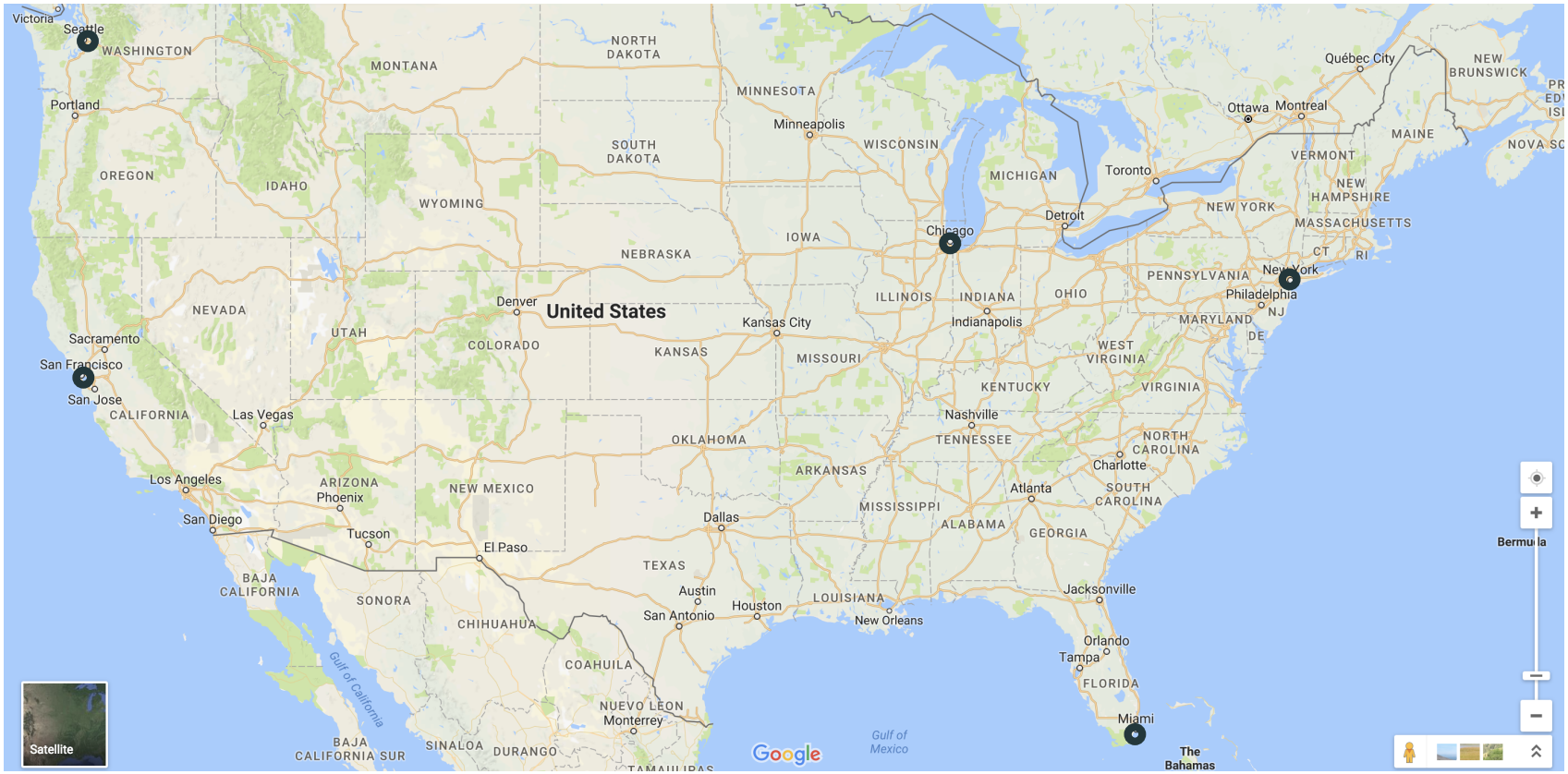
О компании | Продукты и услуги | Отрасли и решения | Истории успеха | Партнёры | Р

Платформа VeeRoute для оптимизации логистики

Платформа VeeRoute может решить различные задачи: начиная с доставки нефтепродуктов или продуктов питания оптимальным маршрутом до срочного планирования экстренных ремонтных работ в сервисных компаниях. Решение VeeRoute отличается своей гибкостью, исключительной технической оснащенностью и универсальностью.

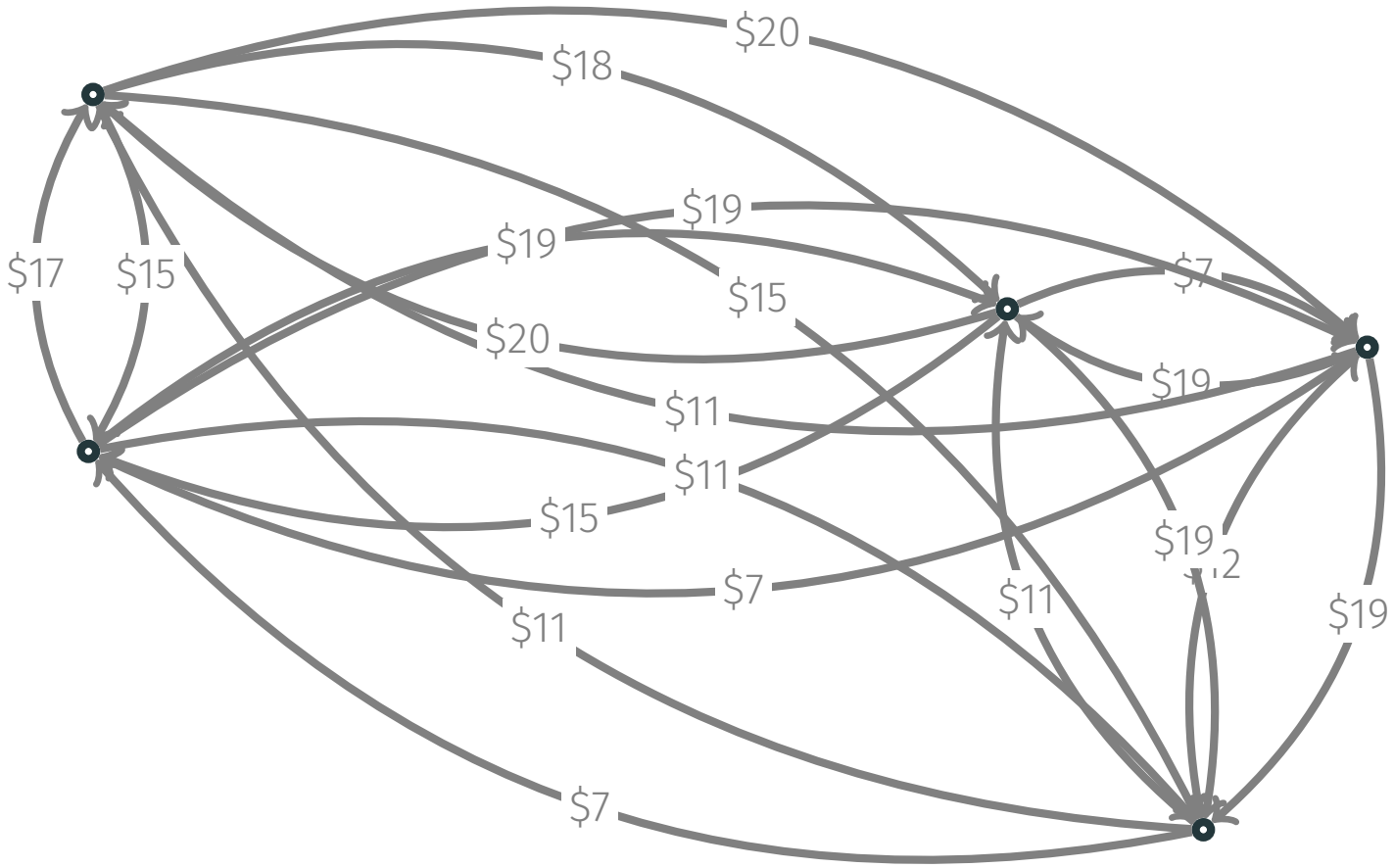
Путешествие

Путешествие

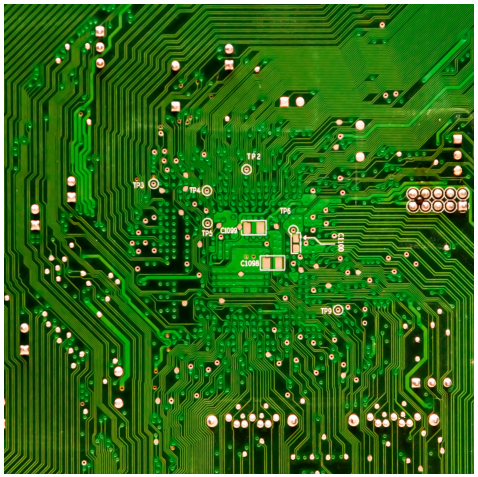


Путешествие

Путешествие

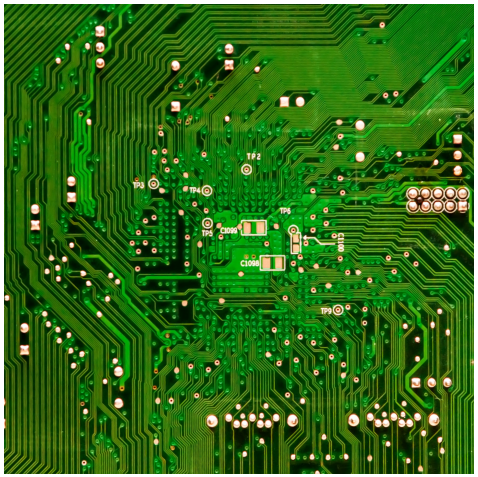


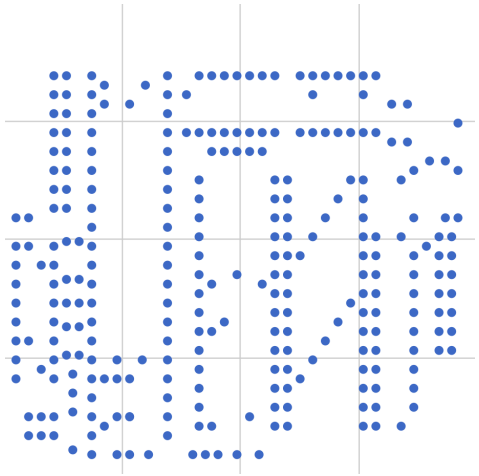
Плата



https://developers.google.com/optimization/routing/tsp/tsp

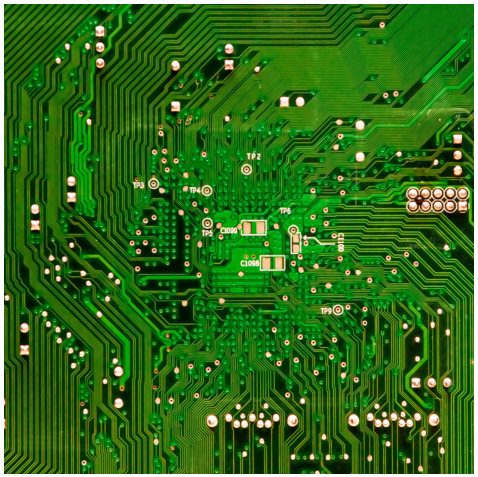
Плата

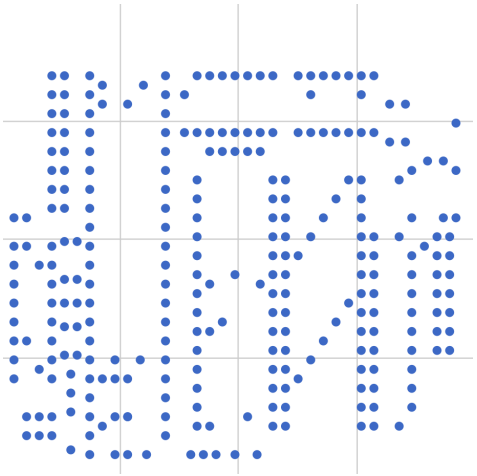


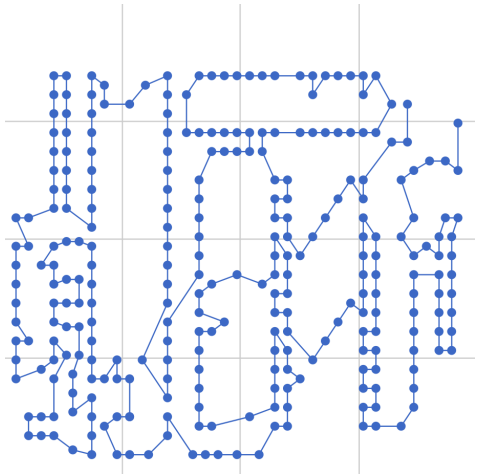


https://developers.google.com/optimization/routing/tsp/tsp

Плата







https://developers.google.com/optimization/routing/tsp/tsp

Евклидов коммивояжёр

- Евклидов коммивояжёр: вместо полного графа даны n точек на плоскости $p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n)$

Евклидов коммивояжёр

- Евклидов коммивояжёр: вместо полного графа даны n точек на плоскости $p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n)$
- Веса заданы неявно:

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

Евклидов коммивояжёр

- Евклидов коммивояжёр: вместо полного графа даны n точек на плоскости $p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n)$
- Веса заданы неявно:

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

- Веса симметричны: $d(p_i, p_j) = d(p_j, p_i)$

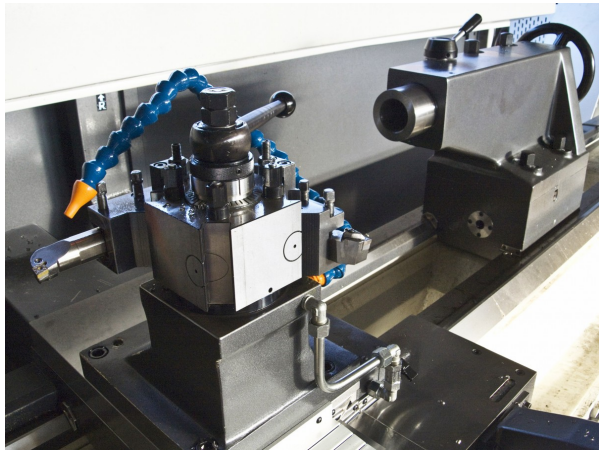
Евклидов коммивояжёр

- Евклидов коммивояжёр: вместо полного графа даны n точек на плоскости $p_1 = (x_1, y_1), \ldots, p_n = (x_n, y_n)$
- Веса заданы неявно:

$$d(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

- Веса симметричны: $d(p_i, p_j) = d(p_j, p_i)$
- И удовлетворяют неравенству треугольника: $d(p_i, p_j) \leq d(p_i, p_k) + d(p_k, p_j)$

Обработка деталей



Необходимо обработать n деталей на станке. Чтобы обработать после i-й детали j-ю, нужно потратить t_{ij} единиц времени на перенастройку станка. Каков оптимальнй порядок обработки всех деталей?

Кратчайшая общая надстрока

- Задача о кратчайшей общей надстроке: по данному множеству $\{s_1, \ldots, s_n\}$ из n строк найти самую короткую строку, содержащую все s_i как подстроки

Кратчайшая общая надстрока

- Задача о кратчайшей общей надстроке: по данному множеству $\{s_1, \ldots, s_n\}$ из n строк найти самую короткую строку, содержащую все s_i как подстроки
- Приложение: хранение и сжатие данных, сборка генома

Кратчайшая общая надстрока

- Задача о кратчайшей общей надстроке: по данному множеству $\{s_1, \ldots, s_n\}$ из n строк найти самую короткую строку, содержащую все s_i как подстроки
- Приложение: хранение и сжатие данных, сборка генома
- На первый взгляд, непонятно, какое эта задача имеет отношение к задаче коммивояжёра

Надстрока: пример

- Рассмотрим такой вход:

  ABE, DFA, DAB, CBD, ECA, ACB

Надстрока: пример

- Рассмотрим такой вход:

  ABE, DFA, DAB, CBD, ECA, ACB

- Наивный способ получить надстроку — склеить строки:

  ABEDFADABCBDECAACB

Надстрока: пример

- Рассмотрим такой вход:

  ABE, DFA, DAB, CBD, ECA, ACB

- Наивный способ получить надстроку — склеить строки:

  ABEDFADABCBDECAACB

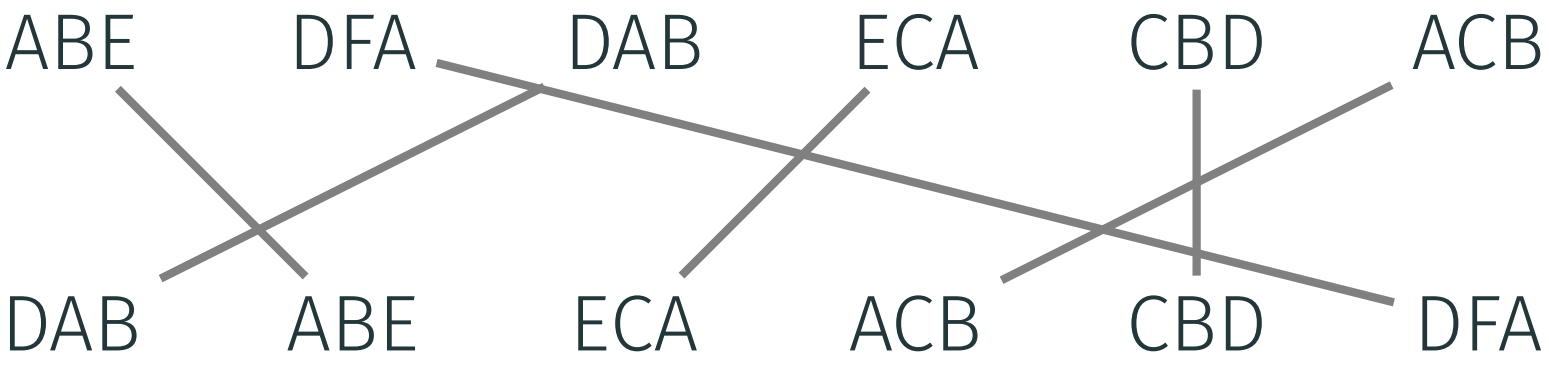
- Однако строки ECA и ACB имеют непустое перекрытие. Имеет смысл наложить их друг на друга:

  ECACB

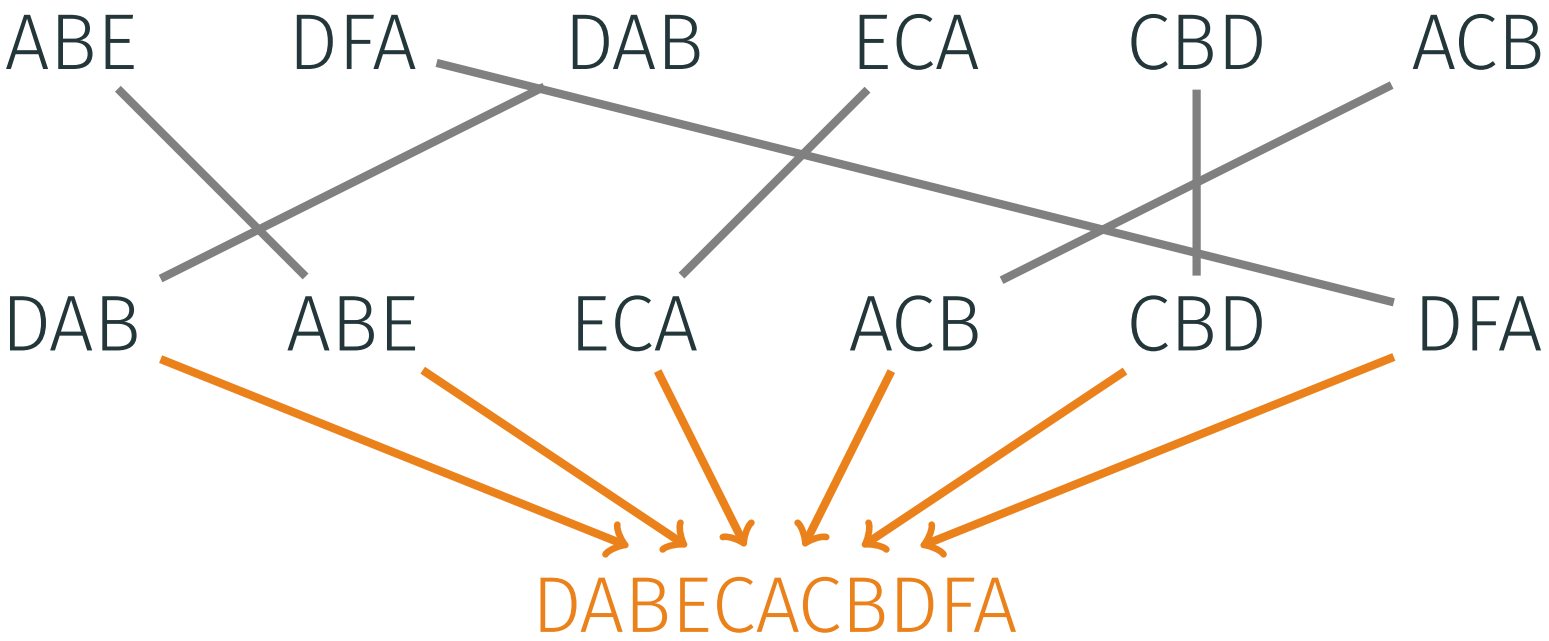
Надстрока: перестановочная задача

ABE DFA DAB ECA CBD ACB

Надстрока: перестановочная задача



Надстрока: перестановочная задача

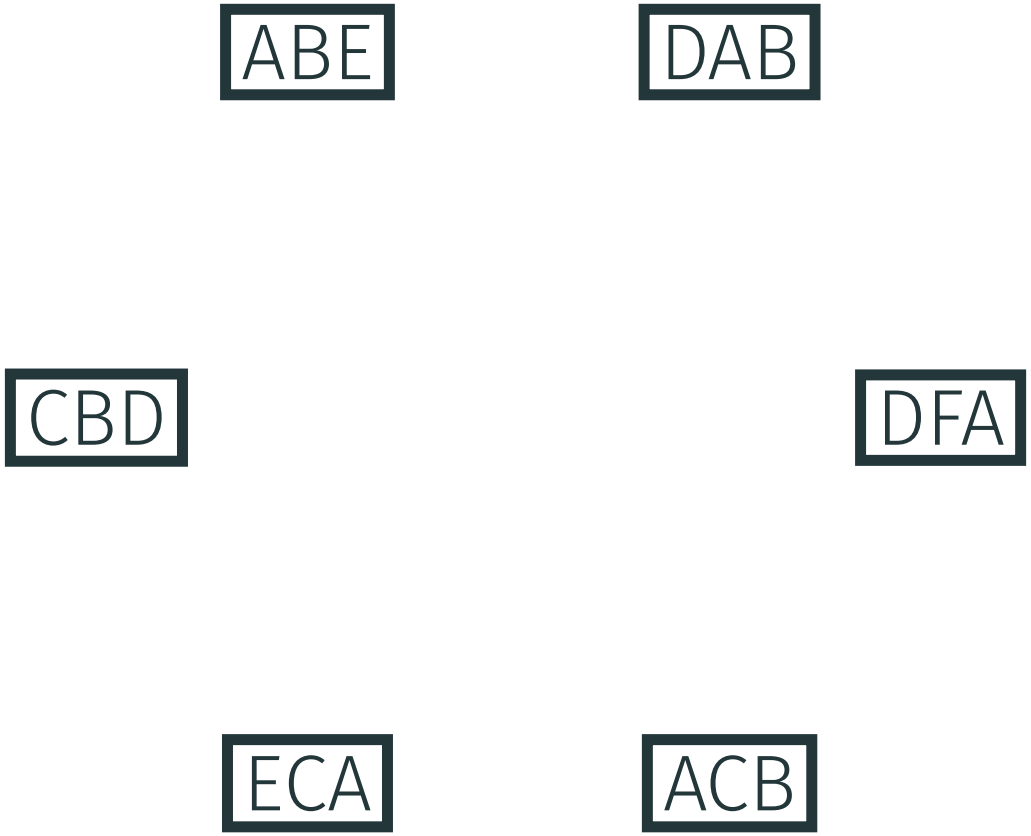


Граф перекрытий: SCS→MAX-ATSP

ABE DFA DAB CBD ECA ACB

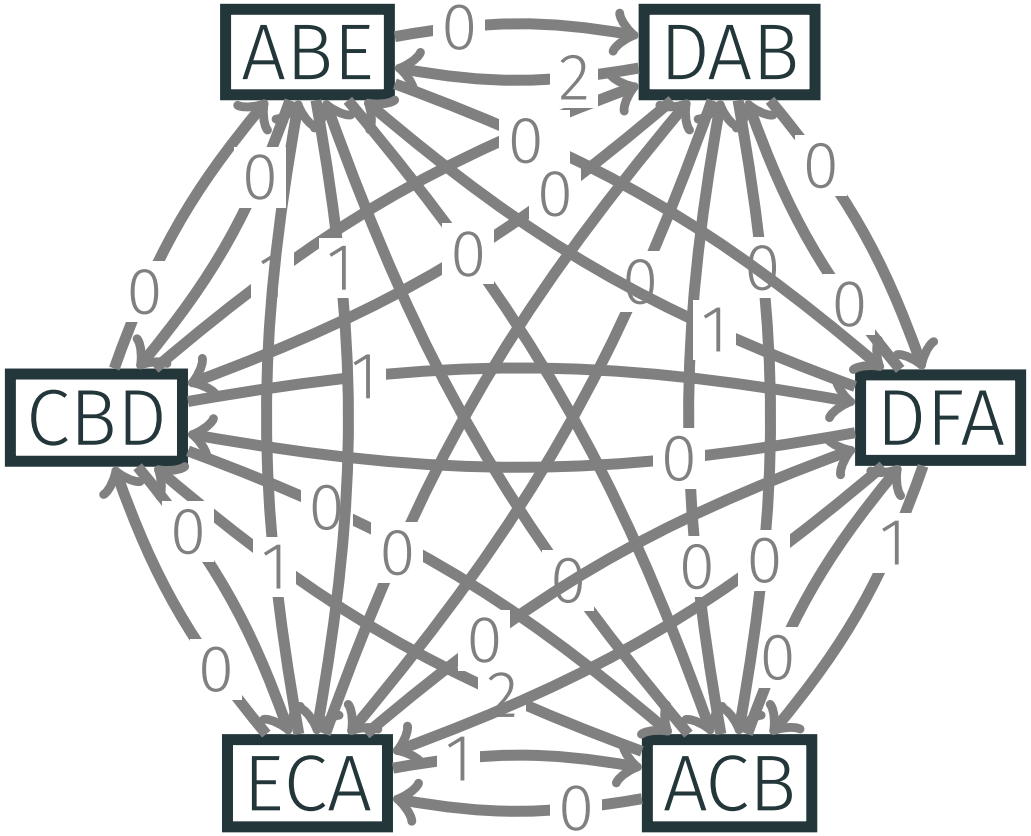
Граф перекрытий: SCS→MAX-ATSP

ABE DFA DAB CBD ECA ACB



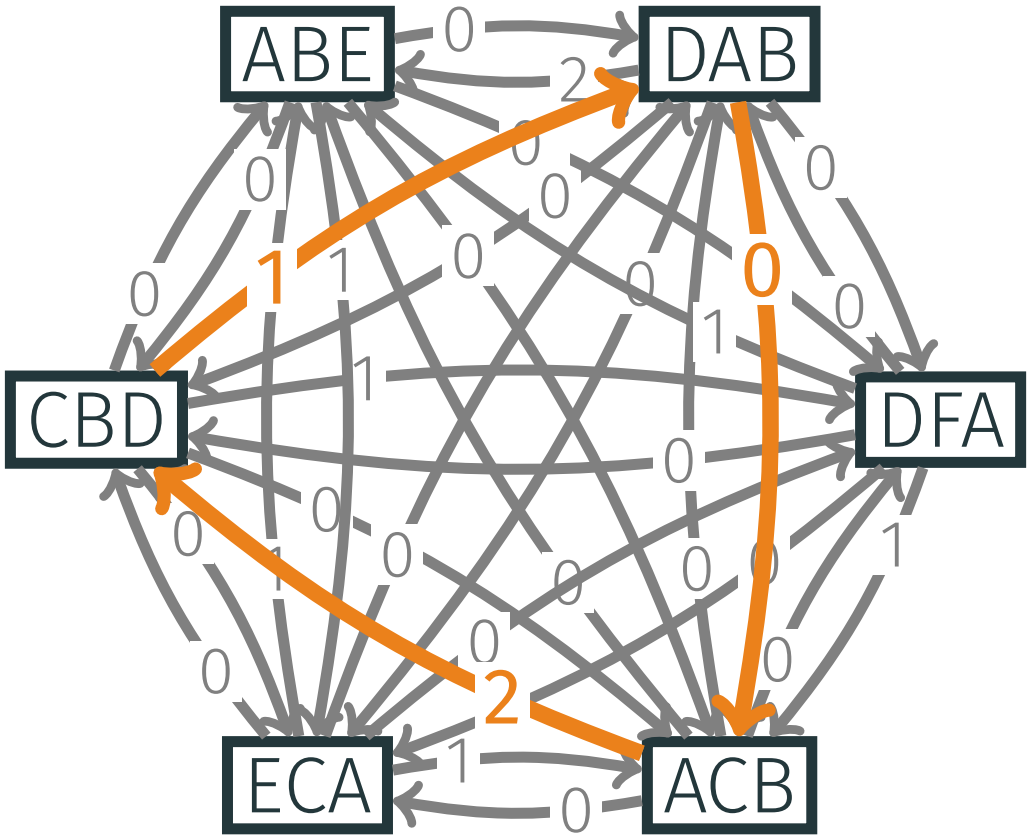
Граф перекрытий: SCS→MAX-ATSP

ABE DFA DAB CBD ECA ACB



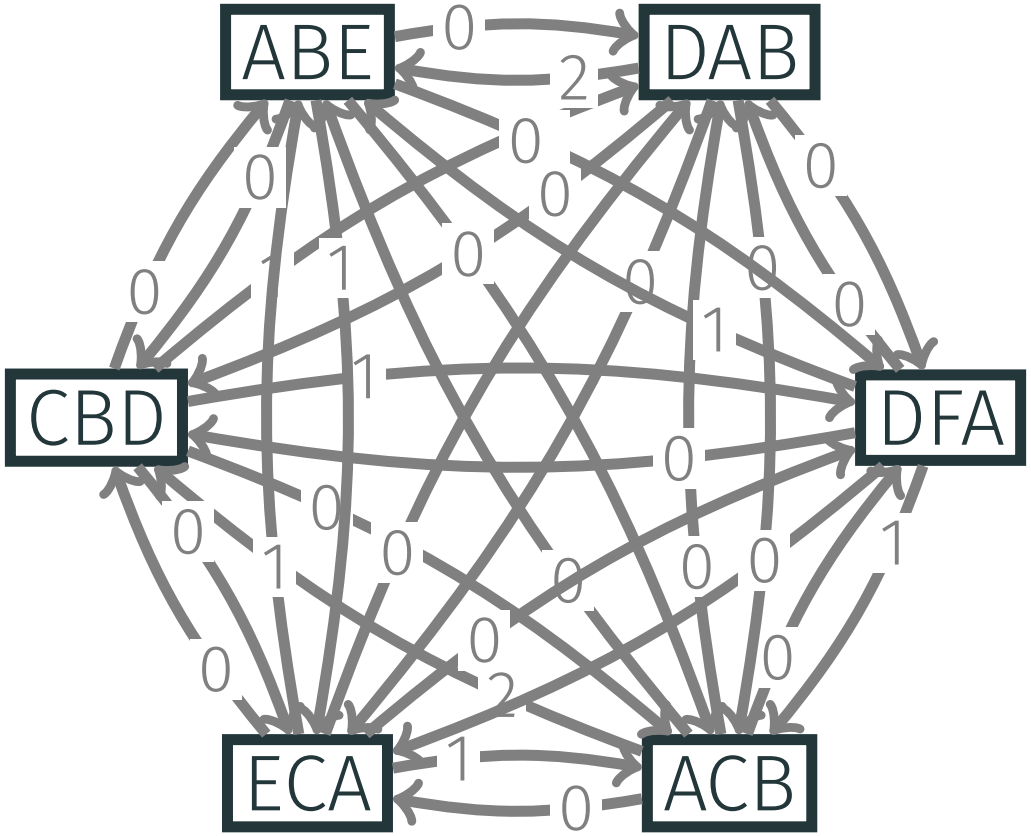
Граф перекрытий: SCS→MAX-ATSP

ABE DFA DAB CBD ECA ACB



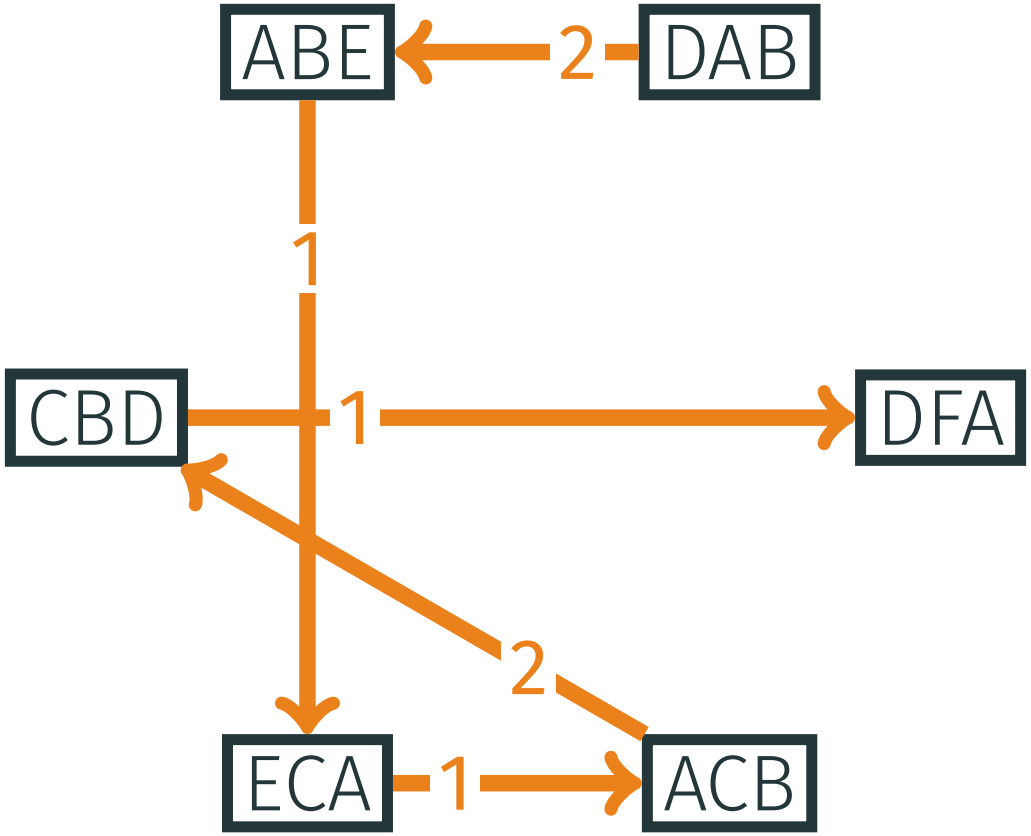
Граф перекрытий: SCS→MAX-ATSP

ABE DFA DAB CBD ECA ACB



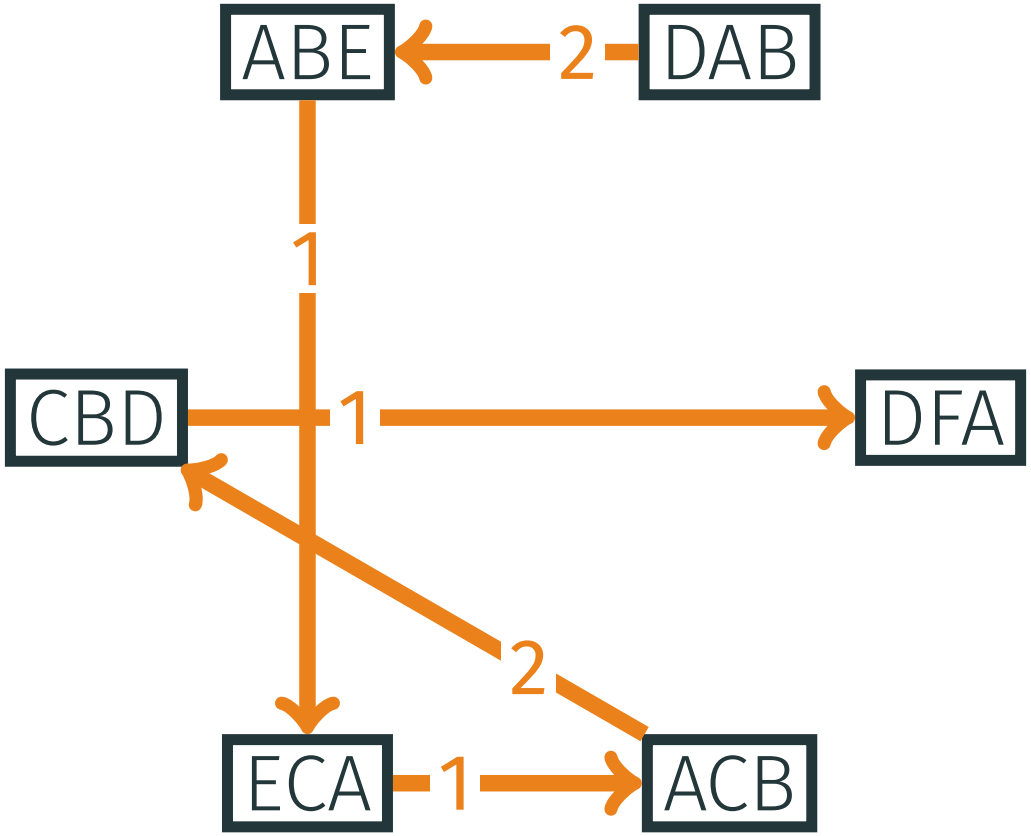
Граф перекрытий: SCS→MAX-ATSP

ABE DFA DAB CBD ECA ACB



Граф перекрытий: SCS→MAX-ATSP

ABE DFA DAB CBD ECA ACB



DABECACBDFA

Эвристические методы

Эвристические методы

Полный перебор

Перебор перестановок

- Найти оптимальную перестановку легко: переберём все перестановки и выберем лучшую

Перебор перестановок

- Найти оптимальную перестановку легко: переберём все перестановки и выберем лучшую
- Но количество перестановок n объектов равно $n!$

$n!$: скорость роста

n	$n!$
5	120
8	40320
10	3628800
13	6227020800
20	2432902008176640000
30	265252859812191058636308480000000

Случайная перестановка

- Перебирать все перестановки слишком долго

Случайная перестановка

- Перебирать все перестановки слишком долго
- Что если просто сгенерировать случайную перестановку?

Случайная перестановка

- Перебирать все перестановки слишком долго
- Что если просто сгенерировать случайную перестановку?
- Её вес может оказаться сильно хуже веса оптимальной перестановки, даже для евклидова коммивояжёра

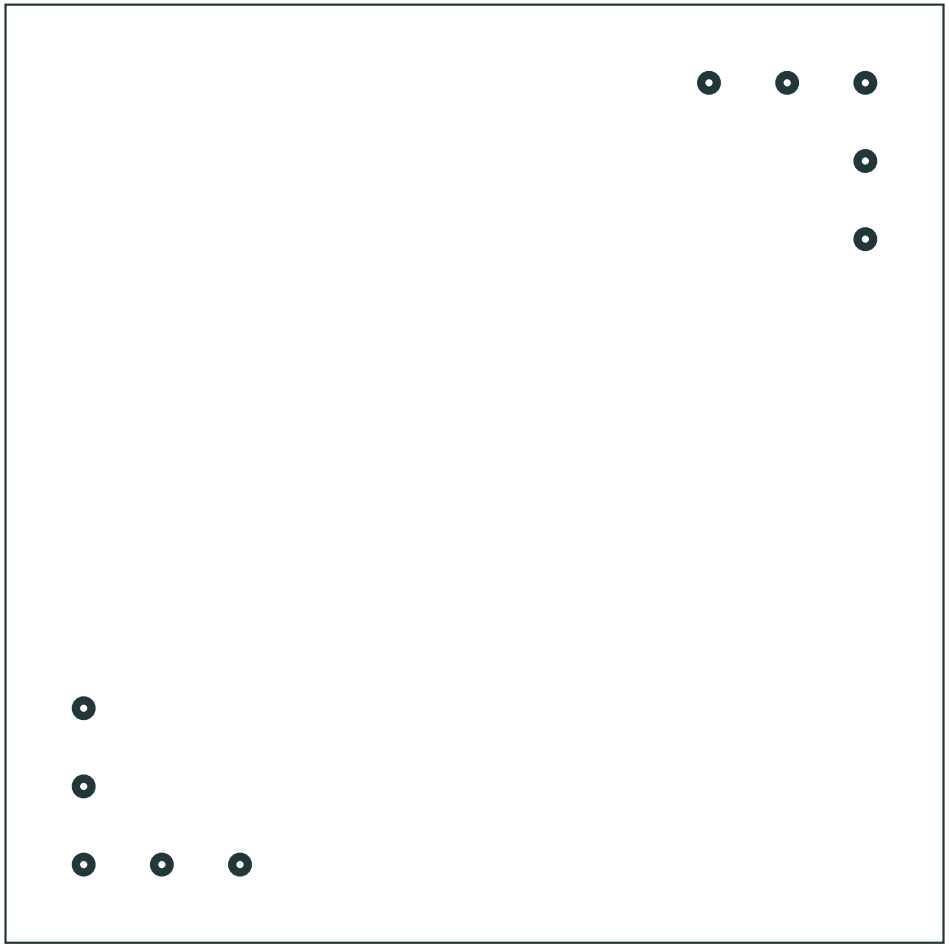
Средний вес

Лемма

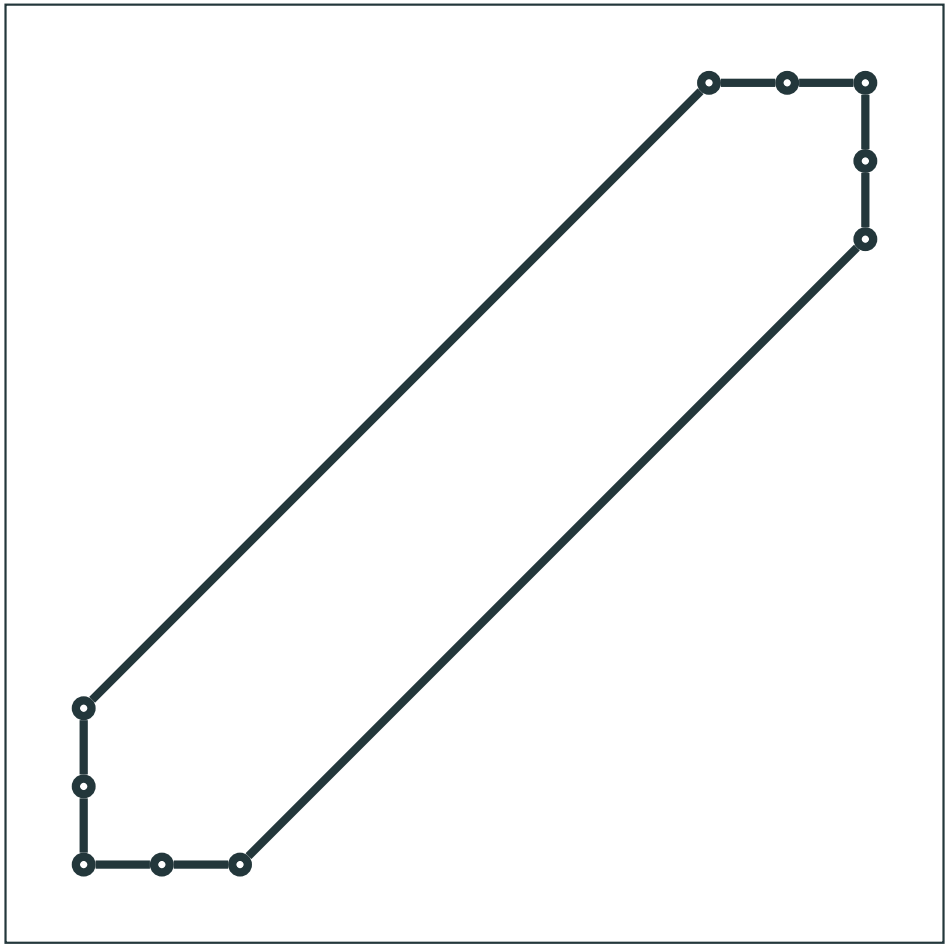
Для полного ориентированного графа G средний вес цикла, соответствующего случайной перестановке, равен

$$\frac{1}{n-1} \cdot \sum_{u,v \in V(G)} w(u,v)$$

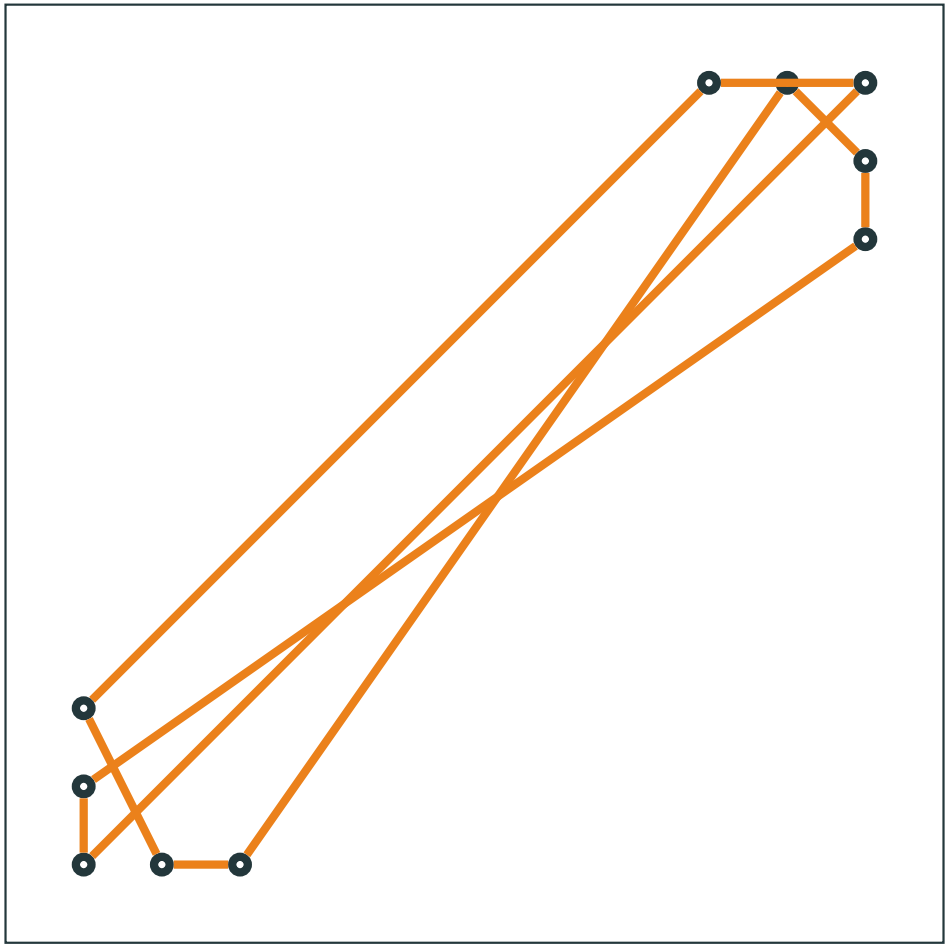
Плохой случай



Плохой случай



Плохой случай



Эвристические методы

Ближайший сосед

Ближайший сосед

- Как насчёт того, чтобы строить цикл постепенно, каждый раз идя в ближайшего соседа (в котором ещё не были)?

Ближайший сосед

- Как насчёт того, чтобы строить цикл постепенно, каждый раз идя в ближайшего соседа (в котором ещё не были)?
- Эффективно, на практике даёт неплохие результаты

Ближайший сосед

- Как насчёт того, чтобы строить цикл постепенно, каждый раз идя в ближайшего соседа (в котором ещё не были)?
- Эффективно, на практике даёт неплохие результаты
- Для общих графов может вернуть цикл, которой сильно хуже оптимального

Ближайший сосед

- Как насчёт того, чтобы строить цикл постепенно, каждый раз идя в ближайшего соседа (в котором ещё не были)?
- Эффективно, на практике даёт неплохие результаты
- Для общих графов может вернуть цикл, которой сильно хуже оптимального
- Для евклидовых графов может найти цикл, который в $\log n$ раз хуже оптимального

Ближайший сосед: плохой случай

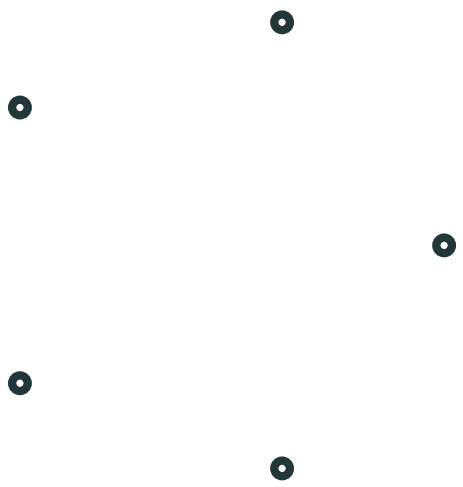
- Как заставить данную эвристику выдать плохой результат?

Ближайший сосед: плохой случай

- Как заставить данную эвристику выдать плохой результат?
- Пусть почти все веса равны 2

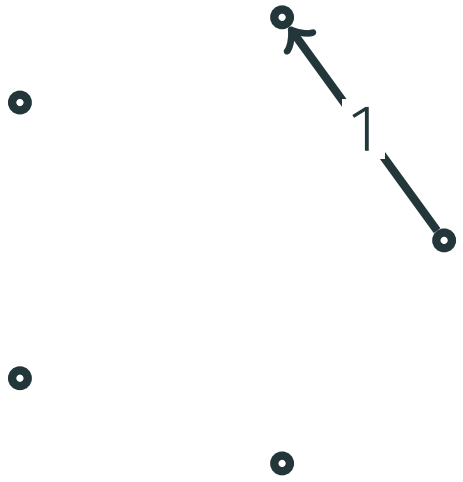
Ближайший сосед: плохой случай

- Как заставить данную эвристику выдать плохой результат?
- Пусть почти все веса равны 2
- И мы начинаем строить цикл:



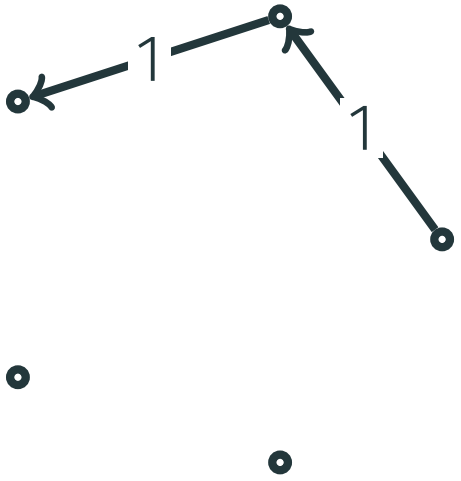
Ближайший сосед: плохой случай

- Как заставить данную эвристику выдать плохой результат?
- Пусть почти все веса равны 2
- И мы начинаем строить цикл:



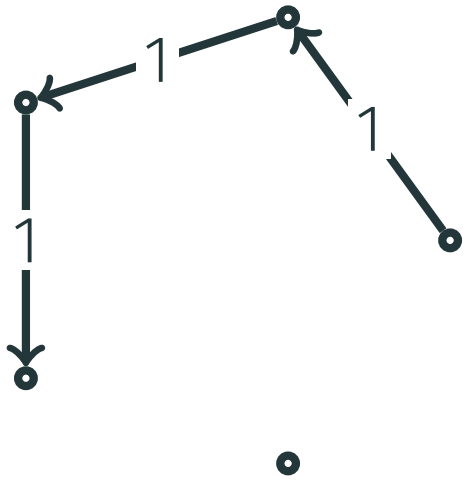
Ближайший сосед: плохой случай

- Как заставить данную эвристику выдать плохой результат?
- Пусть почти все веса равны 2
- И мы начинаем строить цикл:



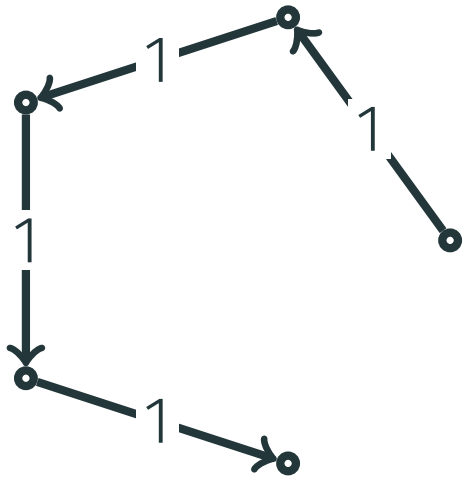
Ближайший сосед: плохой случай

- Как заставить данную эвристику выдать плохой результат?
- Пусть почти все веса равны 2
- И мы начинаем строить цикл:



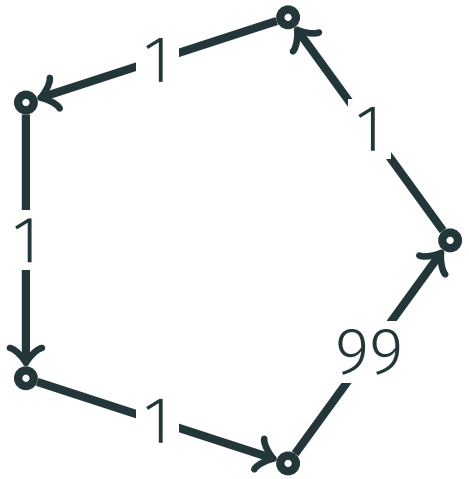
Ближайший сосед: плохой случай

- Как заставить данную эвристику выдать плохой результат?
- Пусть почти все веса равны 2
- И мы начинаем строить цикл:

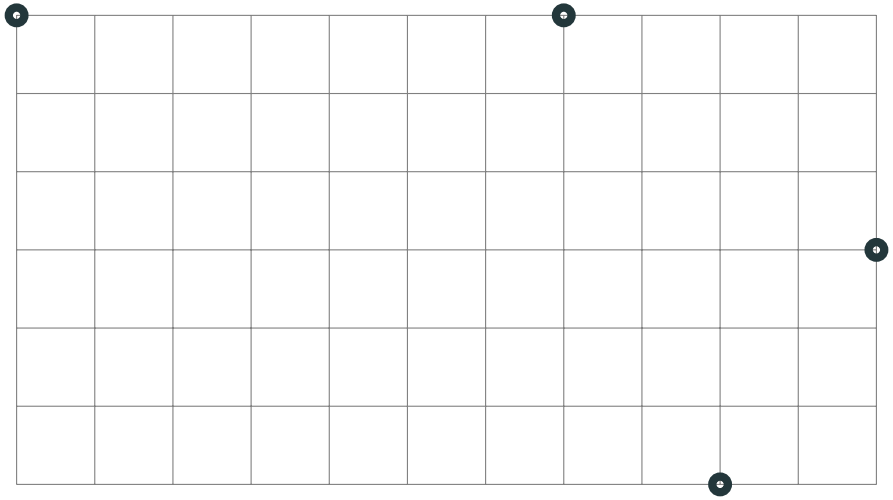


Ближайший сосед: плохой случай

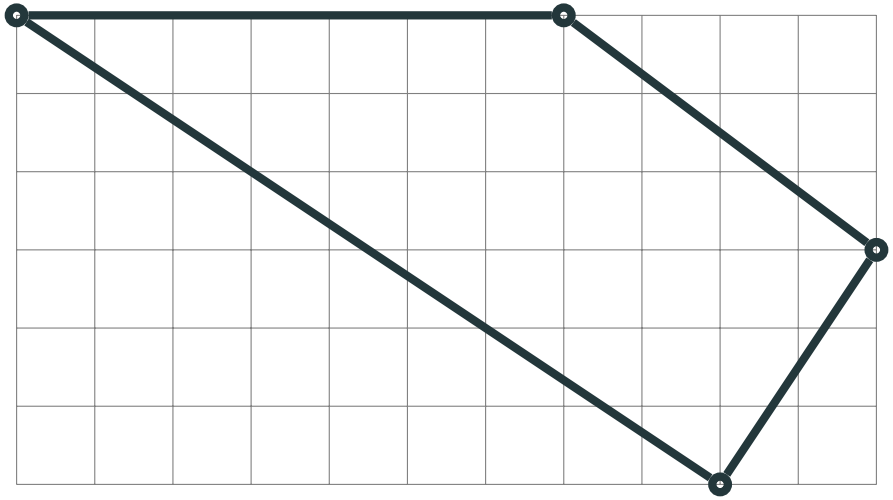
- Как заставить данную эвристику выдать плохой результат?
- Пусть почти все веса равны 2
- И мы начинаем строить цикл:



Субоптимальное решение для евклидова коммивояжёра

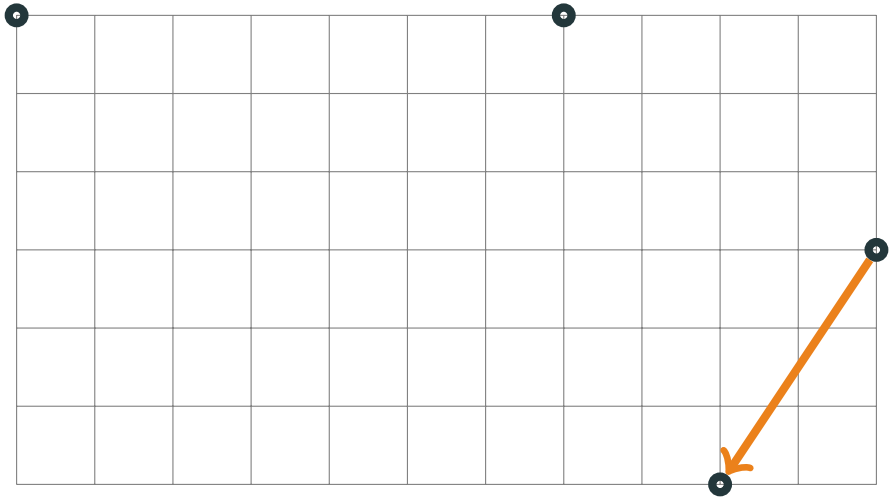


Субоптимальное решение для евклидова коммивояжёра



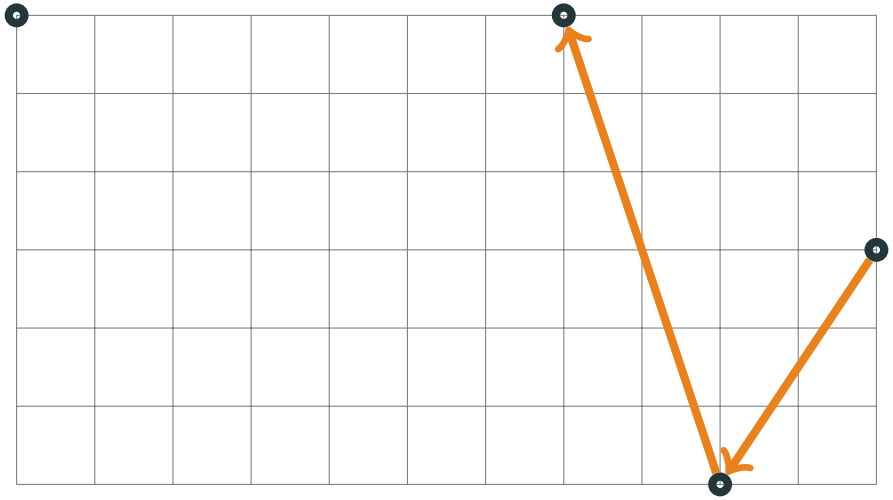
$\mathrm{OPT} \approx 26.42$

Субоптимальное решение для евклидова коммивояжёра



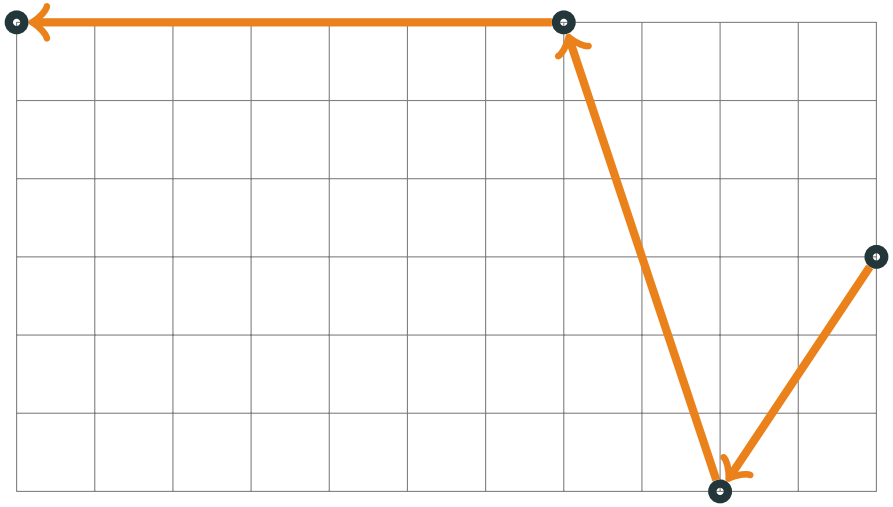
$\mathrm{OPT} \approx 26.42$

Субоптимальное решение для евклидова коммивояжёра



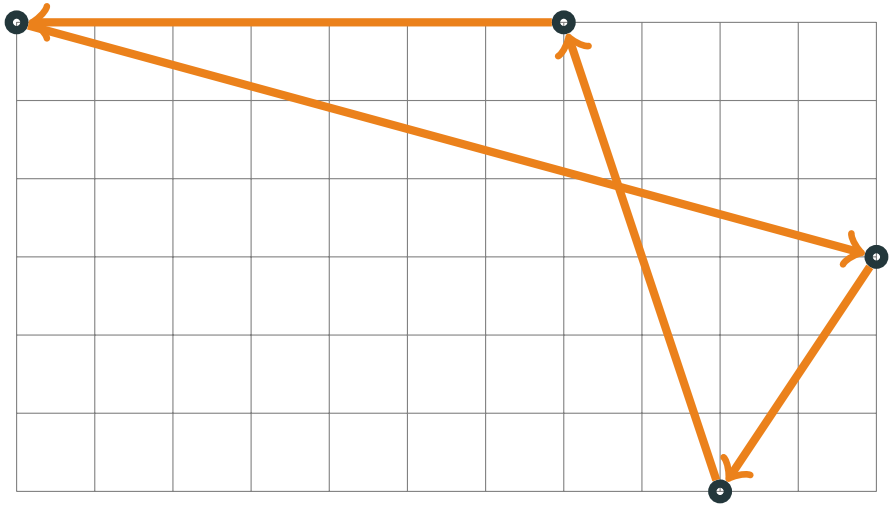
$\mathrm{OPT} \approx 26.42$

Субоптимальное решение для евклидова коммивояжёра



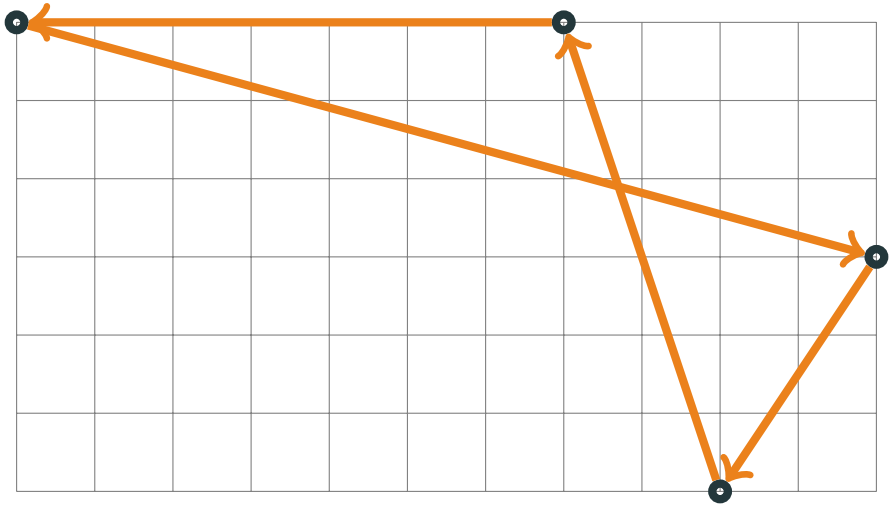
$\mathrm{OPT} \approx 26.42$

Субоптимальное решение для евклидова коммивояжёра



$\mathrm{OPT} \approx 26.42$

Субоптимальное решение для евклидова коммивояжёра



OPT $\approx$ 26.42

NN $\approx$ 28.33

Эвристические методы

Метод ветвей и границ

Основные идеи

- Начнём с какой-нибудь вершины

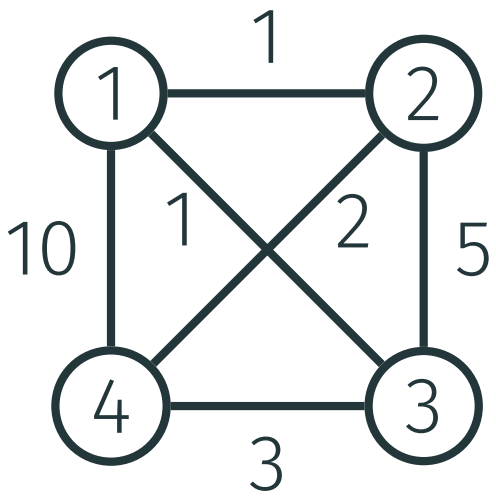
Основные идеи

- Начнём с какой-нибудь вершины
- На каждой итерации рекурсивно дописываем к текущему пути ещё одну вершину, в которую мы до этого не заезжали

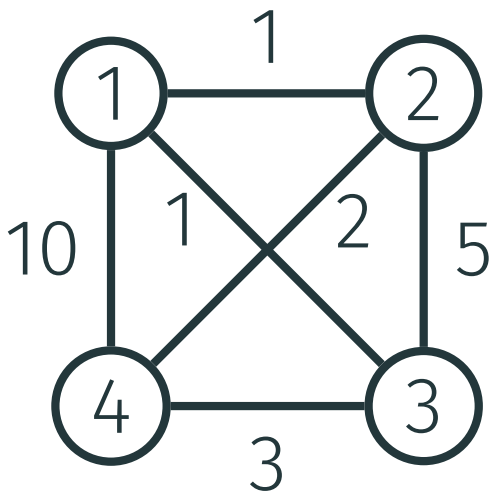
Основные идеи

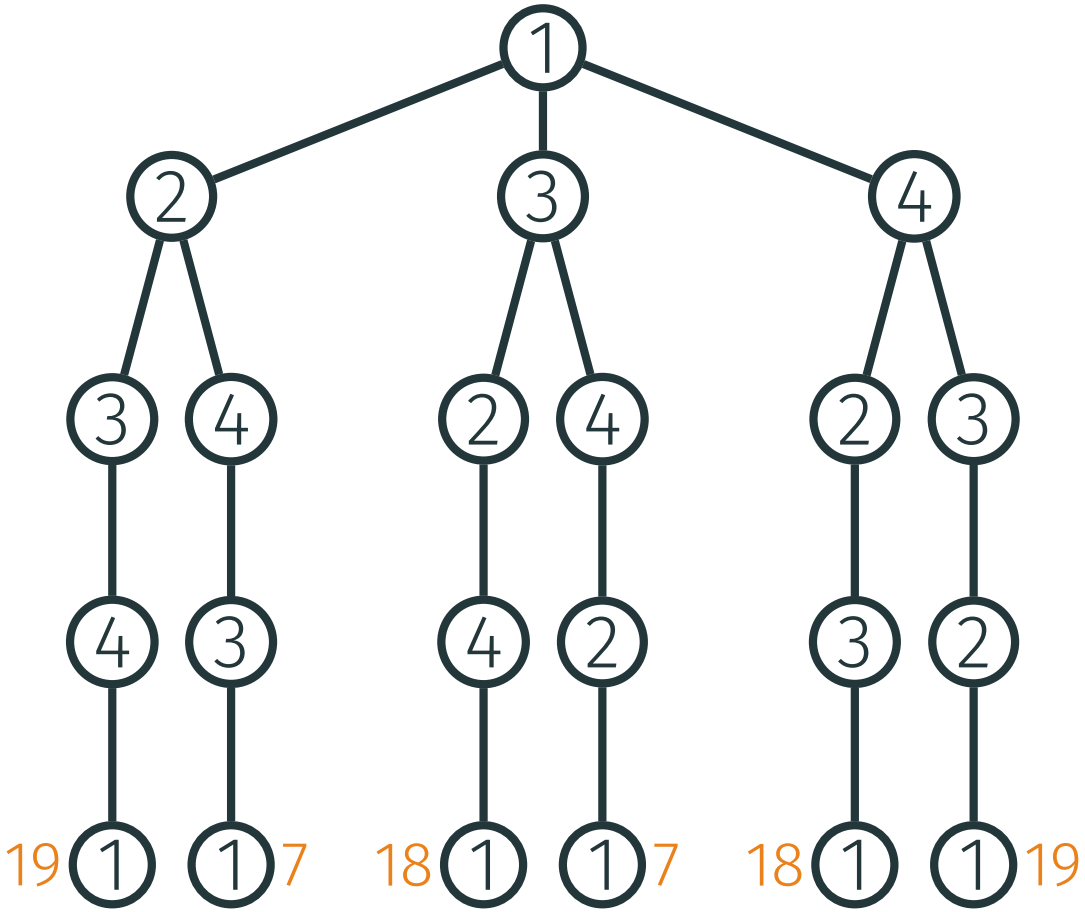
- Начнём с какой-нибудь вершины
- На каждой итерации рекурсивно дописываем к текущему пути ещё одну вершину, в которую мы до этого не заезжали
- Но как только становится ясно, что расширение текущего пути ни к чему хорошему не приведёт, перестаём пытаться его расширить

Пример: полный перебор

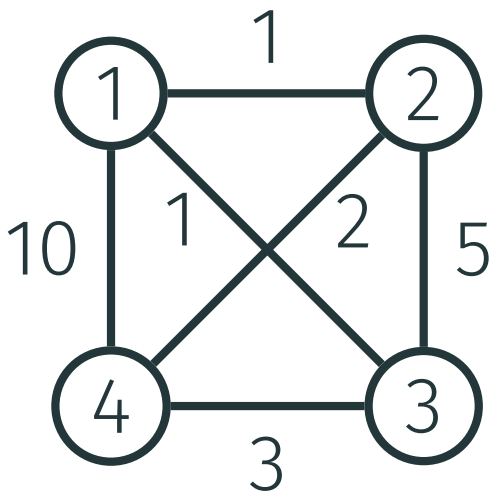


Пример: полный перебор

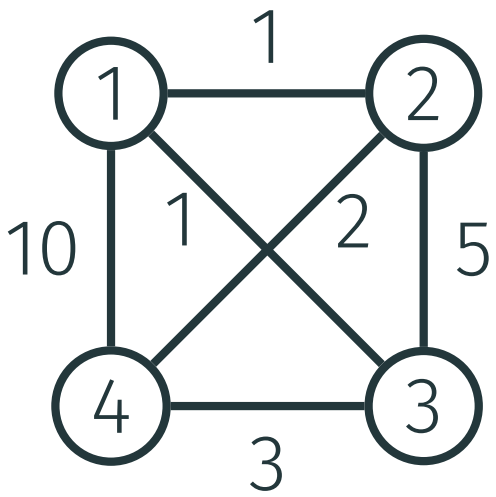


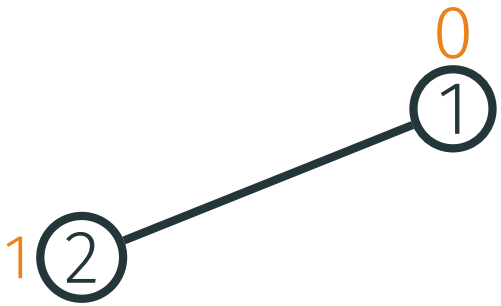


Пример: метод ветвей и границ

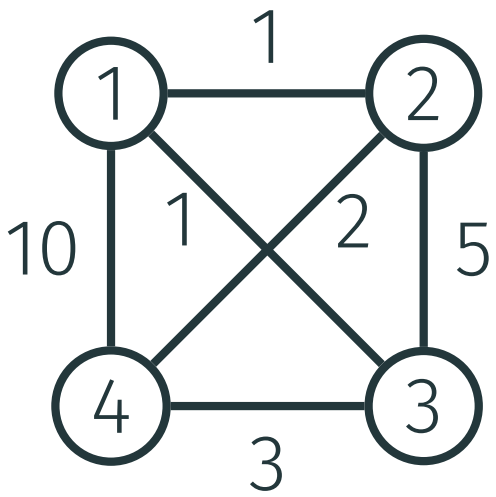


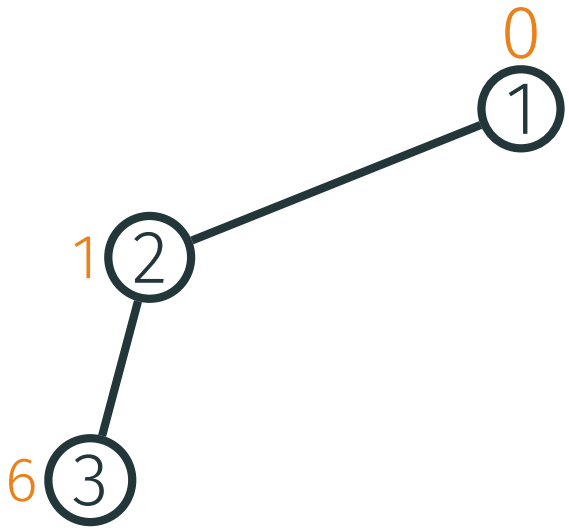
Пример: метод ветвей и границ



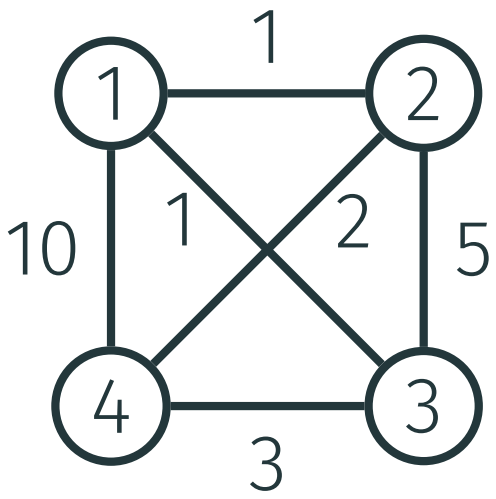


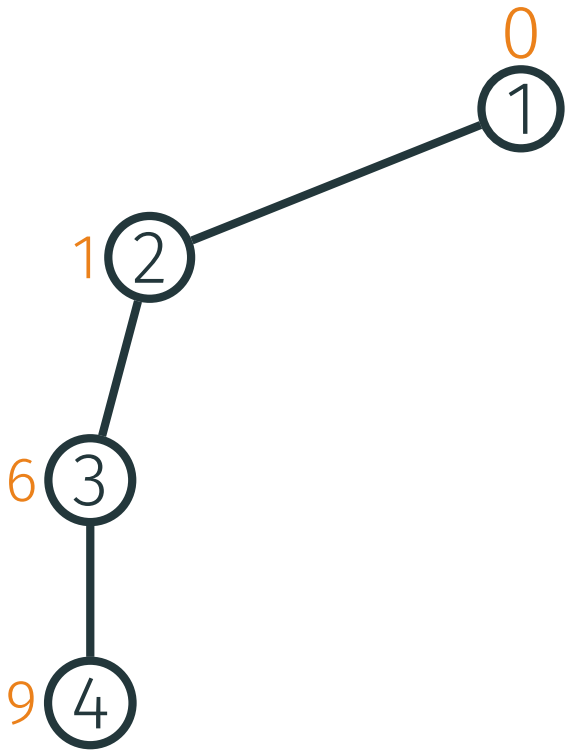
Пример: метод ветвей и границ



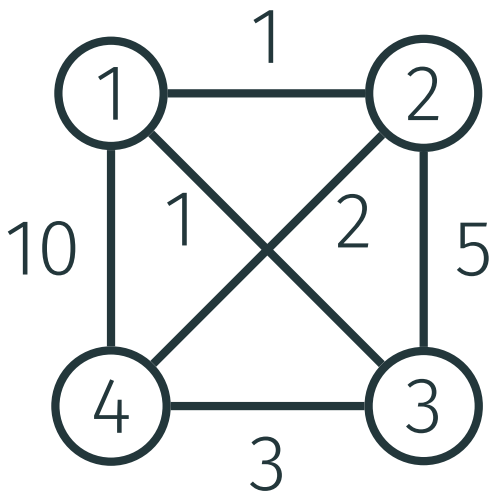


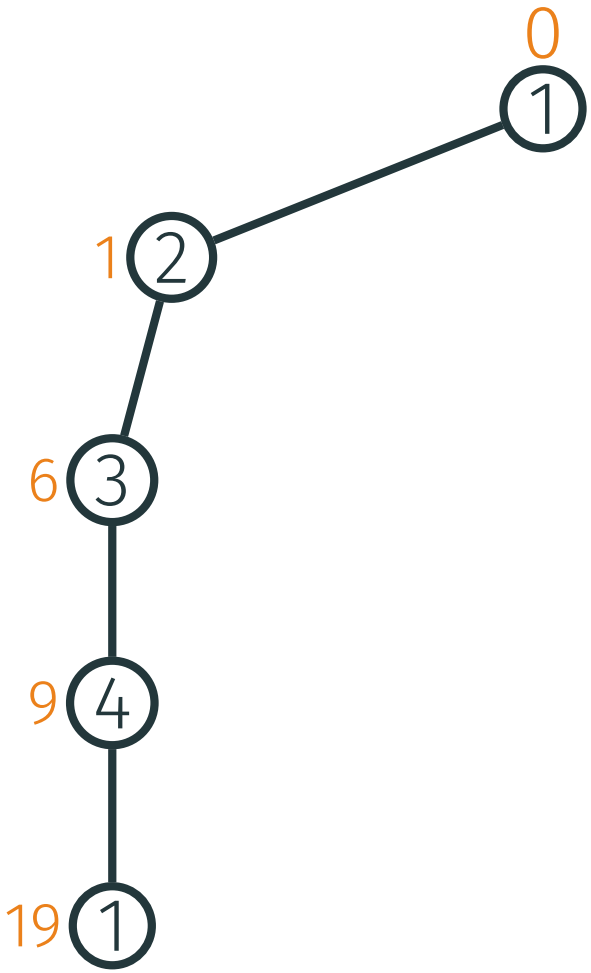
Пример: метод ветвей и границ





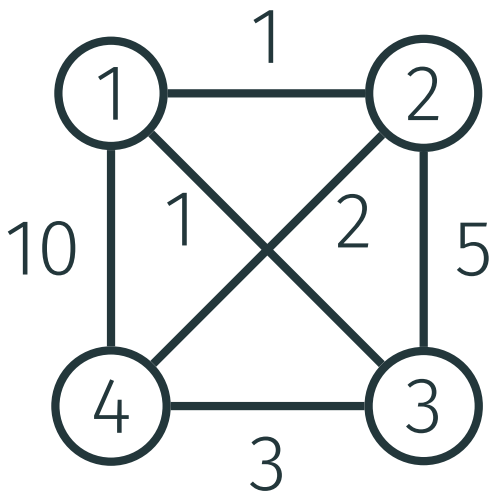
Пример: метод ветвей и границ

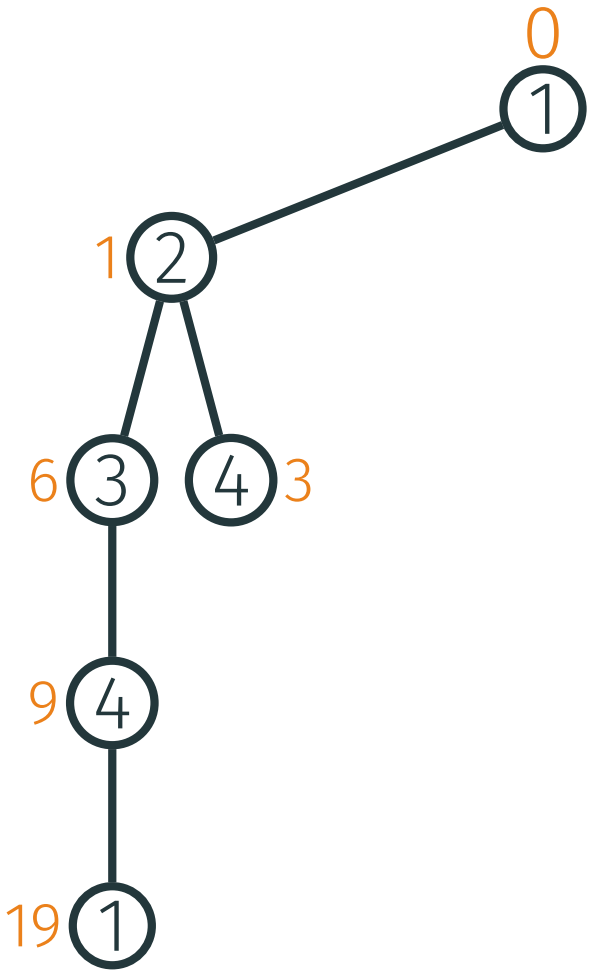




best so far: 19

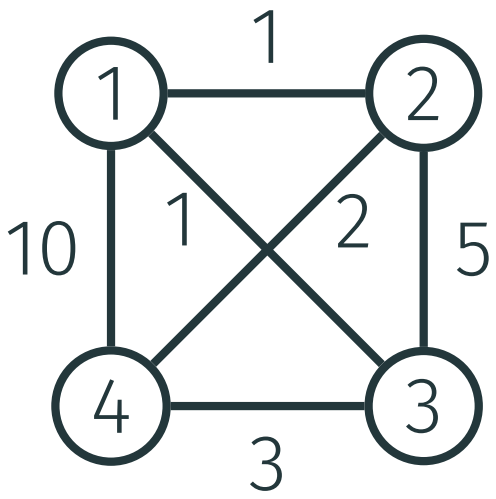
Пример: метод ветвей и границ

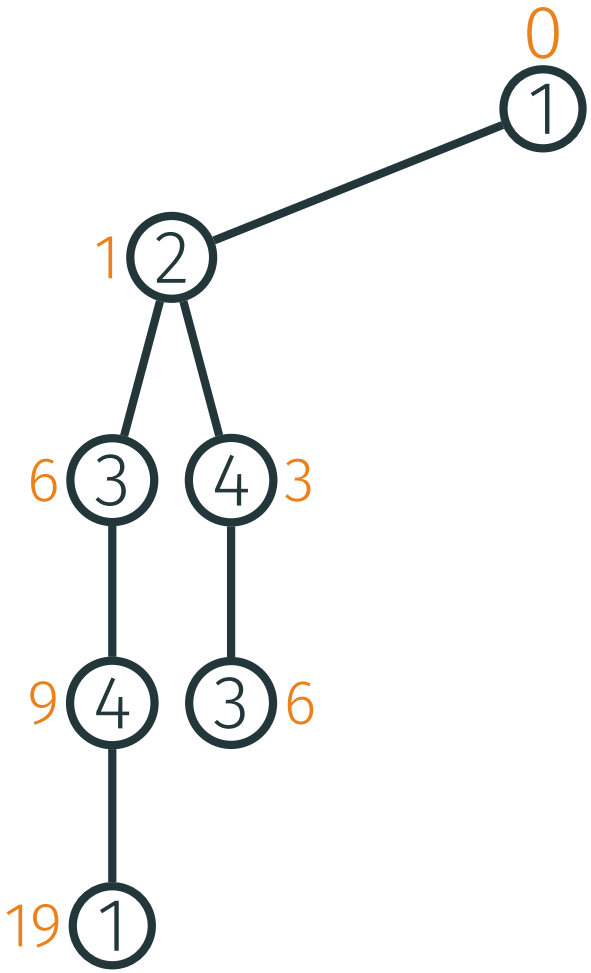




best so far: 19

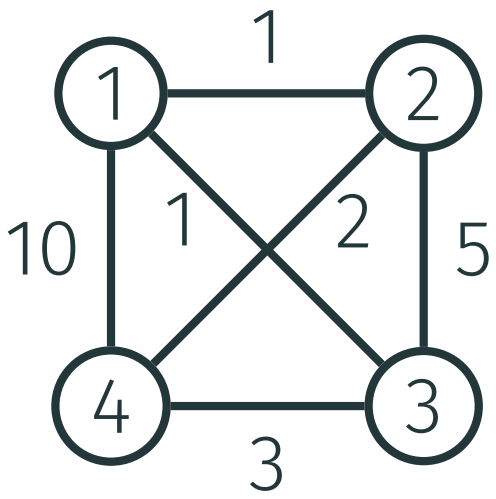
Пример: метод ветвей и границ

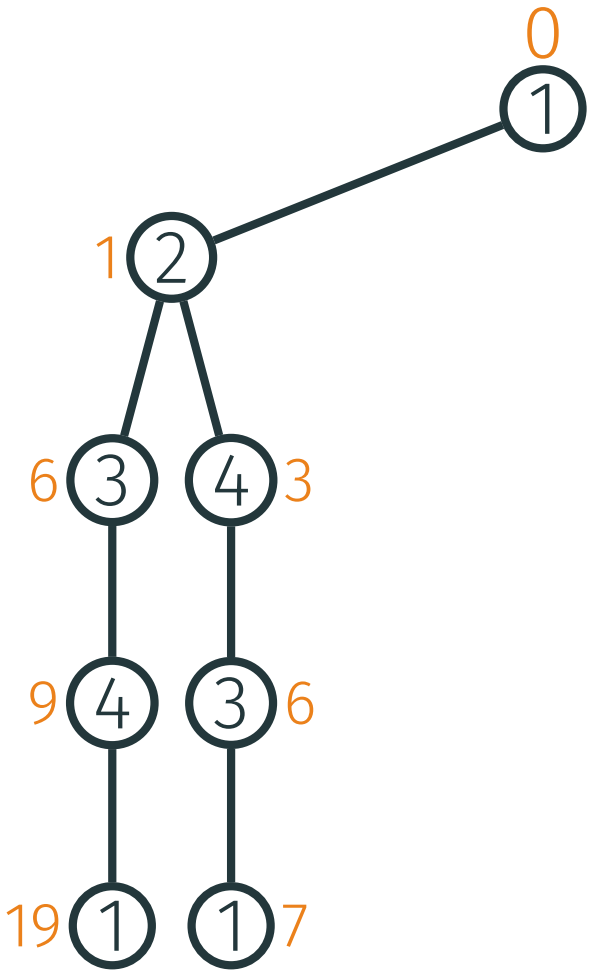




best so far: 19

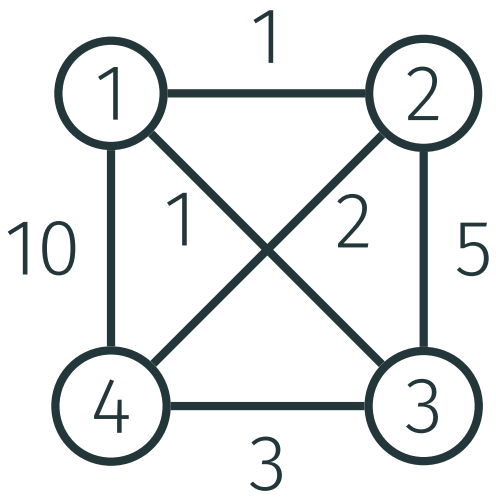
Пример: метод ветвей и границ

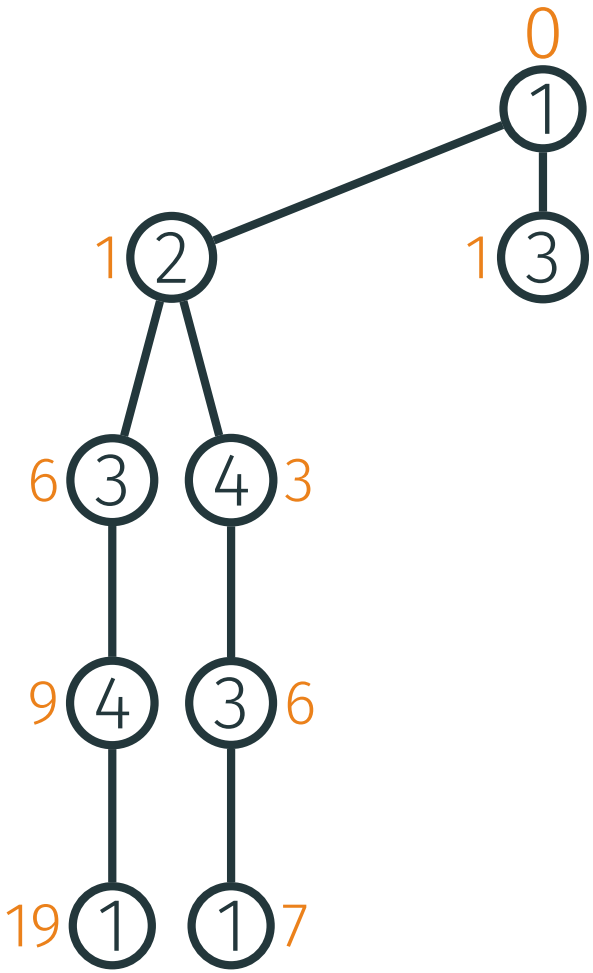




best so far: 7

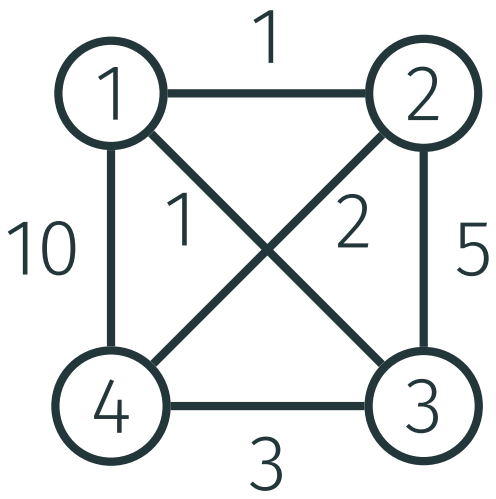
Пример: метод ветвей и границ

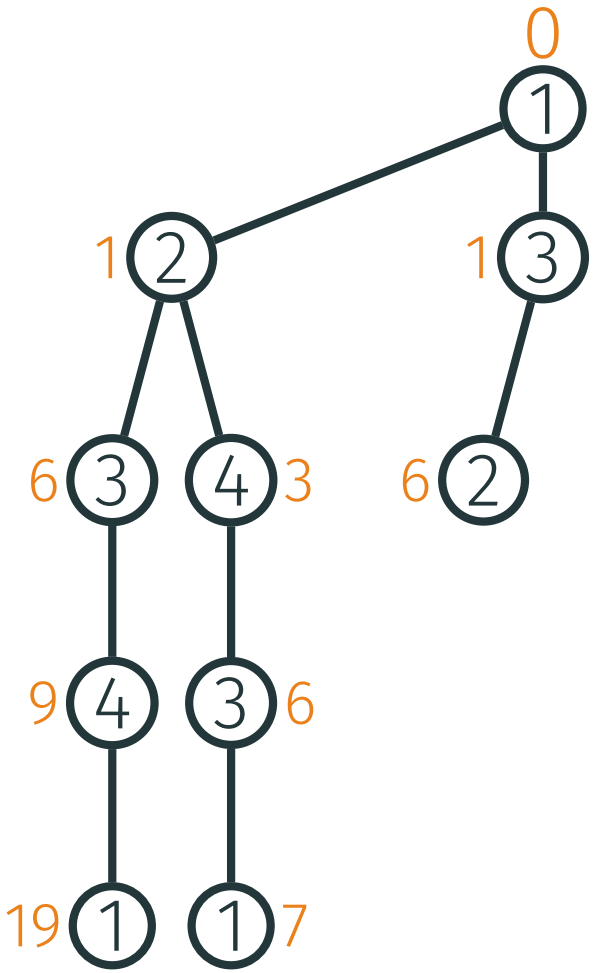




best so far: 7

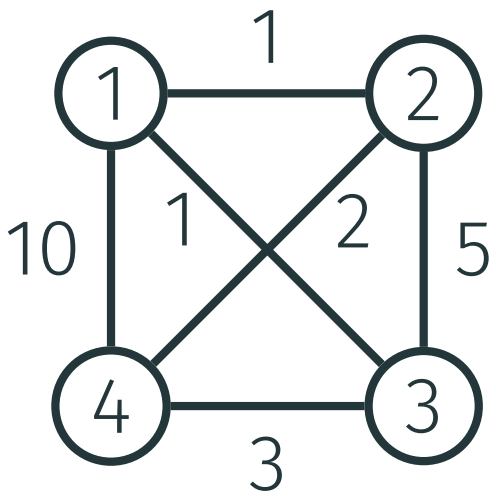
Пример: метод ветвей и границ

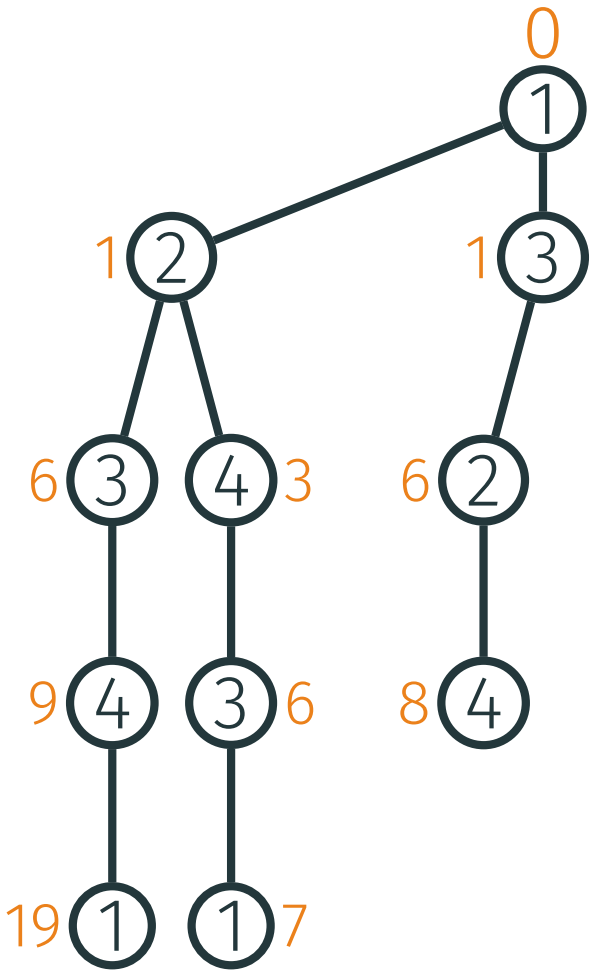




best so far: 7

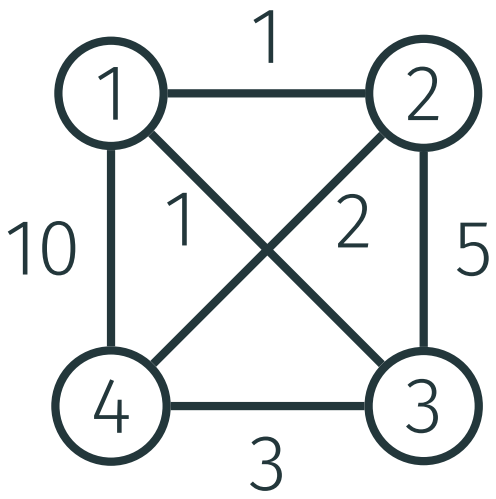
Пример: метод ветвей и границ

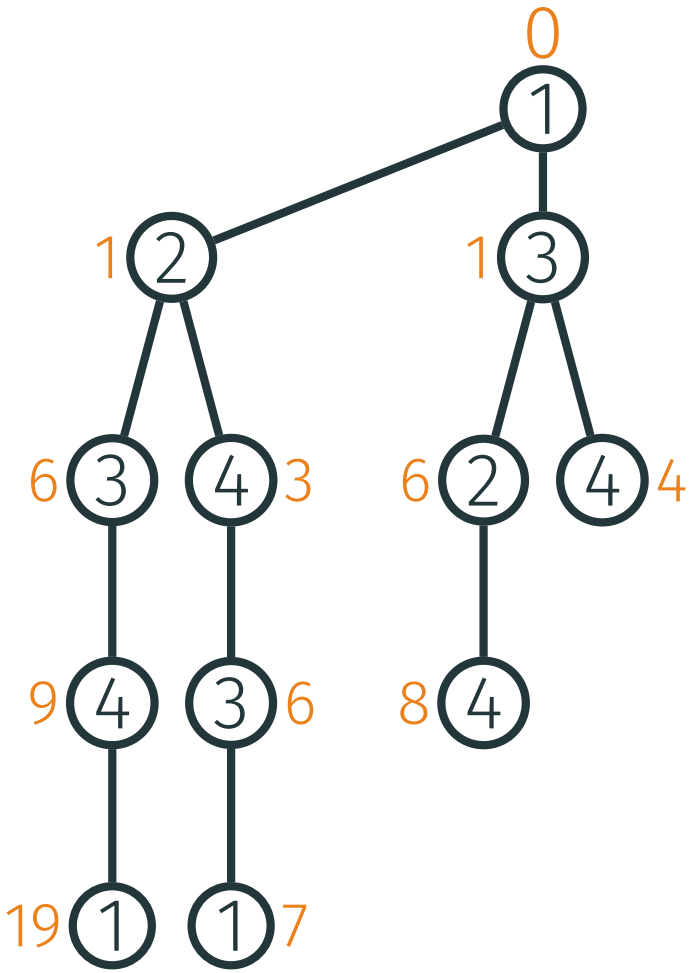




best so far: 7

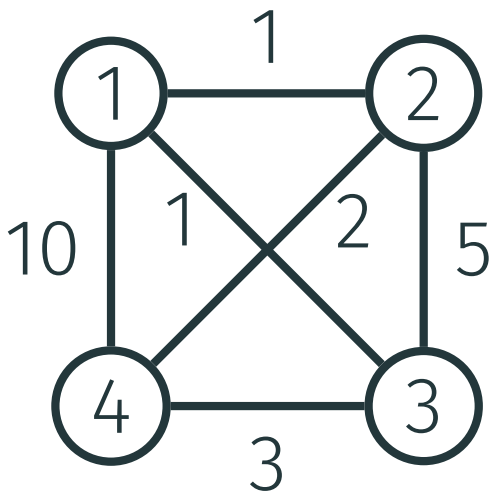
Пример: метод ветвей и границ

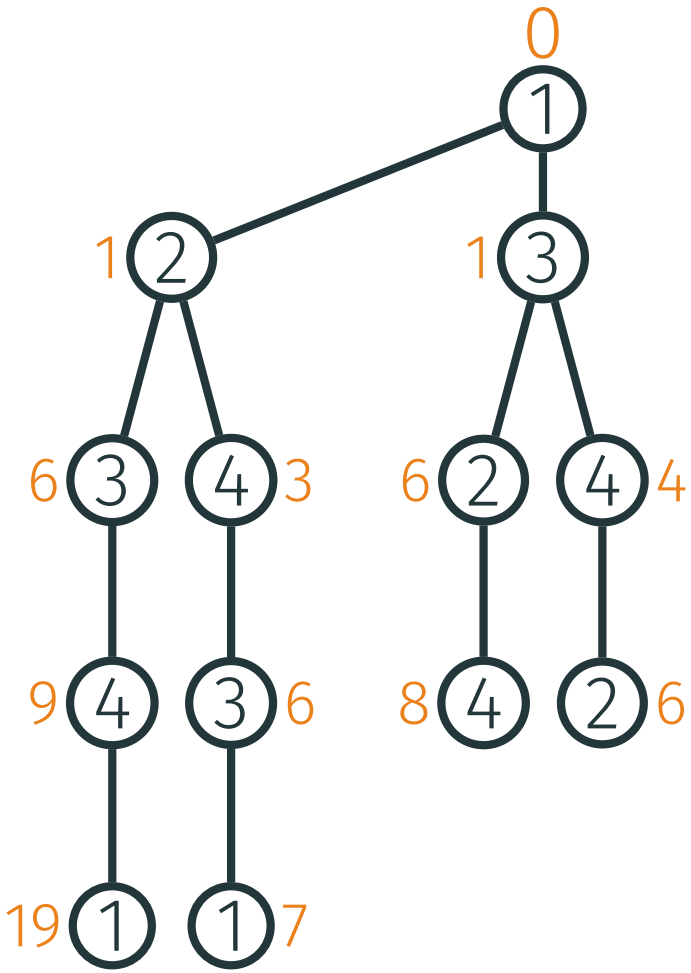




best so far: 7

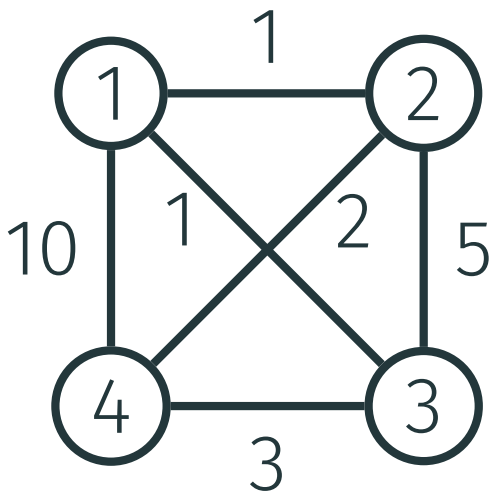
Пример: метод ветвей и границ

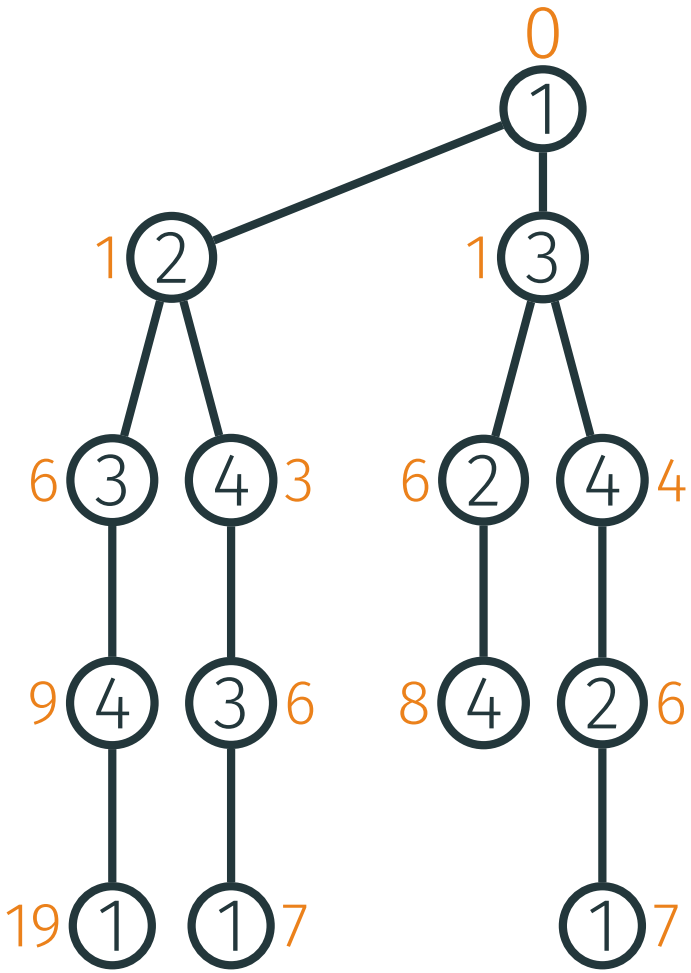




best so far: 7

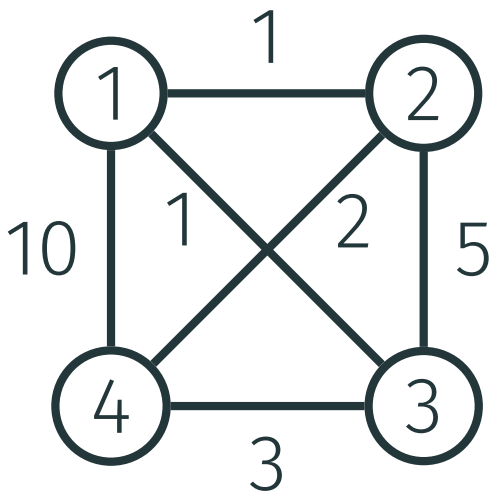
Пример: метод ветвей и границ

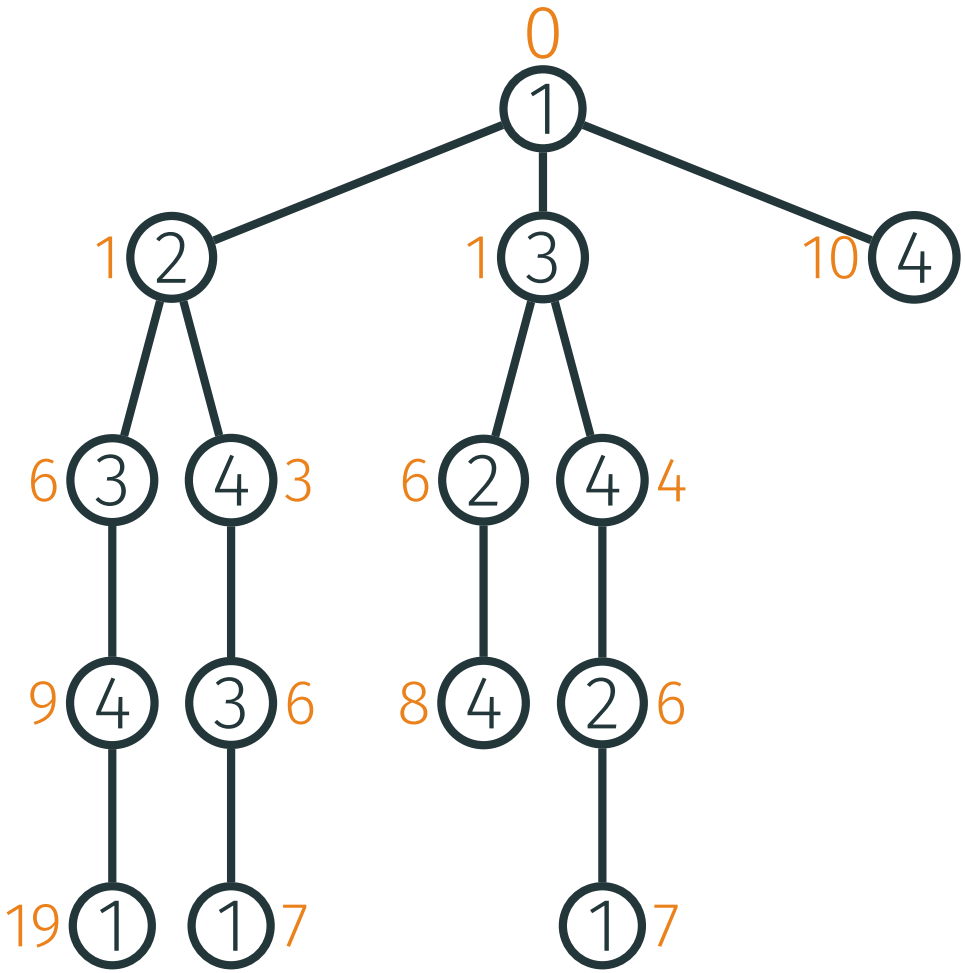




best so far: 7

Пример: метод ветвей и границ





best so far: 7

Нижняя оценка

- Мы использовали простейшую нижнюю оценку: любое удлиннение пути имеет вес не меньше веса пути

Нижняя оценка

- Мы использовали простейшую нижнюю оценку: любое удлиннение пути имеет вес не меньше веса пути
- Современные TSP-солверы используют гораздо более хитрые эвристики для получения нижних оценок

Пример: нижние оценки (всё ещё довольно простые)

Вес оптимального цикла коммивояжёра не меньше

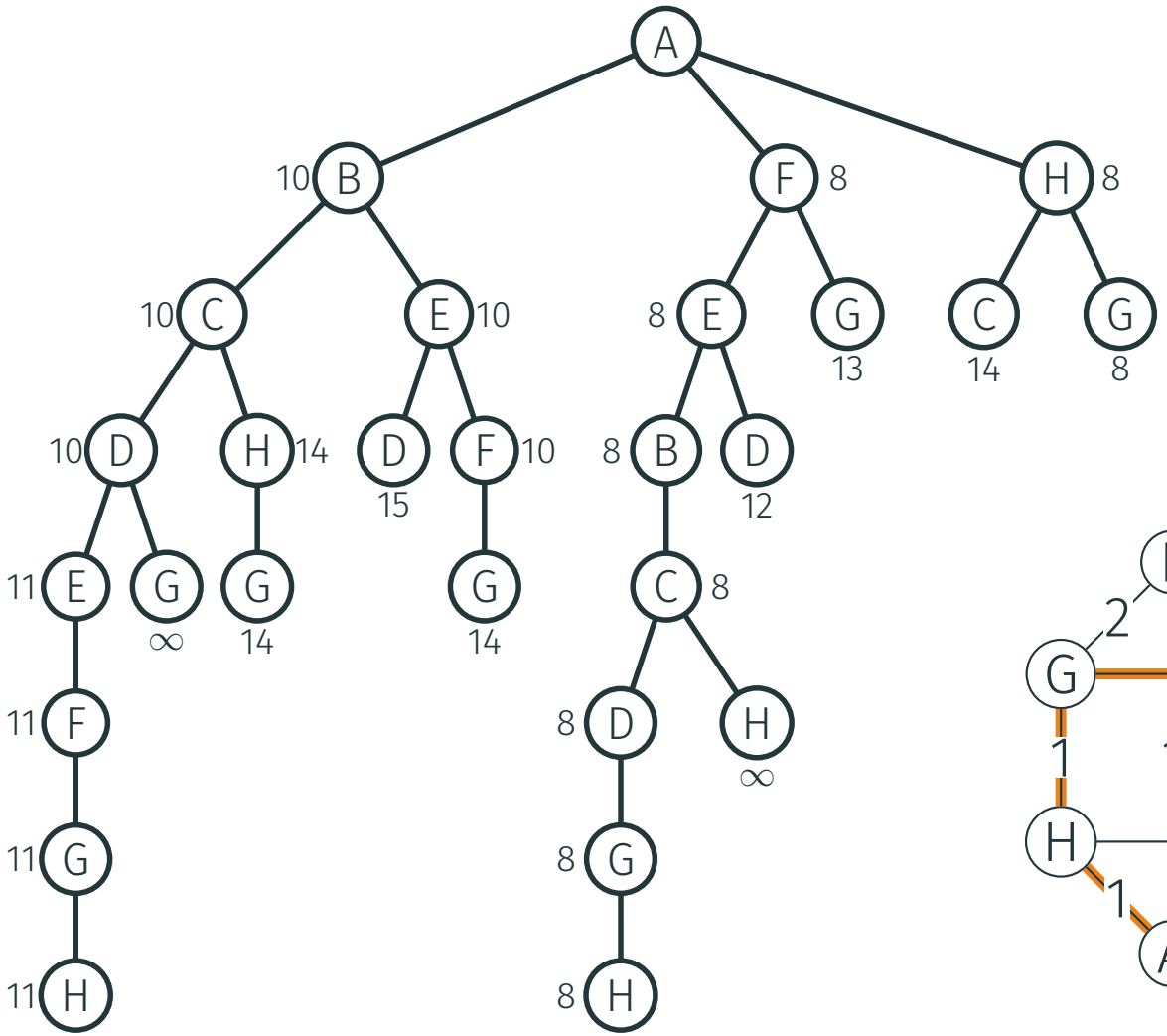
- $\frac{1}{2}\sum_{v \in V}$(два мин. ребра, смежных с v)

Пример: нижние оценки (всё ещё довольно простые)

Вес оптимального цикла коммивояжёра не меньше

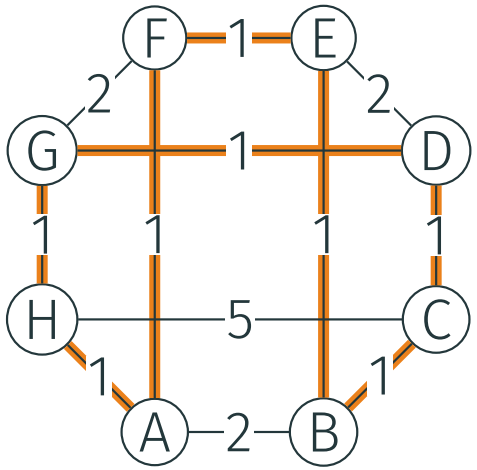
- $\frac{1}{2}\sum_{v \in V}$(два мин. ребра, смежных с v)
- веса минимального покрывающего дерева (при выкидывании ребра из оптимального цикла получается покрывающее дерево)

Ещё пример



Стоимость: 11

Стоимость: 8



Метод ветвей и границ: заключение

- Основные две эвристики:

Метод ветвей и границ: заключение

- Основные две эвристики:
  - Ветви: в каком порядке перебирать ещё не посещённые вершины (например, начинать с ближайших)

Метод ветвей и границ: заключение

- Основные две эвристики:
  - Ветви: в каком порядке перебирать ещё не посещённые вершины (например, начинать с ближайших)
  - Границы: нижняя оценка на длину решения

Метод ветвей и границ: заключение

- Основные две эвристики:
  - Ветви: в каком порядке перебирать ещё не посещённые вершины (например, начинать с ближайших)
  - Границы: нижняя оценка на длину решения
- Находит оптимальное решение

Метод ветвей и границ: заключение

- Основные две эвристики:
  - Ветви: в каком порядке перебирать ещё не посещённые вершины (например, начинать с ближайших)
  - Границы: нижняя оценка на длину решения
- Находит оптимальное решение
- Время работы зависит и от эвристик, и от входных данных

Метод ветвей и границ: заключение

- Основные две эвристики:
  - Ветви: в каком порядке перебирать ещё не посещённые вершины (например, начинать с ближайших)
  - Границы: нижняя оценка на длину решения
- Находит оптимальное решение
- Время работы зависит и от эвристик, и от входных данных
- Используется в современных TSP-солверах, которые способны находить оптимальное решение в графах с тысячами вершин!

Эвристические методы

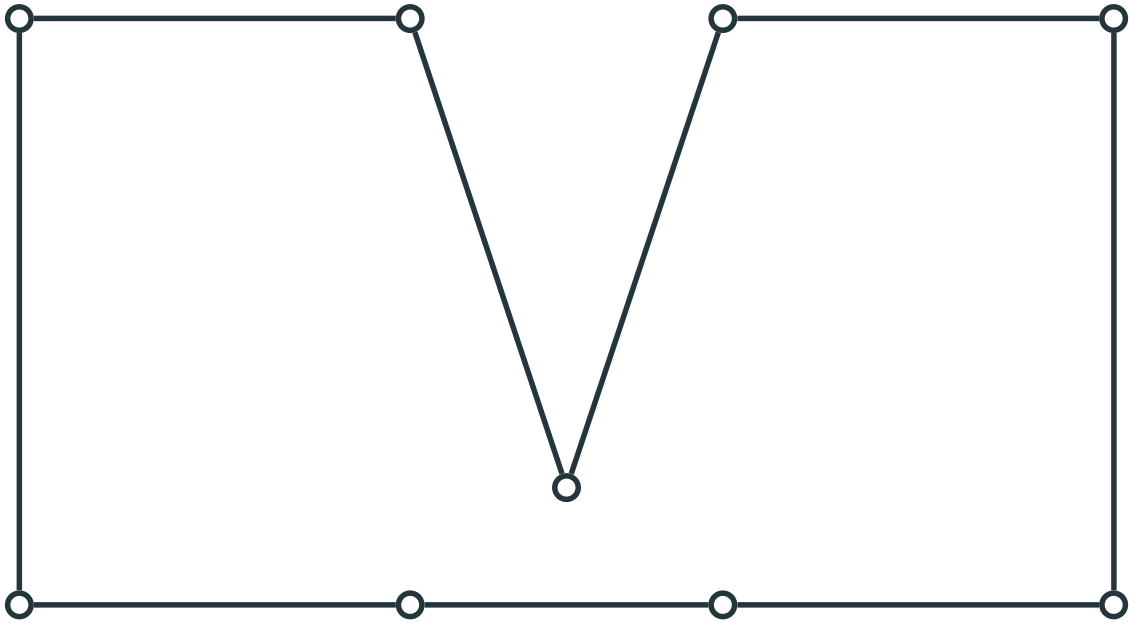
Метод локального поиска

Локальный поиск

$s \leftarrow$ какое-нибудь начальное решение
пока в окрестности s есть решение s'
большего веса:
  заменить s на s'
вернуть s

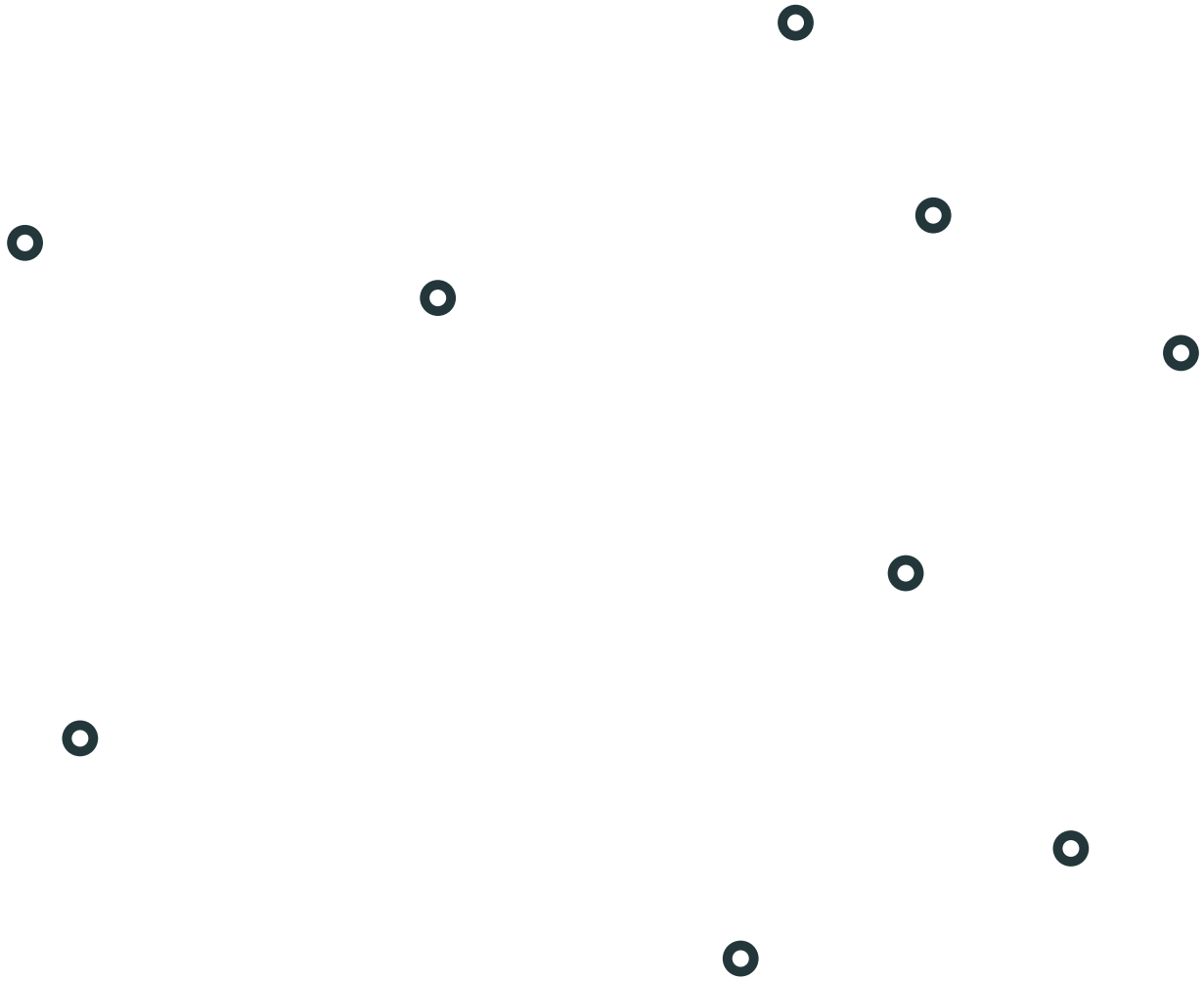
2-окружение

Узкое место

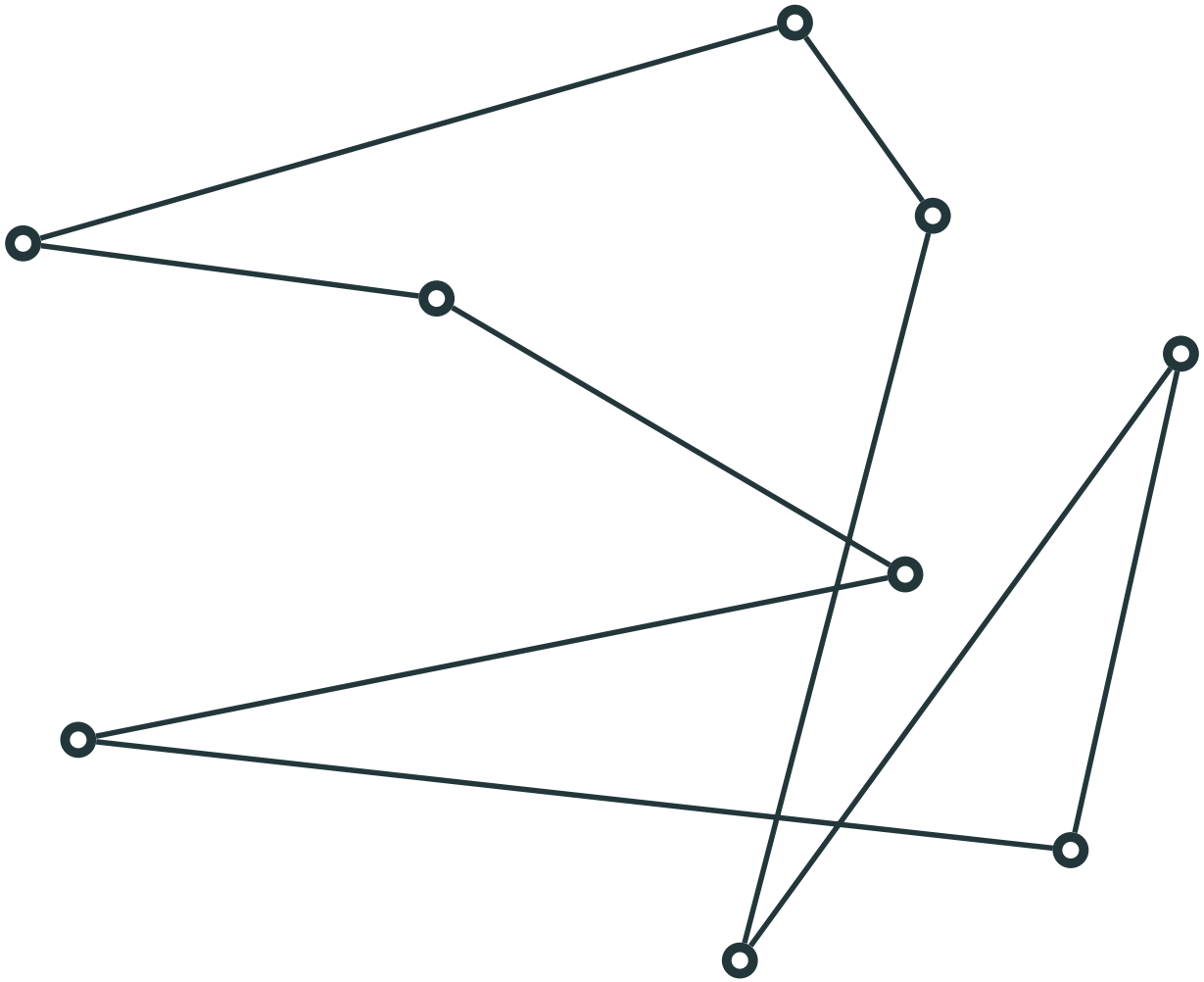


3-окружение

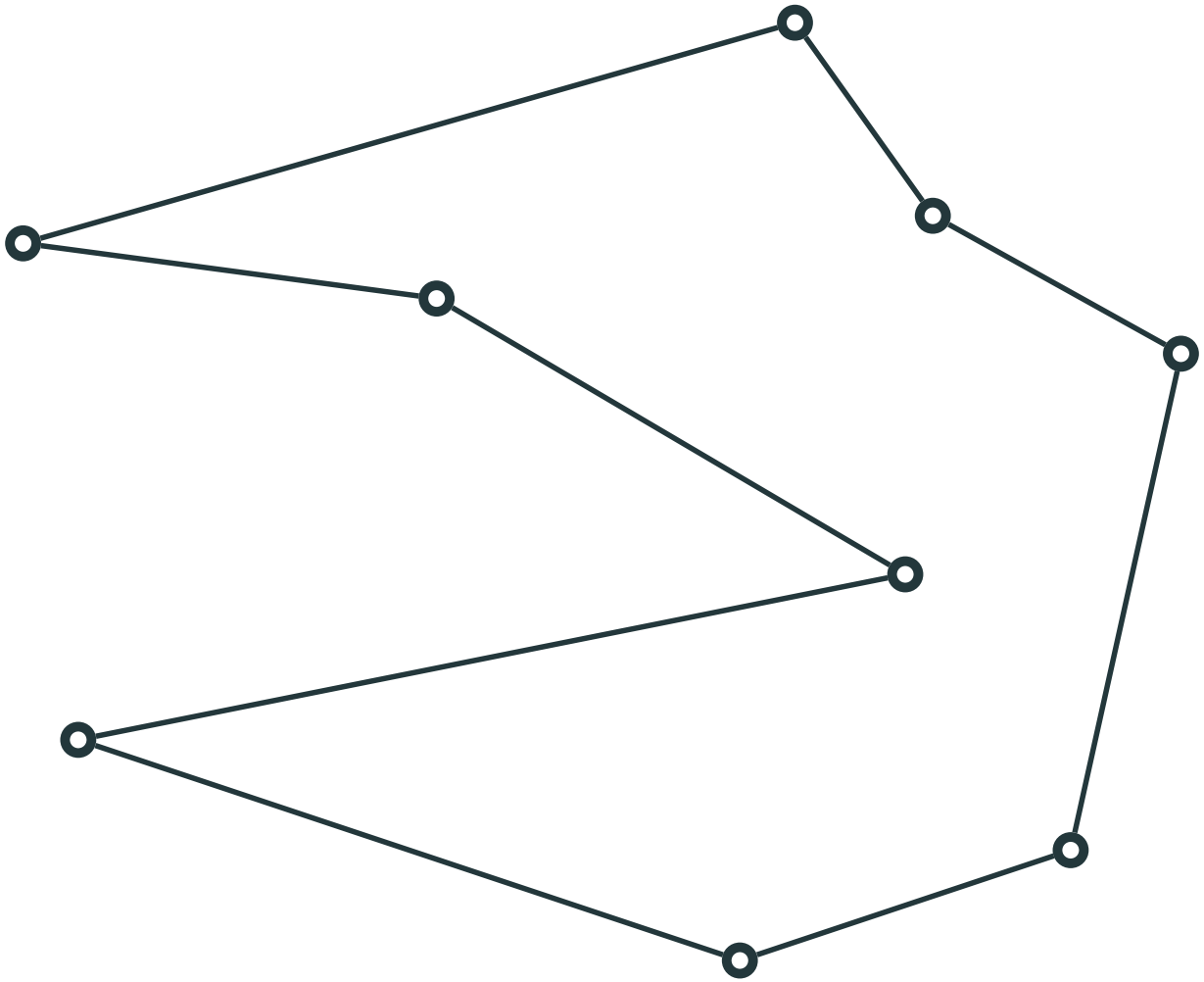
Пример локального поиска (с 3-окружением)



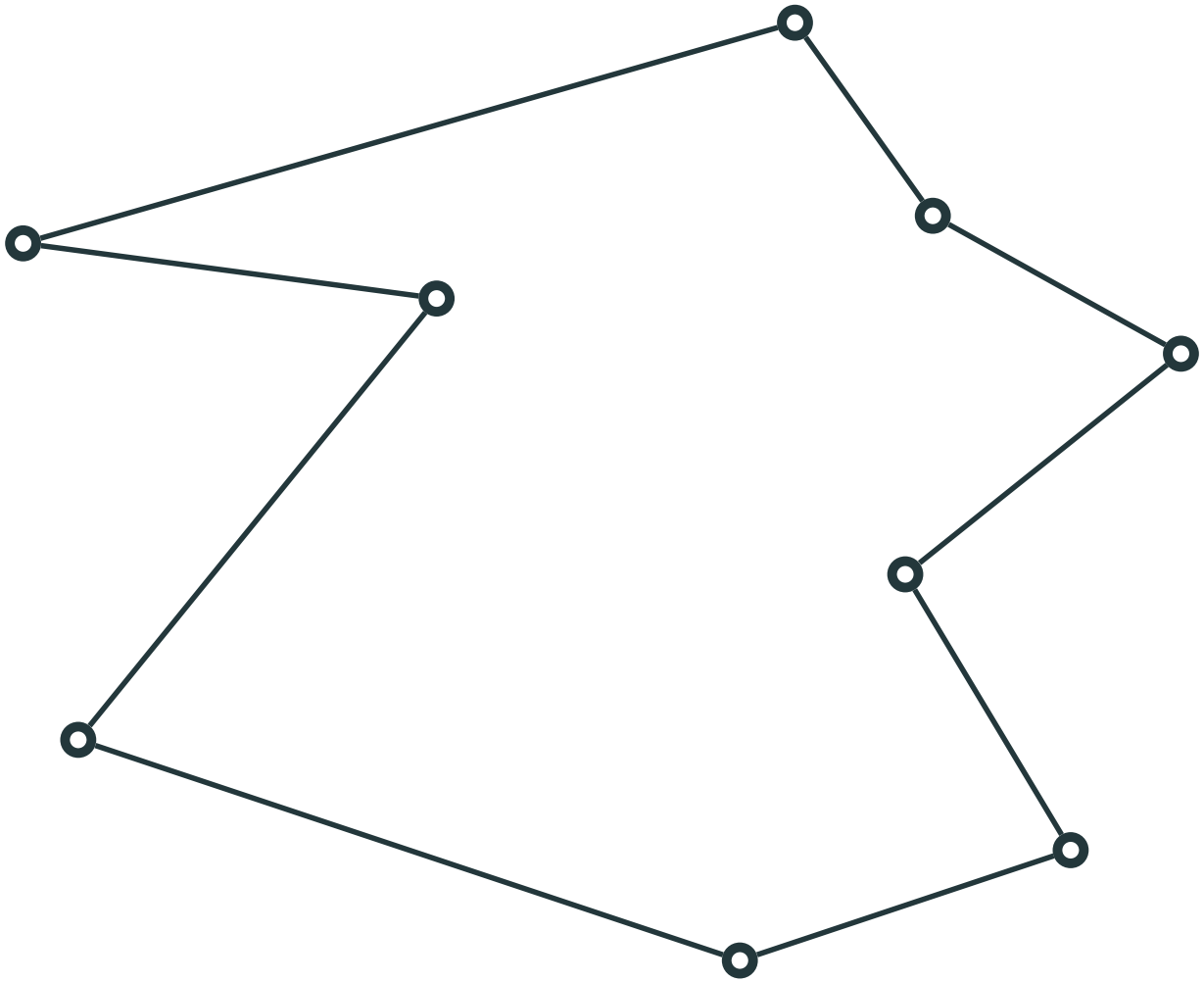
Пример локального поиска (с 3-окружением)



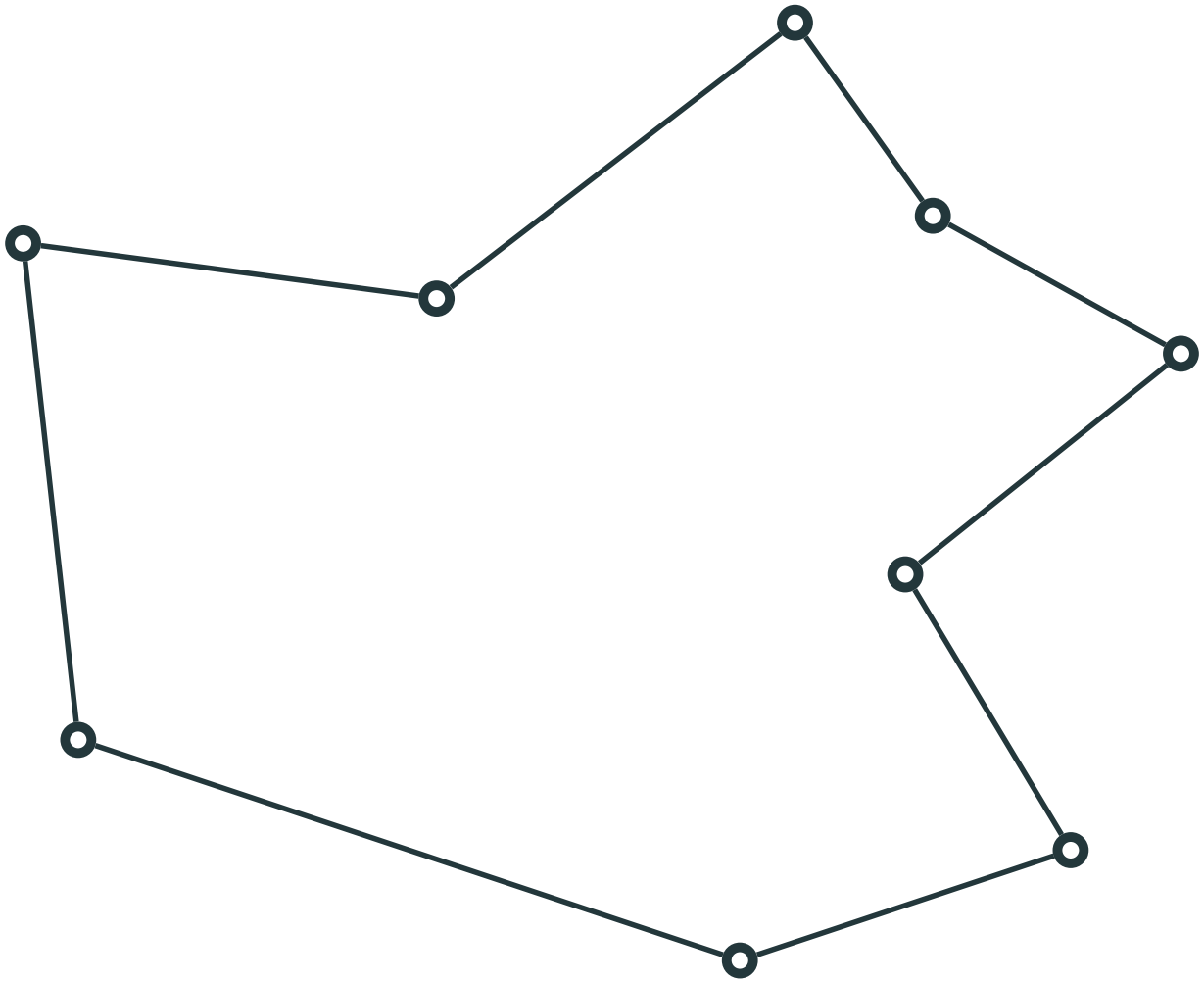
Пример локального поиска (с 3-окружением)



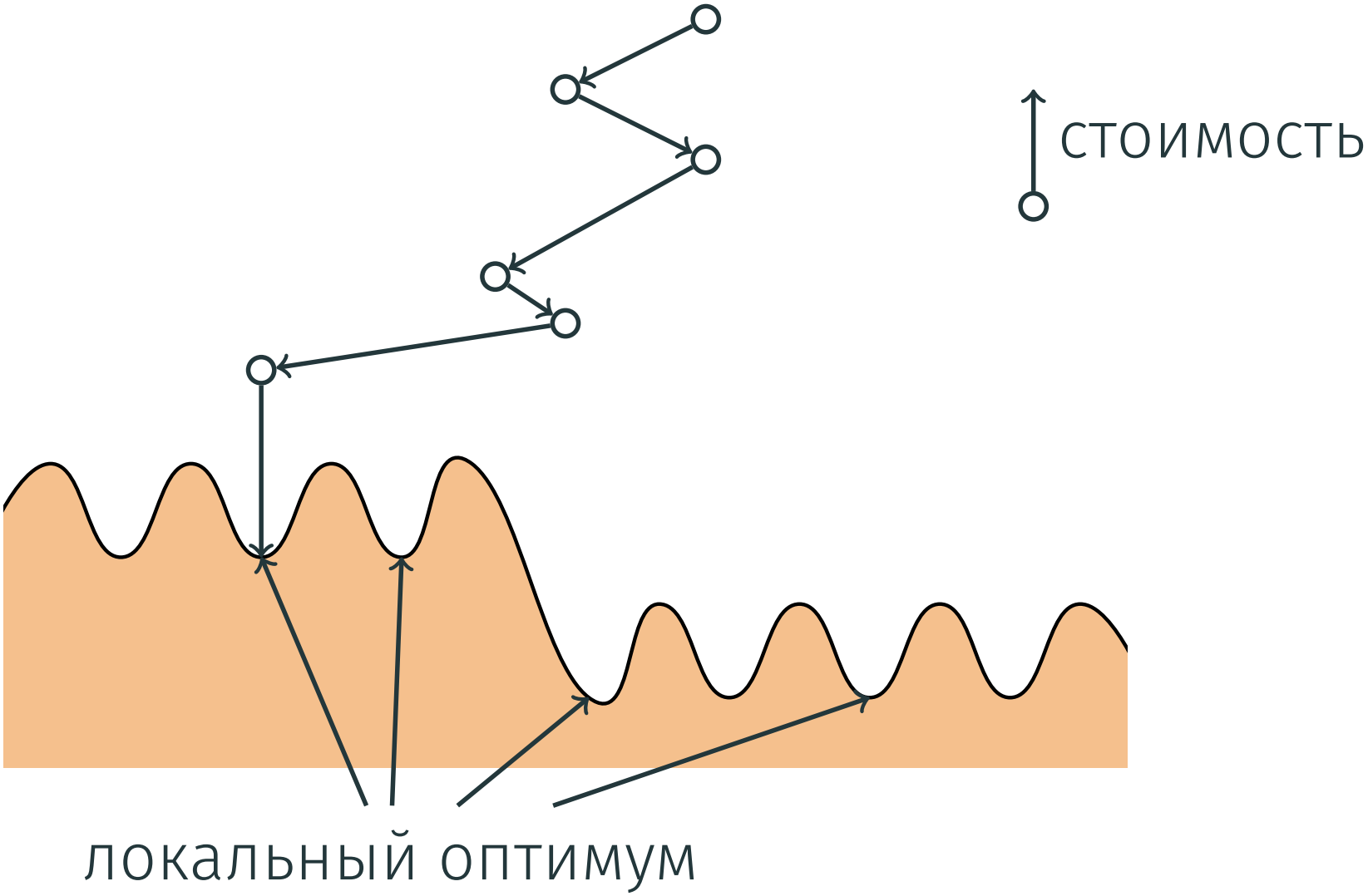
Пример локального поиска (с 3-окружением)



Пример локального поиска (с 3-окружением)



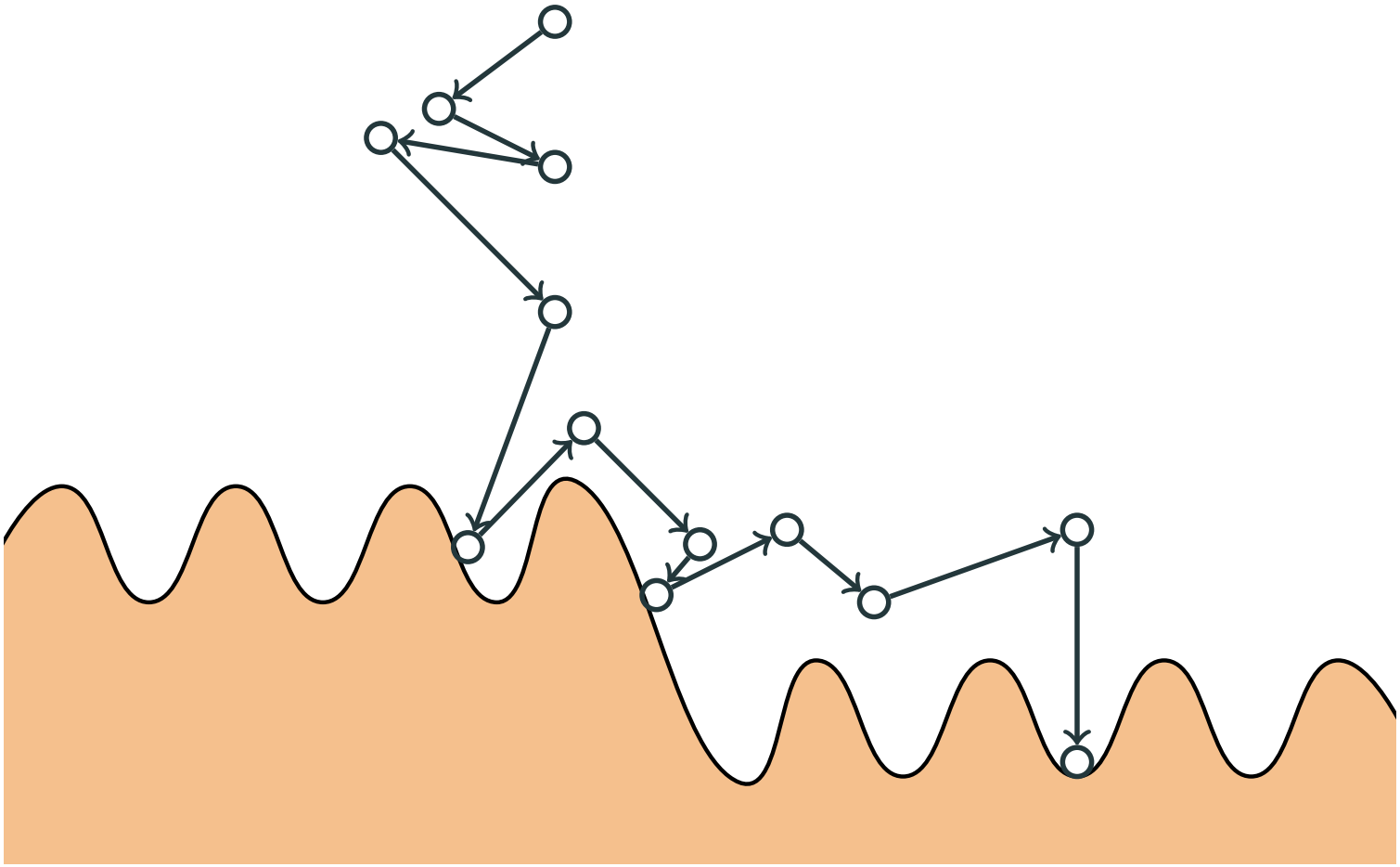
Локальный поиск абстрактно



Метод имитации отжига

$s \leftarrow$ какое-нибудь начальное решение
повторять:
  $s' \leftarrow$ случайное решение из окружения s
  $\Delta \leftarrow \mathtt{cost}(s') - \mathtt{cost}(s)$
  если $\Delta < 0$:
    заменить s на s'
  иначе:
    заменить s на s' с вероятностью $e^{-\Delta/T}$

Метод имитации отжига абстрактно



Приближённые алгоритмы

Приближённые алгоритмы

1.5-приближение для метрического случая

Метрический случай

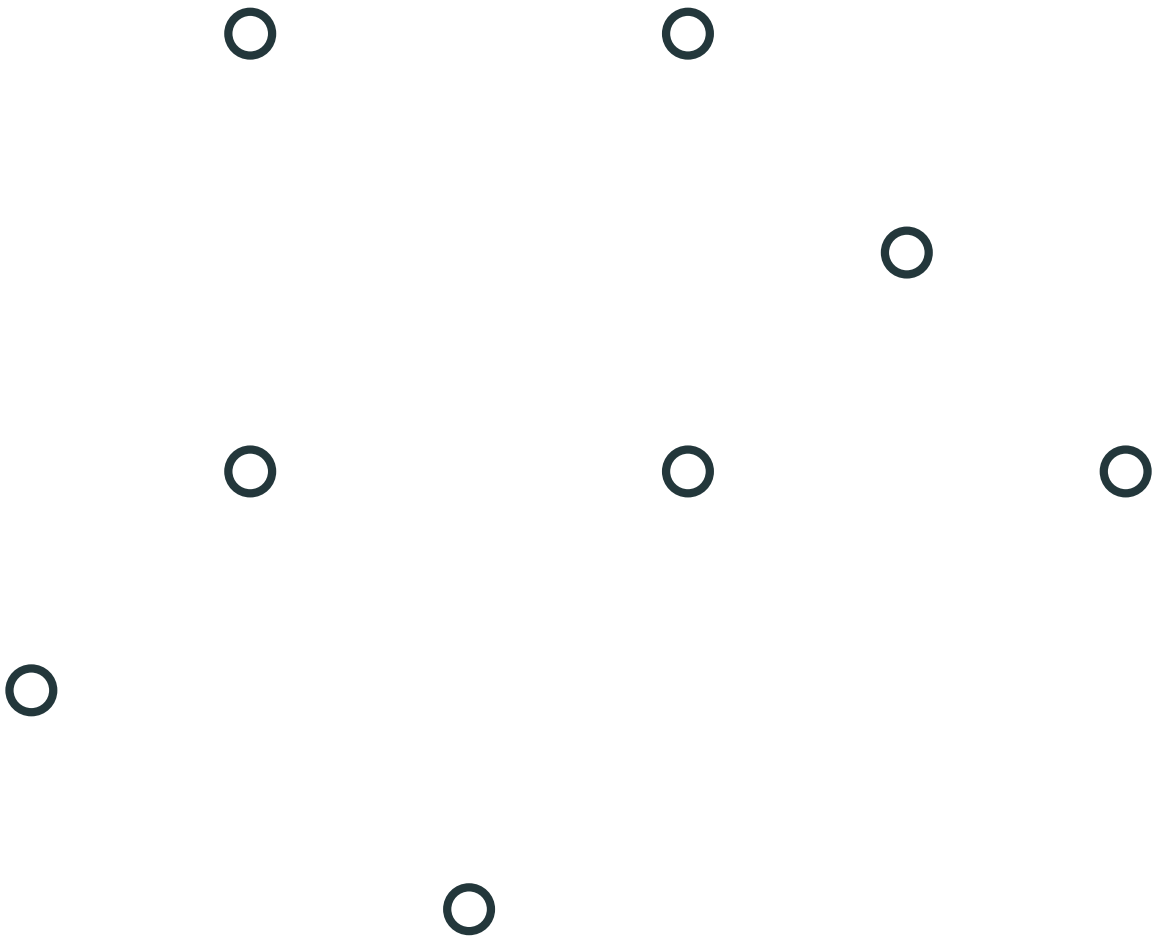
Коммивояжёр в метрическом пространстве: частный случай для неориентированных (симметрических) графов, веса рёбер которых удовлетворяют неравенству треугольника

$$w(i, j) \leq w(i, k) + w(k, j)$$

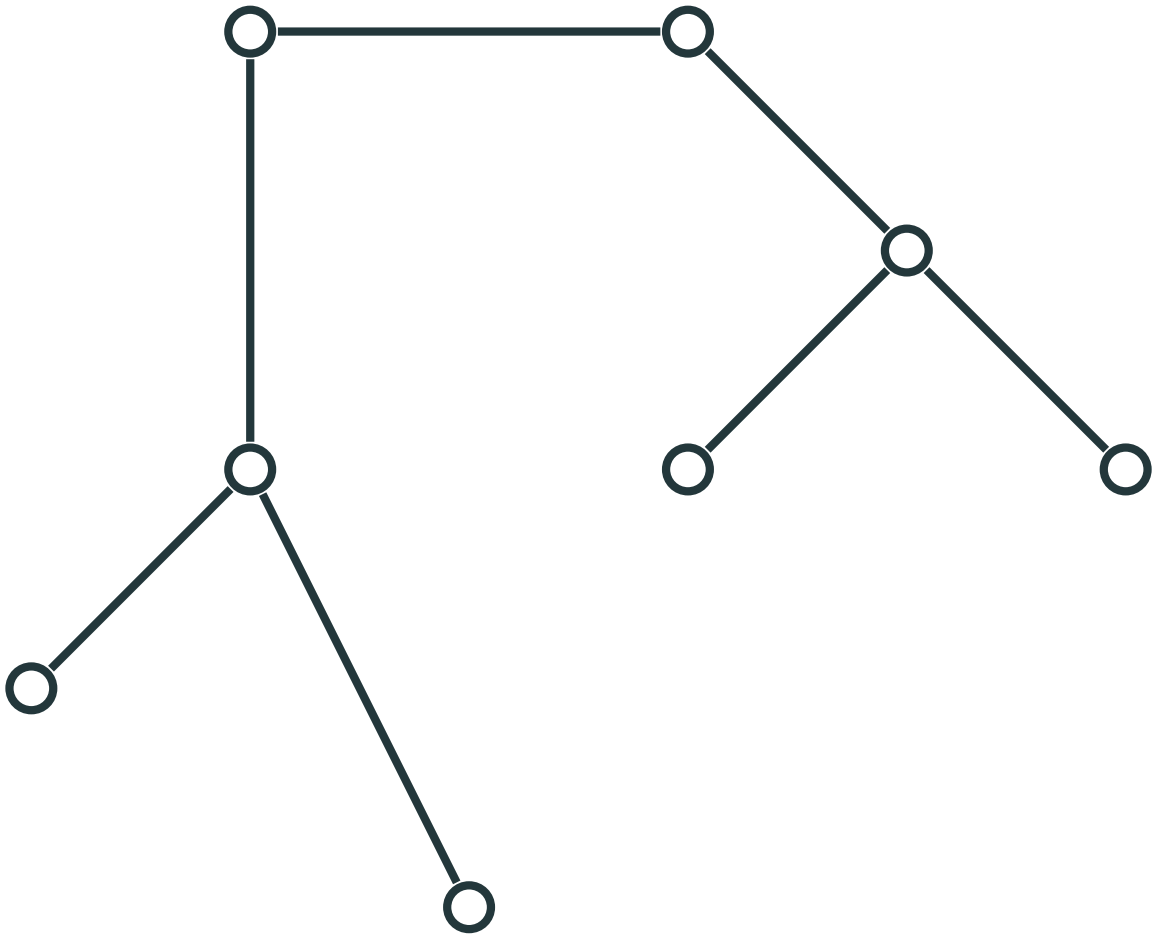
2-приближённый алгоритм

построить минимальное покрывающее дерево T
продублировать каждое ребро дерева T
в полученном графе найти эйлеров цикл
выкинуть из этого цикла все повторения вершин

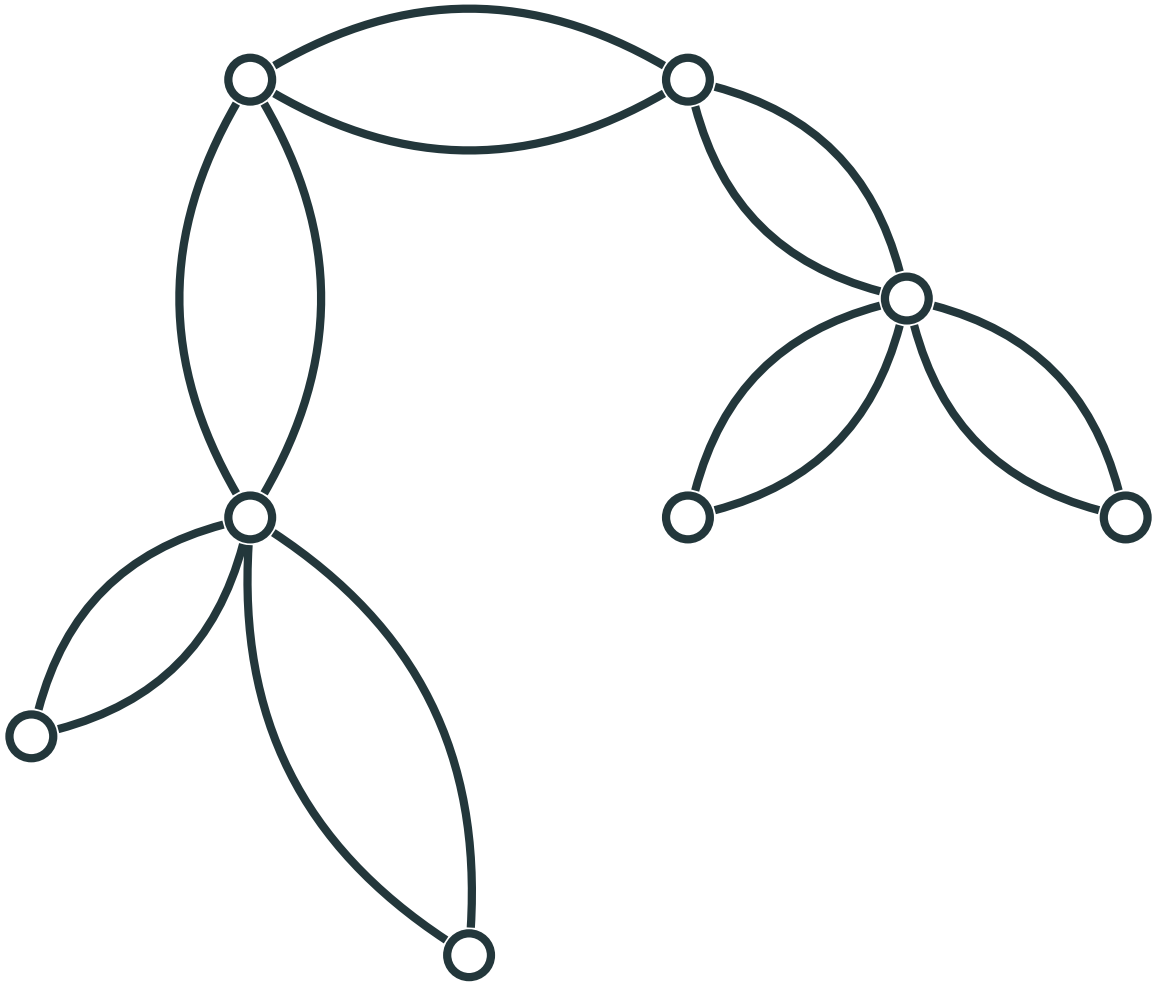
Пример



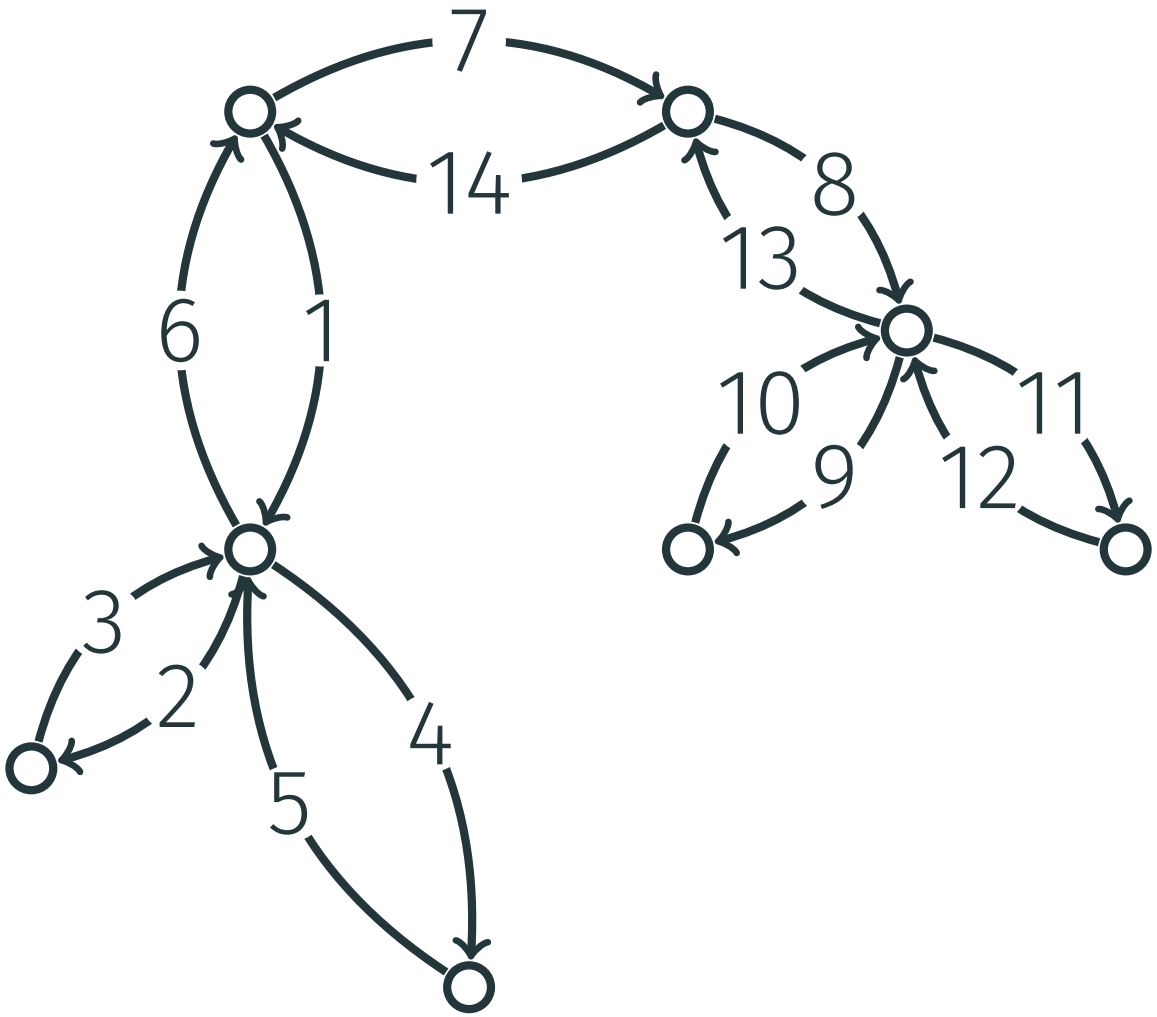
Пример



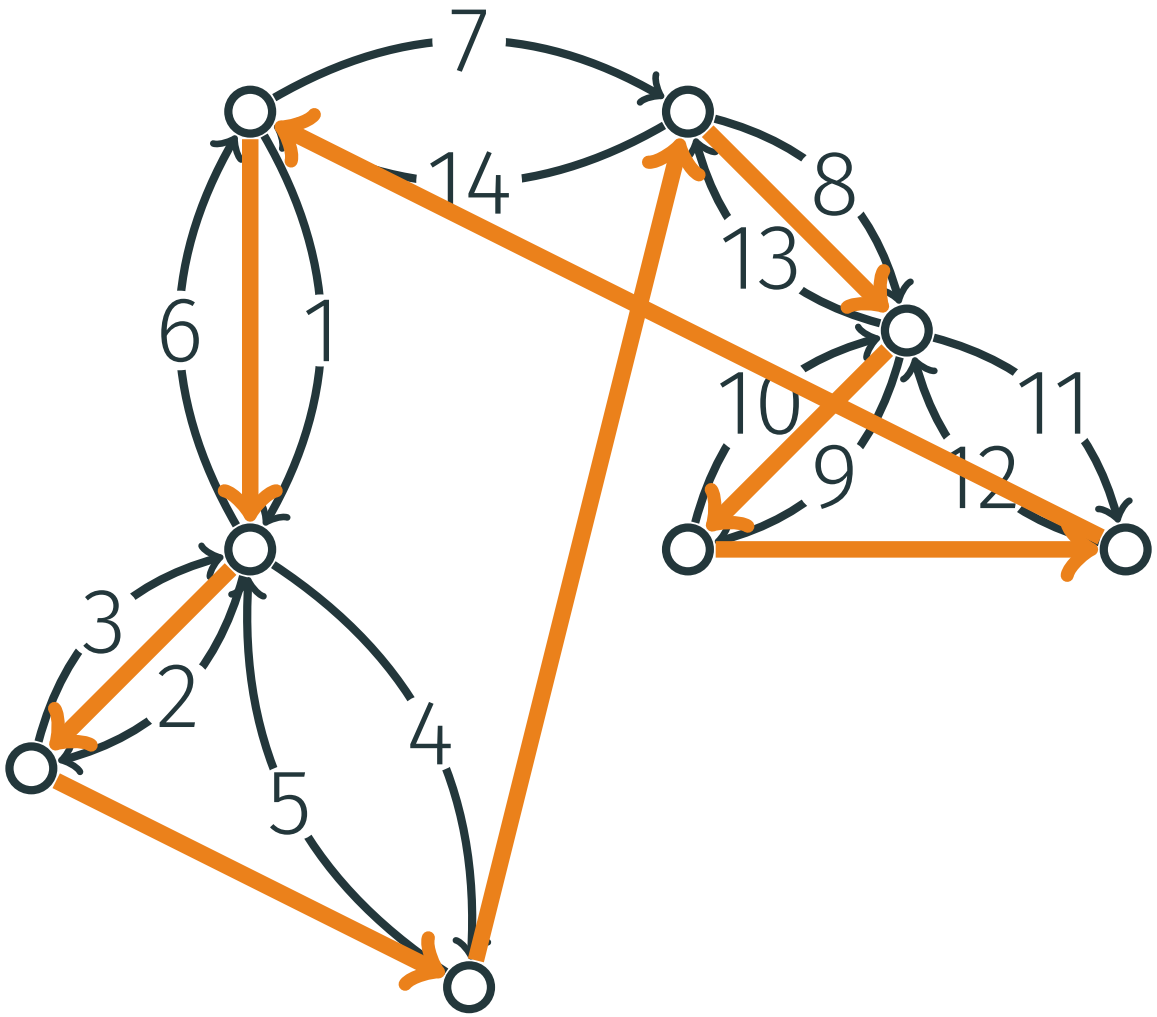
Пример



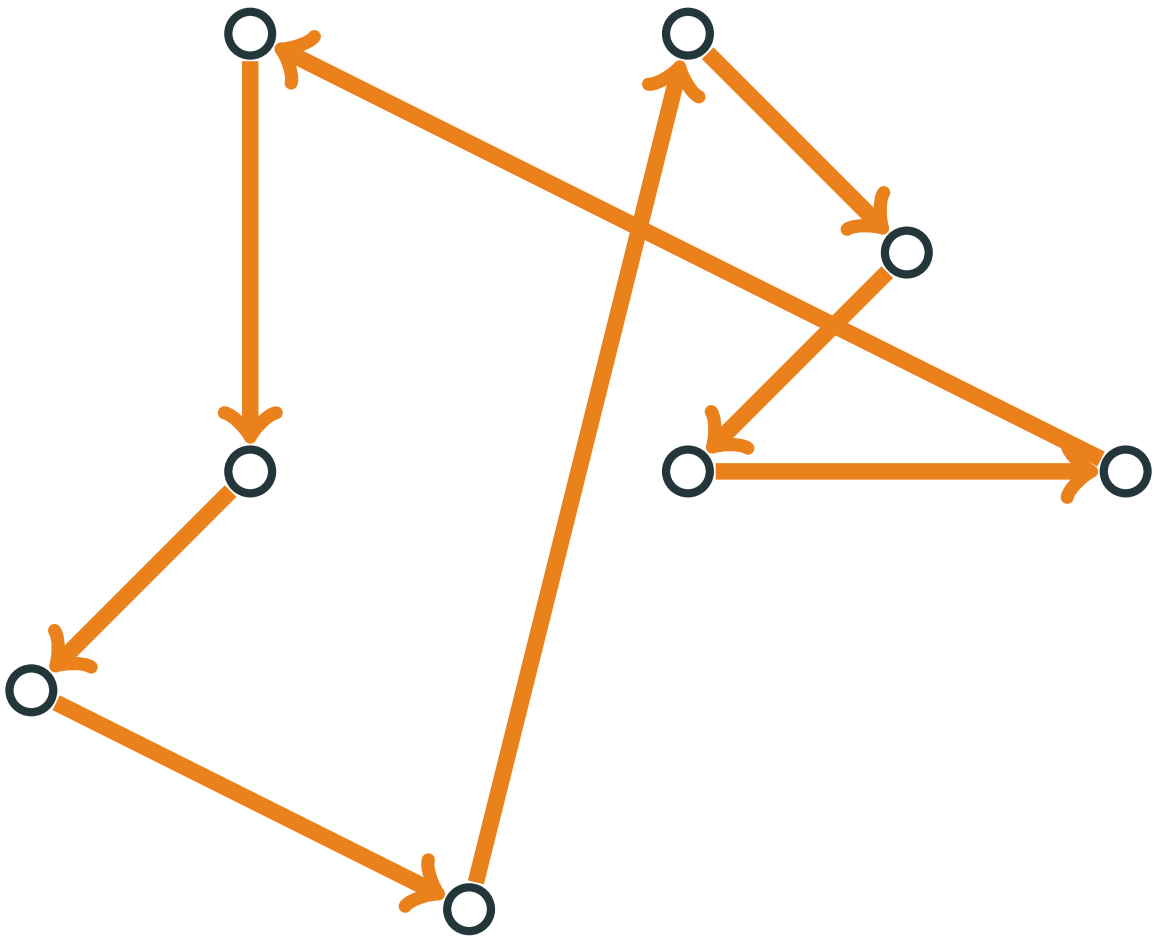
Пример



Пример



Пример



Доказательство

- пусть W_T — вес минимального остовного дерева, а W_{opt} — вес оптимального гамильтонова цикла

Доказательство

- пусть W_T — вес минимального остовного дерева, а W_{opt} — вес оптимального гамильтонова цикла
- $W_T \leq W_{\text{opt}}$, поскольку при выкидывании ребра из гамильтонва цикла получается остовное дерево

Доказательство

- пусть W_T — вес минимального остовного дерева, а W_{opt} — вес оптимального гамильтонова цикла
- $W_T \leq W_{\text{opt}}$, поскольку при выкидывании ребра из гамильтонва цикла получается остовное дерево
- каждое ребро построенного гамильтонова цикла заменяет какой-то путь эйлерова цикла, длина которого по неравенству треугольника не менее длины этого ребра

Доказательство

- пусть W_T — вес минимального остовного дерева, а W_{opt} — вес оптимального гамильтонова цикла
- $W_T \leq W_{\text{opt}}$, поскольку при выкидывании ребра из гамильтонва цикла получается остовное дерево
- каждое ребро построенного гамильтонова цикла заменяет какой-то путь эйлерова цикла, длина которого по неравенству треугольника не менее длины этого ребра
- значит, длина найденного пути не превосходит $2W_T$, а следовательно, и $2W_{\text{opt}}$ $\square$

1.5-приближённый алгоритм

построить минимальное покрывающее дерево T
найти минимальное полное паросочетание всех вершин дерева T нечетной степени
добавить найденные рёбра в дерево T
найти в полученном графе эйлеров цикл
выкинуть из этого цикла все повторения вершин

Доказательство

- как и в предыдущем доказательстве, вес построенного цикла не превосходит $W_T + W_P$, где W_P — вес минимального паросочетания вершин нечетной степени дерева T

Доказательство

- как и в предыдущем доказательстве, вес построенного цикла не превосходит $W_T + W_P$, где W_P — вес минимального паросочетания вершин нечетной степени дерева T
- нужно показать, что $W_P \leq W_{\text{opt}}/2$

Доказательство

- как и в предыдущем доказательстве, вес построенного цикла не превосходит $W_T + W_P$, где W_P — вес минимального паросочетания вершин нечетной степени дерева T
- нужно показать, что $W_P \leq W_{\text{opt}}/2$
- обозначим через A множество всех вершин нечётной степени дерева T

Доказательство

- как и в предыдущем доказательстве, вес построенного цикла не превосходит $W_T + W_P$, где W_P — вес минимального паросочетания вершин нечетной степени дерева T
- нужно показать, что $W_P \leq W_{\text{opt}}/2$
- обозначим через A множество всех вершин нечётной степени дерева T
- рассмотрим такой гамильтонов цикл на вершинах множества A: вершины множества A в нём будут встречаться в такой последовательности, в какой они идут в оптимальном гамильтоновом цикле графа G

Доказательство (продолжение)

- важно отметить, что нам не нужно строить такой цикл; нам важен лишь факт его существования

Доказательство (продолжение)

- важно отметить, что нам не нужно строить такой цикл; нам важен лишь факт его существования
- разбив вершины только что построенного цикла на чётные и нечётные, мы получим два паросочетания

Доказательство (продолжение)

- важно отметить, что нам не нужно строить такой цикл; нам важен лишь факт его существования
- разбив вершины только что построенного цикла на чётные и нечётные, мы получим два паросочетания
- вес хотя бы одного из них будет не более $W_{\text{opt}}/2$

Доказательство (продолжение)

- важно отметить, что нам не нужно строить такой цикл; нам важен лишь факт его существования
- разбив вершины только что построенного цикла на чётные и нечётные, мы получим два паросочетания
- вес хотя бы одного из них будет не более $W_{\text{opt}}/2$
- значит, и вес минимального паросочетания не превосходит $W_{\text{opt}}/2$ $\square$

Цитата

Almost every graduate student in theoretical computer science has at some point spent a few futile weeks or months trying to improve upon Christofides' algorithm.

Sanjeev Arora

Приближённые алгоритмы

Неприближаемость общего случая

Неприближаемость

- Предположим, что существует α-приближённый алгоритм для задачи коммивояжёра.

Неприближаемость

- Предположим, что существует α-приближённый алгоритм для задачи коммивояжёра.
- Возьмём тогда произвольный (невзвешенный и необязательно полный) граф и присвоим всем его рёбрам вес 1.

Неприближаемость

- Предположим, что существует α-приближённый алгоритм для задачи коммивояжёра.
- Возьмём тогда произвольный (невзвешенный и необязательно полный) граф и присвоим всем его рёбрам вес 1.
- Между любыми двумя не соединёнными ребром вершинами добавим ребро веса $\alpha n + 1$.

Неприближаемость

- Предположим, что существует α-приближённый алгоритм для задачи коммивояжёра.
- Возьмём тогда произвольный (невзвешенный и необязательно полный) граф и присвоим всем его рёбрам вес 1.
- Между любыми двумя не соединёнными ребром вершинами добавим ребро веса $\alpha n + 1$.
- Заметим теперь, что если в исходном графе существует гамильтонов цикл, то в новом графе существует гамильтонов цикл веса n.

Неприближаемость (продолжение)

- Если же такого цикла в исходном графе нет, то самый лёгкий цикл в новом графе имеет вес хотя бы $(\alpha n + 1) + (n - 1) > \alpha n$.

Неприближаемость (продолжение)

- Если же такого цикла в исходном графе нет, то самый лёгкий цикл в новом графе имеет вес хотя бы $(\alpha n + 1) + (n - 1) > \alpha n$.
- Таким образом, с помощью α-приближенного алгоритма для задачи о коммивояжёре мы можем понять, стоимость оптимального цикла в построенном графе превосходит n или нет.

Неприближаемость (продолжение)

- Если же такого цикла в исходном графе нет, то самый лёгкий цикл в новом графе имеет вес хотя бы $(\alpha n + 1) + (n - 1) > \alpha n$.
- Таким образом, с помощью α-приближенного алгоритма для задачи о коммивояжёре мы можем понять, стоимость оптимального цикла в построенном графе превосходит n или нет.
- А это позволит нам понять (за полиномиальное время!), есть в исходном графе гамильтонов цикл или нет.

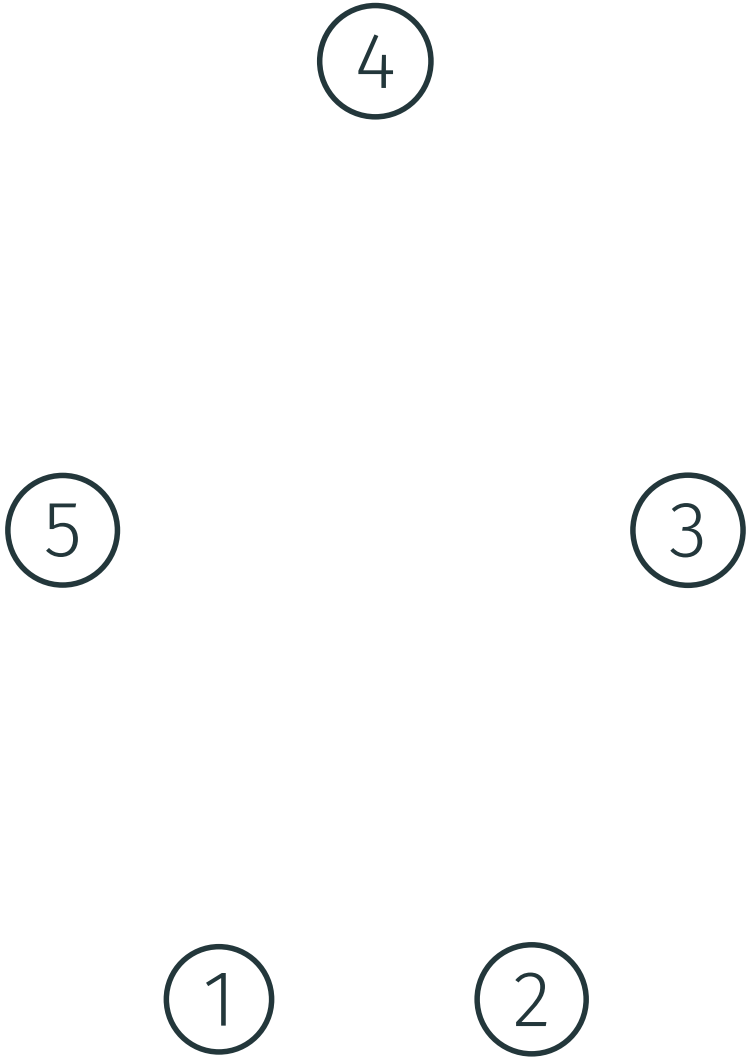
Неприближаемость (продолжение)

- Если же такого цикла в исходном графе нет, то самый лёгкий цикл в новом графе имеет вес хотя бы $(\alpha n + 1) + (n - 1) > \alpha n$.
- Таким образом, с помощью α-приближенного алгоритма для задачи о коммивояжёре мы можем понять, стоимость оптимального цикла в построенном графе превосходит n или нет.
- А это позволит нам понять (за полиномиальное время!), есть в исходном графе гамильтонов цикл или нет.
- Но тогда $\mathrm{P} = \mathrm{NP}$. $\square$

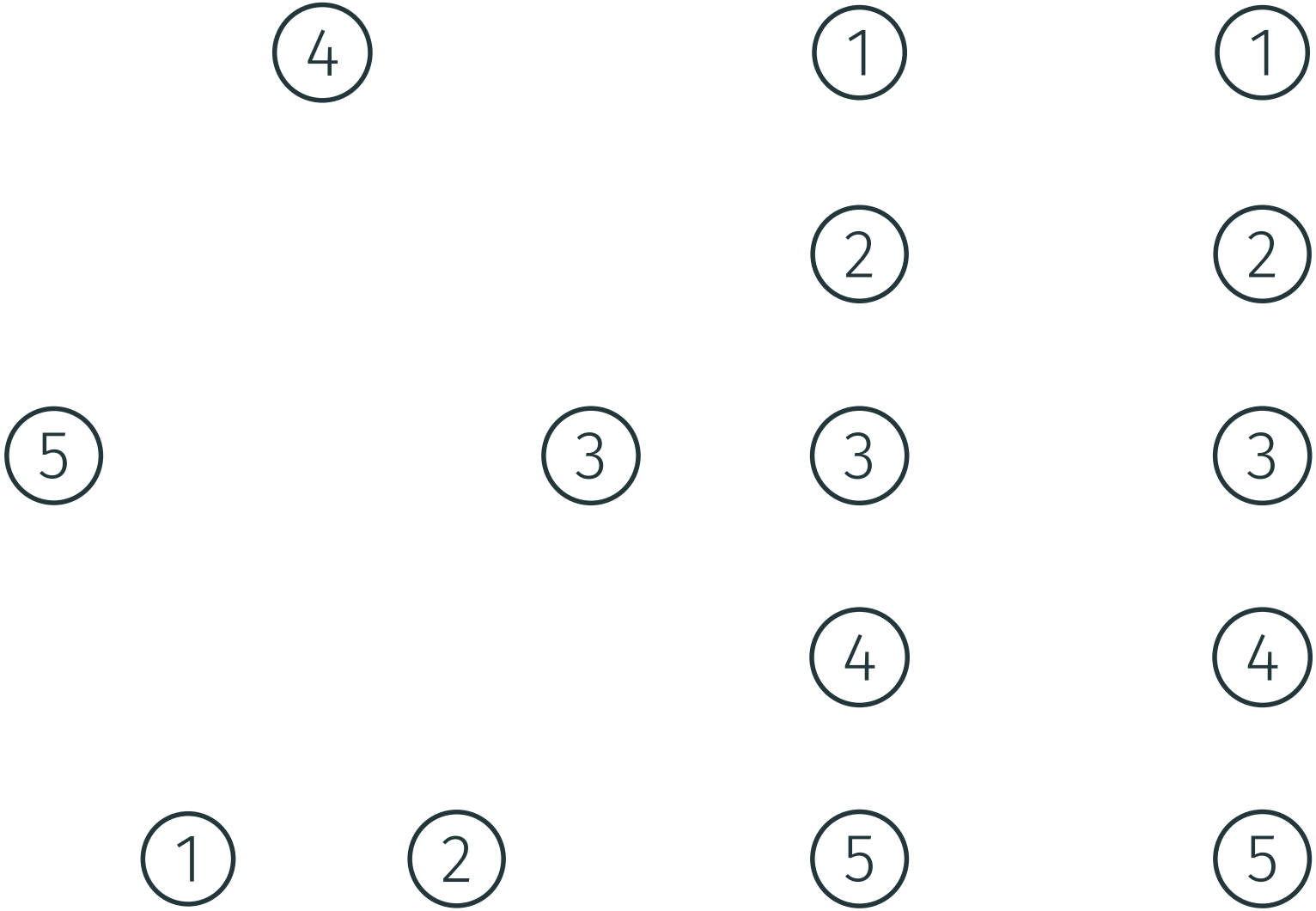
Приближённые алгоритмы

1/2-приближение для максимизационной версии

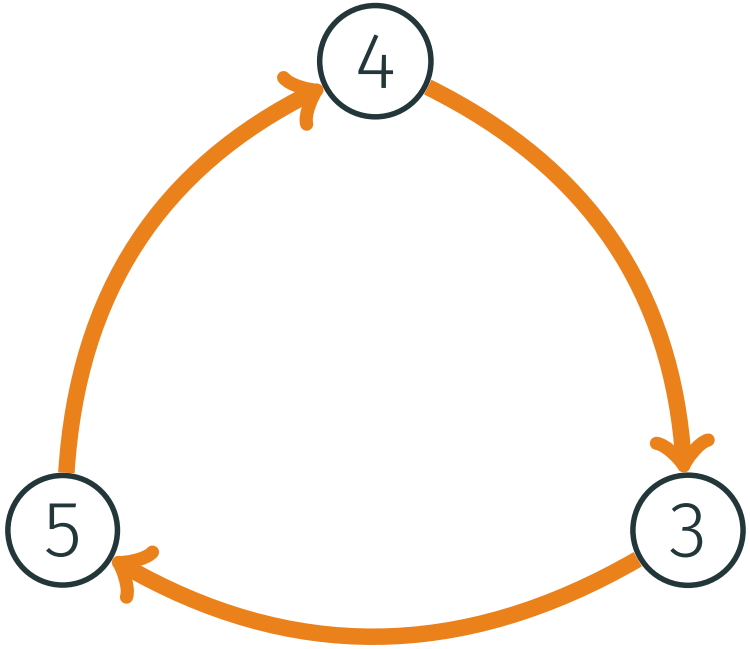
Покрытие ориентированными циклами

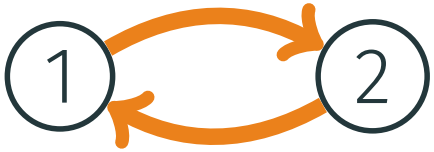


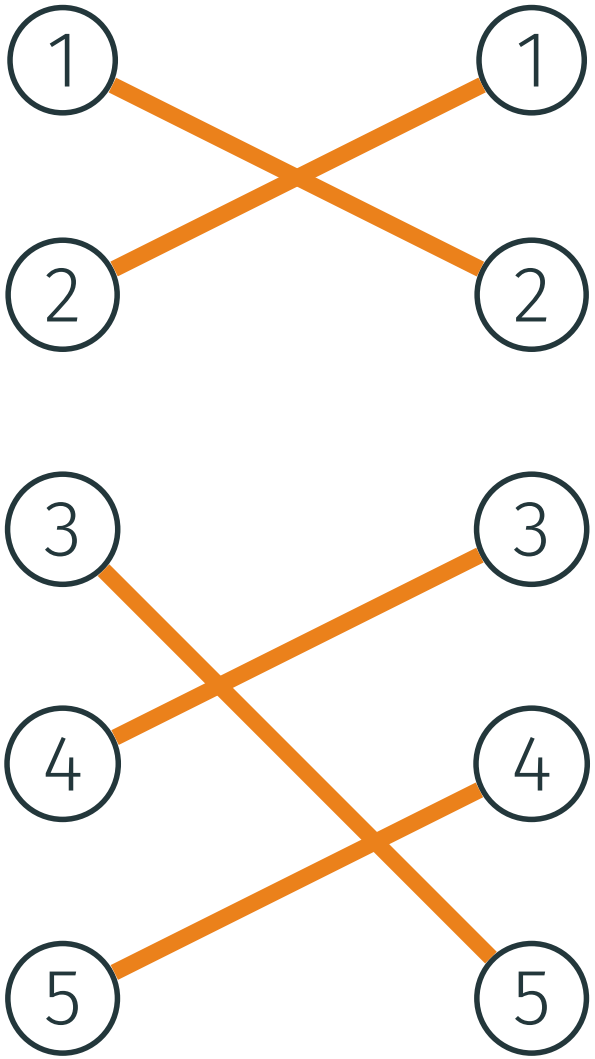
Покрытие ориентированными циклами



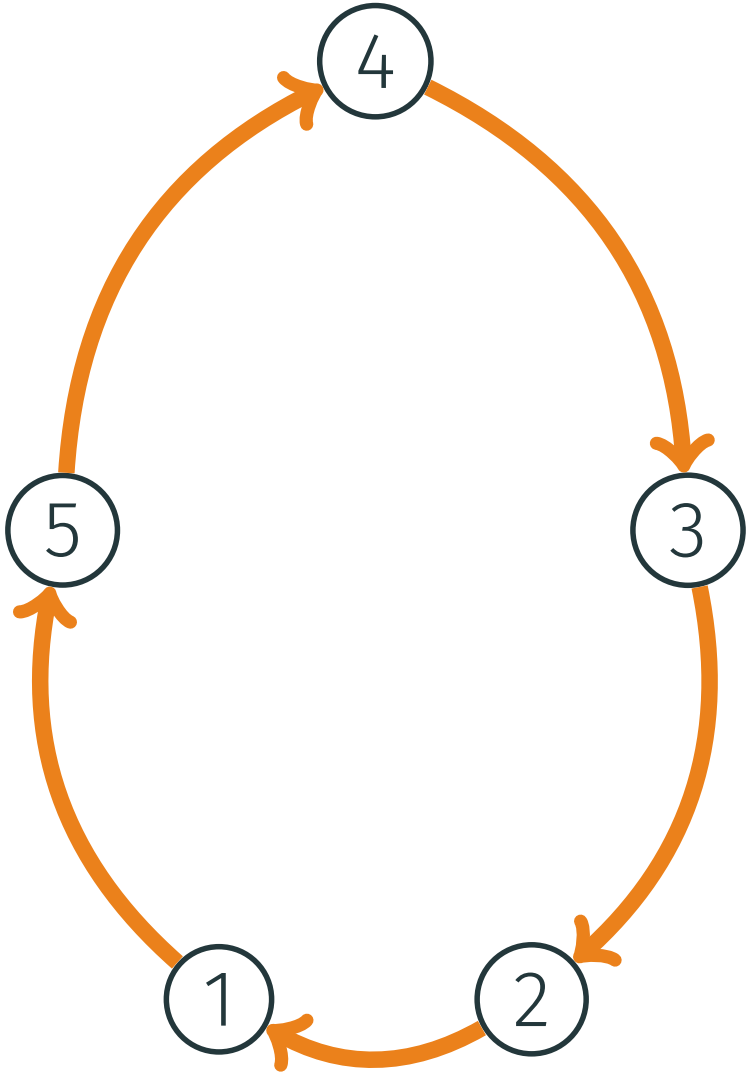
Покрытие ориентированными циклами

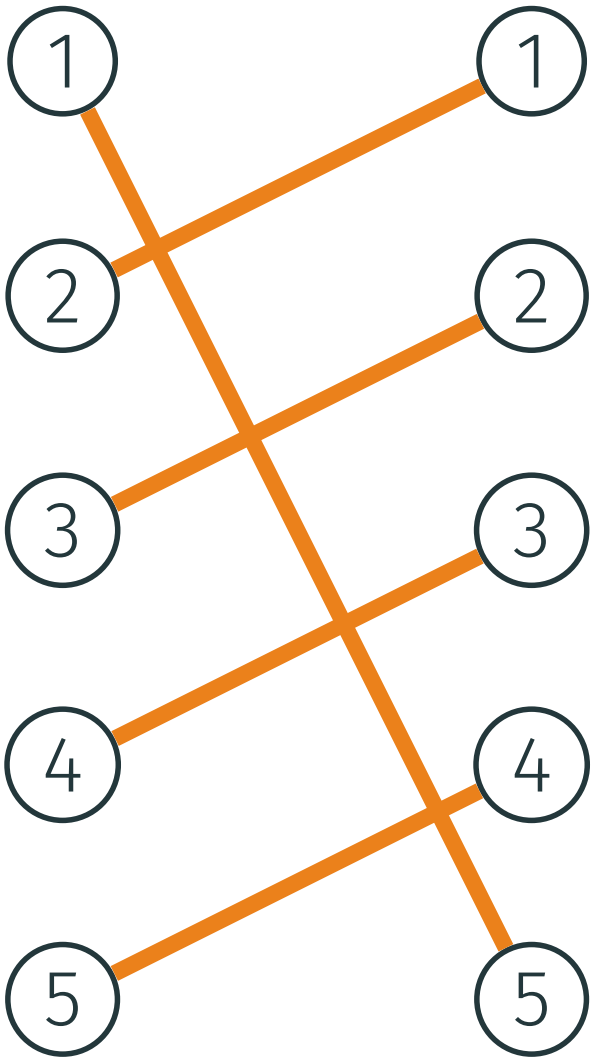






Покрытие ориентированными циклами





1/2-приближение для максимизационной версии

- Найдём покрытие циклами максимального веса. Выкинем из каждого цикла самое лёгкое ребро (при этом суммарный вес уменьшится не более чем вдвое). Соединим полученные пути произвольно.

Обзор непокрытого

- Полиномиальная приближённая схема для евклидова коммивояжёра: для любого $\varepsilon > 0$, существует $(1+\varepsilon)$-приближённый алгоритм со временем работы $n(\log n)^{O(\varepsilon^{-1})}$ (Арора; Митчел, 1997)

Обзор непокрытого

- Полиномиальная приближённая схема для евклидова коммивояжёра: для любого $\varepsilon > 0$, существует $(1+\varepsilon)$-приближённый алгоритм со временем работы $n(\log n)^{O(\varepsilon^{-1})}$ (Арора; Митчел, 1997)
- 2/3- и 3/4-приближение для максимизационной версии (Палук, 2014)

Обзор непокрытого

- Полиномиальная приближённая схема для евклидова коммивояжёра: для любого $\varepsilon > 0$, существует $(1+\varepsilon)$-приближённый алгоритм со временем работы $n(\log n)^{O(\varepsilon^{-1})}$ (Арора; Митчел, 1997)
- 2/3- и 3/4-приближение для максимизационной версии (Палук, 2014)
- 7/5-приближение для графического случая (Себо, Виген, 2012)

Обзор непокрытого

- Полиномиальная приближённая схема для евклидова коммивояжёра: для любого $\varepsilon > 0$, существует $(1+\varepsilon)$-приближённый алгоритм со временем работы $n(\log n)^{O(\varepsilon^{-1})}$ (Арора; Митчел, 1997)
- 2/3- и 3/4-приближение для максимизационной версии (Палук, 2014)
- 7/5-приближение для графического случая (Себо, Виген, 2012)
- $O(1)$-приближение для ориентированных (асимметрических) графов с неравенством треугольника (Свенсон, Тарнавски, Вех, 2017)

Точные алгоритмы

Точные алгоритмы

Динамическое программирование

Динамическое программирование

- Подзадачи: для подмножества городов $S \subseteq \{1, 2, \ldots, n\}$, включающего 1 и j, обозначим через $C[S, j]$ длину кратчайшего пути, начинающегося в 1 и заканчивающегося в j, проходящего через каждый город из множества S ровно один раз.

Динамическое программирование

- Подзадачи: для подмножества городов $S \subseteq \{1, 2, \ldots, n\}$, включающего 1 и j, обозначим через $C[S, j]$ длину кратчайшего пути, начинающегося в 1 и заканчивающегося в j, проходящего через каждый город из множества S ровно один раз.
- Пересчёт: $C[S, j] = \min\limits_{i \in S, i \neq j} \{C[S \setminus \{j\}, i] + d_{ij}\}$.

Code

```
def dp(G):
  n = G.number_of_nodes()
  T = [[float("inf")] * (1 << n) for _ in range(n)]
  T[0][1] = 0
  for s in range(1 << n):
    if sum(((s >> j) & 1) for j in range(n)) <= 1 or not (s & 1):
      continue

    for i in range(1, n):
      if not ((s >> i) & 1):
        continue
      for j in range(n):
        if j == i or not ((s >> j) & 1):
          continue

        T[i][s] = min(T[i][s],
                      T[j][s ^ (1 << i)] + G[i][j]['weight'])

  return min(T[i][(1 << n) - 1] + G[0][i]['weight']
             for i in range(1, n))
```

Динамическое программирование: заключение

- Время работы: $O(n^2 2^n)$

Динамическое программирование: заключение

- Время работы: $O(n^2 2^n)$
- Лучше $n!$, но всё ещё очень медленно (долго уже даже для $n = 20$)

Точные алгоритмы

Формула включений-исключений

Формула включений-исключений

Лемма

Пусть A — некоторое множество,
$f, g\colon 2^A \to \mathbb{R}$, т.ч. $f(X) = \sum_{Y \subseteq X} g(Y)$. Тогда

$$g(X) = \sum_{Y \subseteq X} (-1)^{|X-Y|} f(Y)\,.$$

Доказательство

$$\begin{aligned}\sum_{Y\subseteq X}(-1)^{|X-Y|}f(Y) &= \sum_{Y\subseteq X}\sum_{Z\subseteq Y}(-1)^{|X-Y|}g(Z) = \\ &= \sum_{Z\subseteq X} g(Z) \sum_{Z\subseteq Y\subseteq X}(-1)^{|X-Y|} = g(X)\end{aligned}$$

(последняя сумма равна 1, если $Z = X$, и нулю иначе). $\square$

Задача о гамильтоновом пути

- Проверить, есть ли в данном графе простой путь, проходящий через все вершины, начинающийся в заданной вершине s и заканчивающийся в заданной вершине t

Задача о гамильтоновом пути

- Проверить, есть ли в данном графе простой путь, проходящий через все вершины, начинающийся в заданной вершине s и заканчивающийся в заданной вершине t
- Для $\{s, t\} \subseteq X \subseteq V$, пусть $f(X)$ — количество путей (не обязательно простых! путь может проходить по некоторым вершинам несколько раз, а по некоторым вообще не проходить) длины $n - 1$ из s в t, проходящих только по вершинам из X

Задача о гамильтоновом пути

- Проверить, есть ли в данном графе простой путь, проходящий через все вершины, начинающийся в заданной вершине s и заканчивающийся в заданной вершине t
- Для $\{s, t\} \subseteq X \subseteq V$, пусть $f(X)$ — количество путей (не обязательно простых! путь может проходить по некоторым вершинам несколько раз, а по некоторым вообще не проходить) длины $n - 1$ из s в t, проходящих только по вершинам из X
- $f(X) = A_X[s, t]$, где A_X — матрица смежности графа $G[X]$

Задача о гамильтоновом пути

- Пусть теперь $g(X)$ есть количество путей длины $n-1$ из s в t, проходящих по всем вершинам множества X В частности, $g(V)$ есть количество гамильтоновых путей из s в t.

Задача о гамильтоновом пути

- Пусть теперь $g(X)$ есть количество путей длины $n-1$ из s в t, проходящих по всем вершинам множества X В частности, $g(V)$ есть количество гамильтоновых путей из s в t.
- Тогда
$$g(V) = \sum_{Y \subseteq V} (-1)^{|V-Y|} f(Y)\,.$$

Задача о гамильтоновом пути

- Пусть теперь $g(X)$ есть количество путей длины $n-1$ из s в t, проходящих по всем вершинам множества X В частности, $g(V)$ есть количество гамильтоновых путей из s в t.
- Тогда
$$g(V) = \sum_{Y \subseteq V} (-1)^{|V-Y|} f(Y)\,.$$
- Таким образом, количество гамильтоновых путей в графе может быть найдено за время $O^*(2^n)$ и полиномиальную память

Задача о гамильтоновом пути

- Пусть теперь $g(X)$ есть количество путей длины $n-1$ из s в t, проходящих по всем вершинам множества X В частности, $g(V)$ есть количество гамильтоновых путей из s в t.
- Тогда
$$g(V) = \sum_{Y \subseteq V} (-1)^{|V-Y|} f(Y)\,.$$
- Таким образом, количество гамильтоновых путей в графе может быть найдено за время $O^*(2^n)$ и полиномиальную память
- Данный алгоритм переизобретался три раза

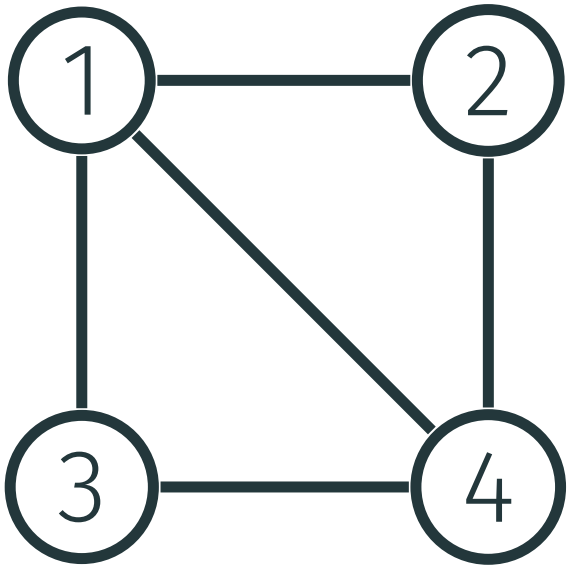
Точные алгоритмы

Матрица Татта и перманент

Замечание

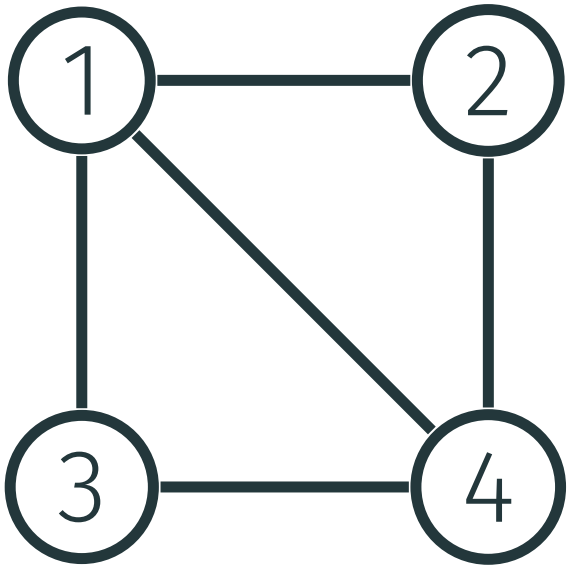
Далее мы рассмотрим алгоритм Бьорклунда для решения задачи о гамильтоновом цикле в двудольном *неориентированном* графе за время $O^*(2^{n/2})$. Для общего случая задачи коммивояжёра оценка на время работы алгоритма Бьорклунда составляет $O^*(1.657^n \cdot W)$ (W — максимальный вес ребра).

Перманент матрицы Татта



$$
\begin{matrix}
 & x_{12} & x_{13} & x_{14} \\
x_{12} & & & x_{24} \\
x_{13} & & & x_{34} \\
x_{14} & x_{24} & x_{34} &
\end{matrix}
$$

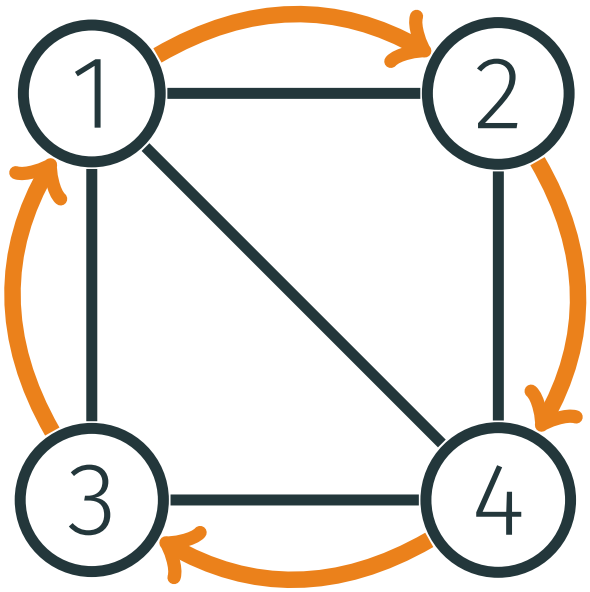
Перманент матрицы Татта



$$\begin{matrix} & x_{12} & x_{13} & x_{14} \\ x_{12} & & & x_{24} \\ x_{13} & & & x_{34} \\ x_{14} & x_{24} & x_{34} & \end{matrix}$$

$$\mathrm{perm}(M) =$$

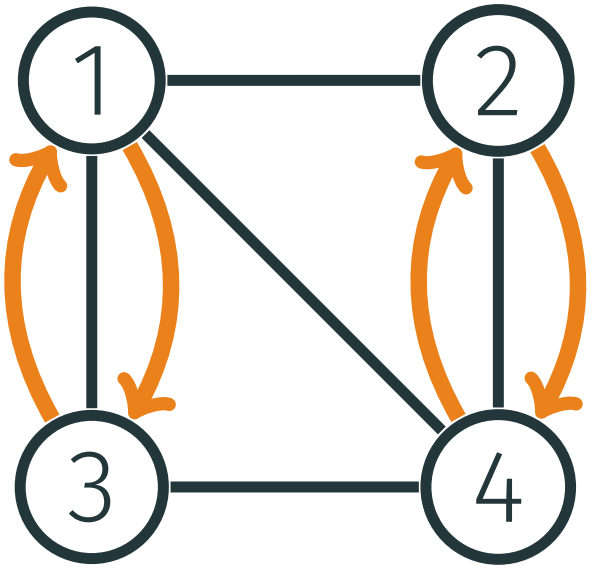
Перманент матрицы Татта



$$\begin{matrix} & x_{12} & x_{13} & x_{14} \\ x_{12} & & & x_{24} \\ x_{13} & & & x_{34} \\ x_{14} & x_{24} & x_{34} & \end{matrix}$$

$$\mathrm{perm}(M) = x_{12}x_{24}x_{43}x_{31} + \ldots$$

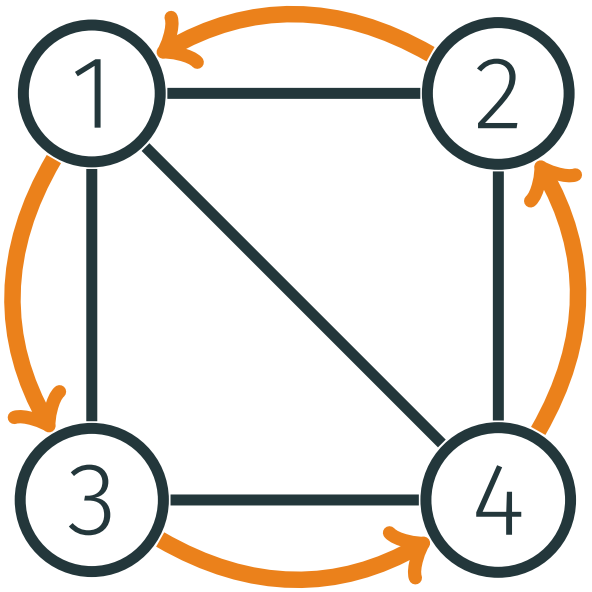
Перманент матрицы Татта



$$\begin{matrix} & x_{12} & x_{13} & x_{14} \\ x_{12} & & & x_{24} \\ x_{13} & & & x_{34} \\ x_{14} & x_{24} & x_{34} & \end{matrix}$$

$$\mathrm{perm}(M) = x_{12}x_{24}x_{43}x_{31} + x_{13}^2x_{24}^2 + \dots$$

Перманент матрицы Татта



$$\begin{matrix} & x_{12} & x_{13} & x_{14} \\ x_{12} & & & x_{24} \\ x_{13} & & & x_{34} \\ x_{14} & x_{24} & x_{34} & \end{matrix}$$

$$\mathrm{perm}(M) = x_{12}x_{24}x_{43}x_{31} + x_{13}^2x_{24}^2 + x_{12}x_{24}x_{43}x_{31} + \ldots$$

Поле характеристики два

- Если вычислять перманент над полем характеристики 2, то все циклы, не полностью состоящие из циклов длины 2, сократятся: если в покрытии циклами есть цикл длины не 2, то возьмём первый из них (относительного какого-нибудь порядка на вершинах) и обратим в нём все рёбра. Получим другое покрытие циклами, которому соответствует тот же самый моном

Поле характеристики два

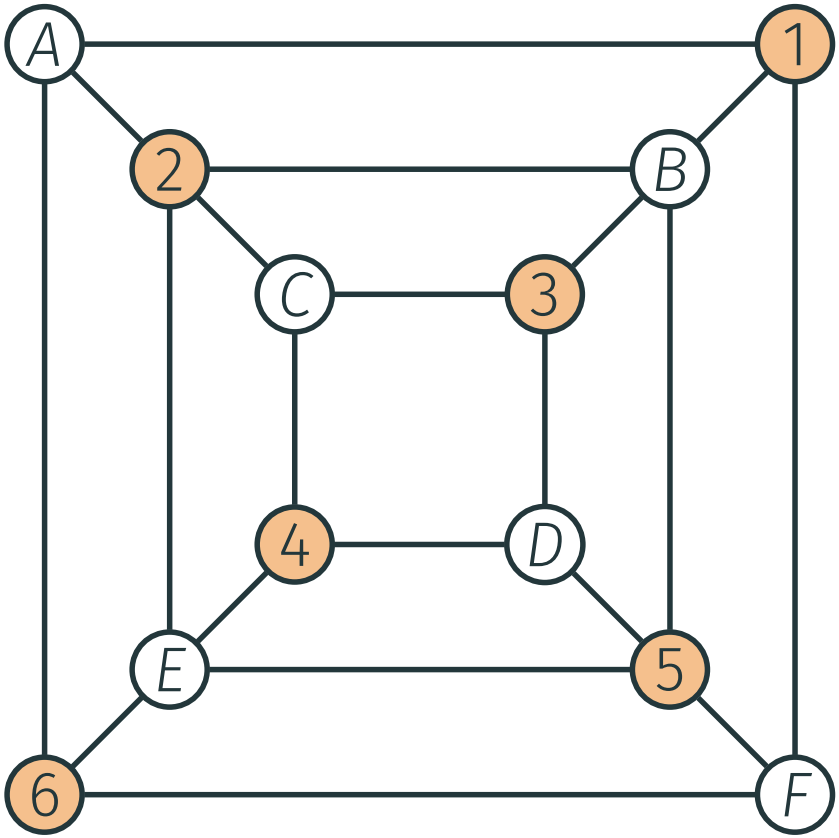
- Если вычислять перманент над полем характеристики 2, то все циклы, не полностью состоящие из циклов длины 2, сократятся: если в покрытии циклами есть цикл длины не 2, то возьмём первый из них (относительного какого-нибудь порядка на вершинах) и обратим в нём все рёбра. Получим другое покрытие циклами, которому соответствует тот же самый моном
- Мы хотим исправить следующие два момента: во-первых, чтобы гамильтоновы циклы не сокращались, а во-вторых, чтобы покрытия с циклами длины 2 всё же пропадали

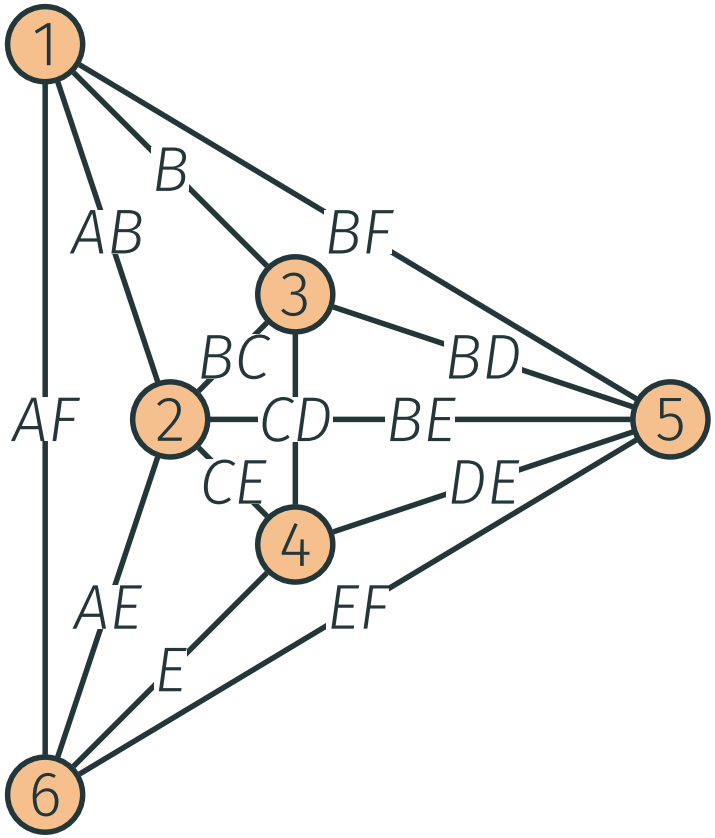
Первая цель: оставить гам. циклы

Сделаем вершину 1 графа выделенной: $T_G[1, j] = x_{1j}$, но $T_G[j, 1] = x_{j1}$ для ребра $\{1, j\} \in E$. Тогда каждому гамильтонову циклу будут соответствовать два разных монома.

$$
\begin{array}{cccc}
 & x_{12} & x_{13} & x_{14} \\
x_{21} & & & x_{24} \\
x_{31} & & & x_{34} \\
x_{41} & x_{24} & x_{34} &
\end{array}
$$

Вторая цель: сократить всё остальное





Почему же всё сократится?

- В новом графе нам нужен помеченный гамильтонов цикл

Почему же всё сократится?

- В новом графе нам нужен помеченный гамильтонов цикл
- В матрице Татта теперь будут помеченные переменные: вместо x_{24} будет $x_{24,C} + x_{24,E}$

Почему же всё сократится?

- В новом графе нам нужен помеченный гамильтонов цикл
- В матрице Татта теперь будут помеченные переменные: вместо x_{24} будет $x_{24,C} + x_{24,E}$
- При вычислении над полем характеристики 2 гамильтоновы циклы по-прежнему не сократятся (из-за специальной переменной 1)

Всё остальное

- покрытия циклами, в которых используются не все пометки — сократятся по формуле включений-исключений (рассмотрим все $2^{n/2}$ подмножеств пометок)

Всё остальное

- покрытия циклами, в которых используются не все пометки — сократятся по формуле включений-исключений (рассмотрим все $2^{n/2}$ подмножеств пометок)
- негамилтьтоновы покрытия циклами, в которых есть все пометки, разобьются на пары и сократятся (из-за пометок)

Получившийся алгоритм

- Рассмотрим многочлен

$$P(\bar{x}) = \sum_{S \subseteq L} \operatorname{perm}(M_S)$$

Получившийся алгоритм

- Рассмотрим многочлен

$$P(\bar{x}) = \sum_{S \subseteq L} \operatorname{perm}(M_S)$$

- Как мы уже выяснили, $P \not\equiv 0$ тогда и только тогда, когда в графе есть гамильтонов цикл

Получившийся алгоритм

- Рассмотрим многочлен

$$P(\bar{x}) = \sum_{S \subseteq L} \operatorname{perm}(M_S)$$

- Как мы уже выяснили, $P \not\equiv 0$ тогда и только тогда, когда в графе есть гамильтонов цикл
- Проверять, равен он нулю или нет, мы будем вероятностно: подставим случайные значения всем переменным; если получился не ноль, то и многочлен точно не равен нулю; в противном случае мы могли попасть в корень

Проверка равенства нулю многочлена

Лемма Шварца–Зиппеля

Пусть P — ненулевой многочлен полной степени d над полем $\mathbb{F}$ и пусть $S \subseteq \mathbb{F}$. Тогда

$$\Pr_{r_1,\ldots,r_n \in S}\{P(r_1,\ldots,r_n) = 0\} \leq d/|S|\,.$$

Непокрытое: обзор

- k-путь: $O^*(1.66^k)$ в неориентированных графах, $O^*(2^k)$ в ориентированных (Бьорклунд, 2010)
- $O^*(3^{n/2})$ в двудольных ориентированных графах ($O^*(3^{n-\alpha(G)})$) (Бьорклунд, Каски, Кутис, 2017)
- $O^*(1.619^n)$ для вычисления чётности числа гамильтоновых циклов в ориентированном графе (Бьорклунд, Хусфельд, 2013)

Коммивояжёр: открытые задачи!

- Постройте $O^*(1.99^n)$-точный алгоритм для ориентированного гамильтонова пути!
- Постройте 1.499-приближённый алгоритм для коммивояжёра в метрическом пространстве!
- Постройте детереминированный $O^*(2^k)$-точный алгоритм для k-пути!
- Постройте 0.76-приближённый алгоритм для задачи максимального коммивояжёра!

Надстрока: открытые задачи!

- Постройте $O^*(1.99^n)$-точный алгоритм для задачи о надстроке!
- Докажите жадную гипотезу для задачи о надстроке!
- Постройте 2,3-приближённый алгоритм для задачи о надстроке!
- Сводится ли задача коммивояжёра на графе с n вершинами к задаче о надстроке с $O(n)$ строками (разумной длины)?