Мини-задача #48 (2 балла, дополнительная)

Реализовать наивный перебор для TSP. Проверить, на каком размере графа он все еще работает.

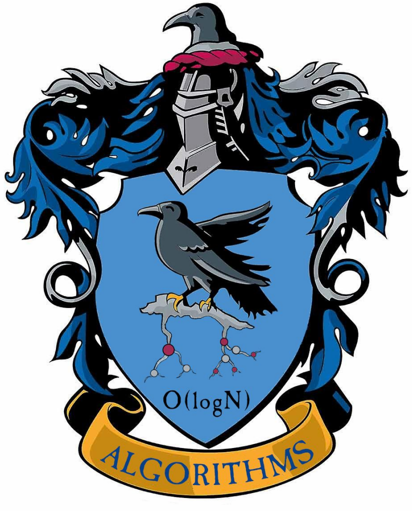
Реализовать алгоритм Беллмана-Хелда-Карпа для решения задачи TSP.

Подумайте об удобном кодировании множеств S (и используйте его в решении).

Сравнить скорость решения с наивным перебором (на графах, где наивный еще работает за разумное время).

Алгоритмы и структуры данных

TSP, классы сложности P и NP

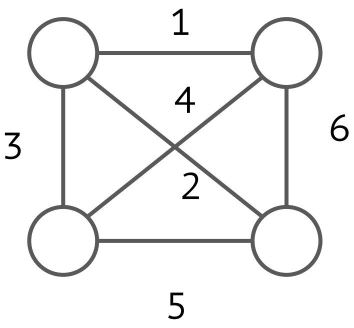


Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.

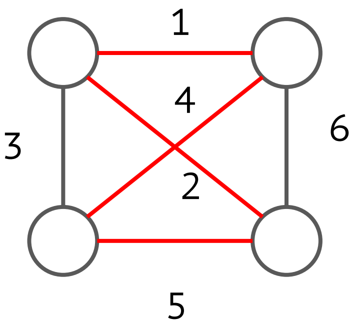
Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.



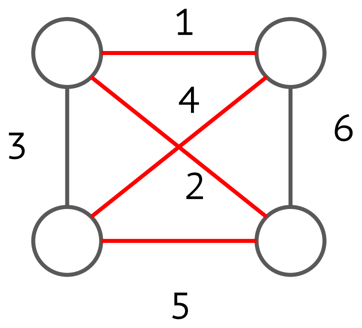
Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.



Задача коммивояжера a.k.a TSP

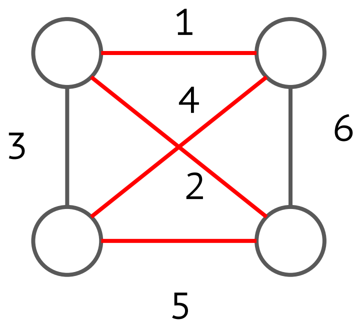
Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.



Изначальная легенда - поиск кратчайшего маршрута для странствующего торговца (travelling salesman, коммивояжера)

Задача коммивояжера a.k.a TSP

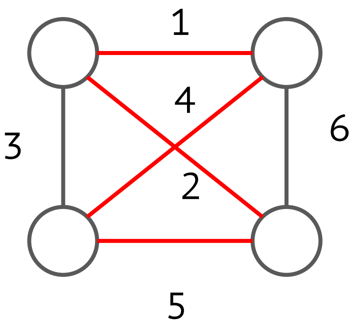
Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.



Как решать?

Задача коммивояжера a.k.a TSP

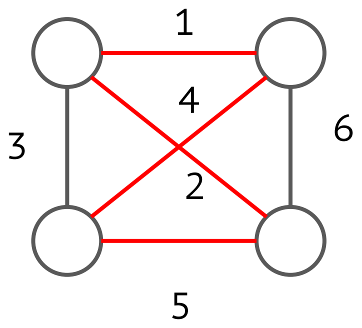
Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.



Как решать? Можно перебором. Худший случай?

Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.

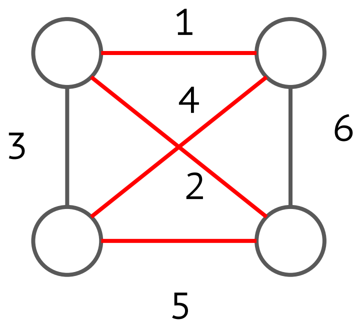


Как решать? Можно перебором. Худший случай?

Полный граф! Сколько в нем есть простых циклов?

Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.

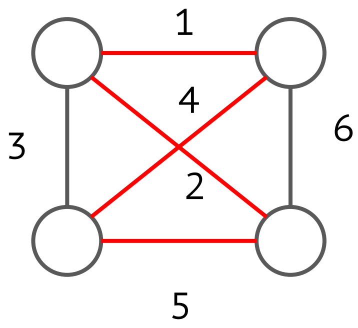


Как решать? Можно перебором. Худший случай?

Полный граф! Сколько в нем есть гамильтоновых циклов? В полном графе есть любые пути, значит каждый цикл можно задать перестановкой вершин (их n!)

Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.

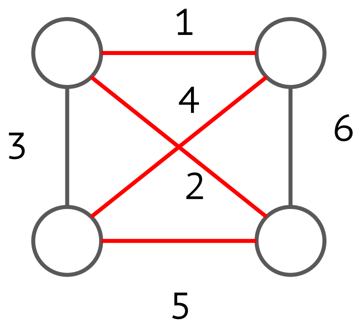


Как решать? Можно перебором. Худший случай?

Полный граф! Сколько в нем есть гамильтоновых циклов? В полном графе есть любые пути, значит каждый цикл можно задать перестановкой вершин (их n!), но нужно выкинуть повторения: пути в обе стороны (т.е. делим на 2) и пути, начинающиеся с разных вершин (т.е. делим на n)

Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.

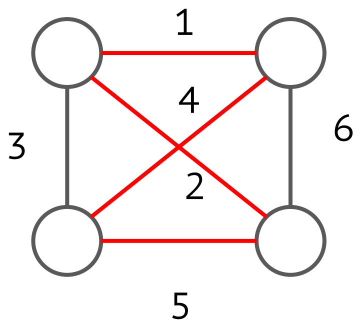


Как решать? Можно перебором. Худший случай?

Полный граф! Сколько в нем есть гамильтоновых циклов? В полном графе есть любые пути, значит каждый цикл можно задать перестановкой вершин (их n!), но нужно выкинуть повторения: пути в обе стороны (т.е. делим на 2) и пути, начинающиеся с разных вершин (т.е. делим на n). Итого: $O(\frac{n!}{2n}) = O(\frac{1}{2}(n-1)!) = O(n!)$

Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.

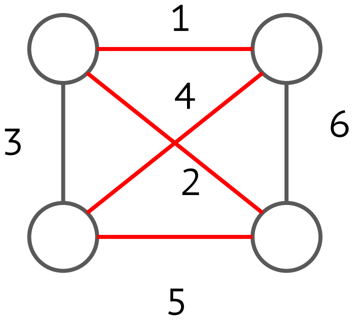


Как решать?

Можно перебором. За $O(\frac{n!}{2n}) = O(\frac{1}{2}(n-1)!) = O(n!)$

Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.



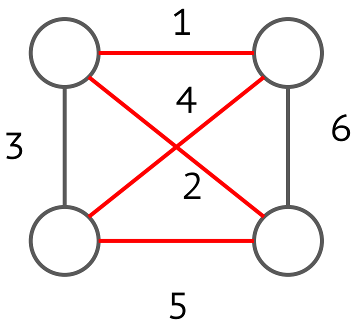
Как решать?

Можно перебором. За $O(\frac{n!}{2n}) = O(\frac{1}{2}(n-1)!) = O(n!)$

Можем ли мы лучше?

Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.



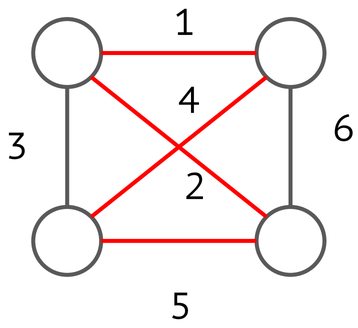
Как решать?

Можно перебором. За $O(\frac{n!}{2n}) = O(\frac{1}{2}(n-1)!) = O(n!)$

Можем ли мы лучше?
На что вообще похоже?

Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.



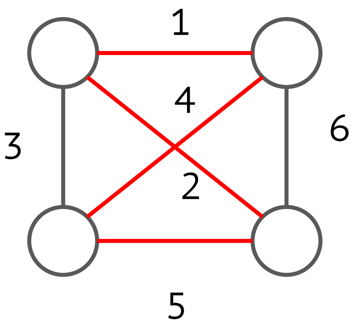
Как решать?

Можно перебором. За $O(\frac{n!}{2n}) = O(\frac{1}{2}(n-1)!) = O(n!)$

Можем ли мы лучше?
На что вообще похоже?
На поиск кратчайших путей (привет, Беллман-Форд!)

Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.



Как решать?

Можно перебором. За $O(\frac{n!}{2n}) = O(\frac{1}{2}(n-1)!) = O(n!)$

Можем ли мы лучше?
На что вообще похоже?
На поиск кратчайших путей (привет, Беллман-Форд!)
Пробуем динамическое программирование.

Оптимальная подструктура (Беллман-Форд)

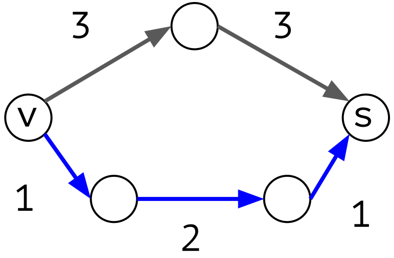
Какую выбрать подструктуру в графовой задаче? В отличие от прошлых задач это не так очевидно.

Например, выкидывать последнее из ребер из оптимального решения нельзя - полученное решение приведет вас уже в другую вершину!

Вместо этого будем ограничивать количество ребер, которые можно использовать в пути.

1) Если можно брать максимум 2 ребра, то ответ - 6

2) Если можно брать 3 ребра, то 4



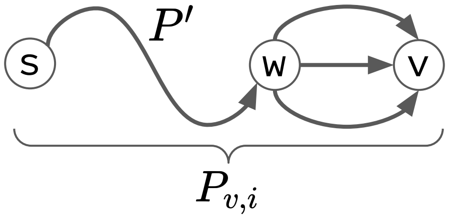
Оптимальная подструктура (Беллман-Форд)

Пусть дан взвешенный граф $G = \langle V, E \rangle$ и вершина **s**, из которой будем искать кратчайшие пути в остальные вершины.

Для каждой вершины $v \in V$ и $i \in \{1, 2, 3, \ldots\}$ введем $P_{v,i}$ - кратчайший путь от s до v, который использует не больше, чем i ребер.

1-ый случай: в $P_{v,i}$ оказалось $\leq (i-1)$ ребер. Тогда: $P_{v,i} = P_{v,i-1}$ 🎉

2-ой случай: в $P_{v,i}$ используется ровно **i** ребер. Тогда $P' = P_{w,i-1}$.



$P_{v,i} = P_{w,i-1} + (w, v)$, где (w, v) - самое короткое ребро между w и v 🎉

Рекуррентное соотношение (Беллман-Форд)

Введем $L_{v,i}$- длину кратчайшего пути из **s** в **v**, использующего не больше, чем **i** ребер ($+\infty$, если такого пути нет).

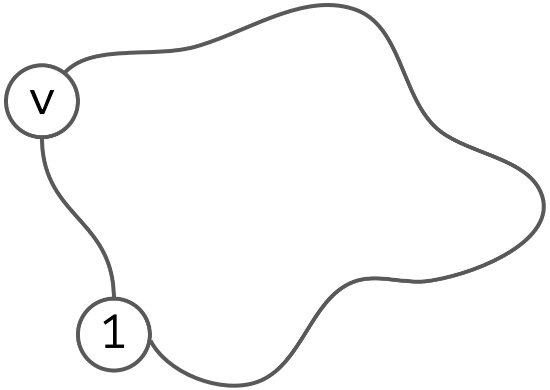
Тогда для всех $v \in V$ и $i \in \{1, 2, 3, \ldots\}$ верно:

$$L_{v,i} = \min \begin{cases} L_{v,i-1} \\ \min\limits_{(w,v) \in E} L_{w,i-1} + c_{w,v} \end{cases}$$

Здесь $c_{w,v}$ - вес ребра (w, v)

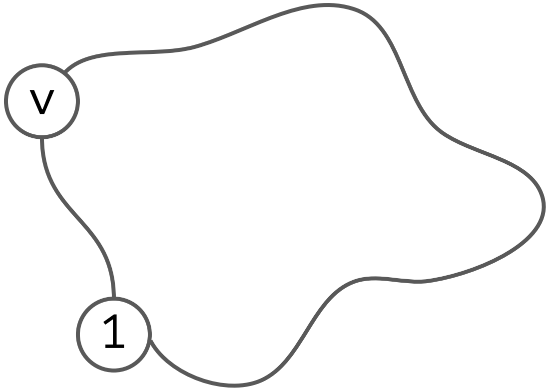
Оптимальная подструктура (TSP)

Оптимальная подструктура (TSP)



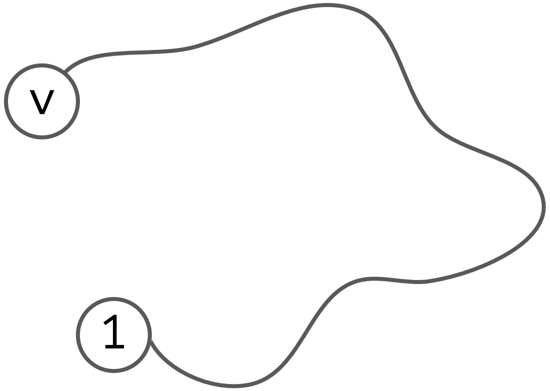
Допустим, что у нас кратчайший Гамильтонов цикл из вершины 1 в вершину 1.

Оптимальная подструктура (TSP)



Допустим, что у нас кратчайший Гамильтонов цикл из вершины 1 в вершину 1. И пусть v - последняя вершина в этом пути.

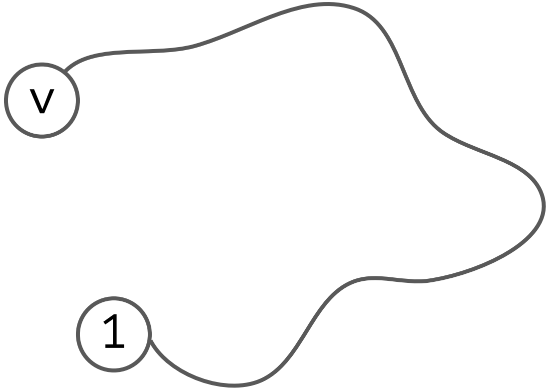
Оптимальная подструктура (TSP)



Допустим, что у нас кратчайший Гамильтонов цикл из вершины 1 в вершину 1. И пусть v - последняя вершина в этом пути.

Тогда наша задача очевидно сводится к поиску кратчайшего пути из 1 в v…

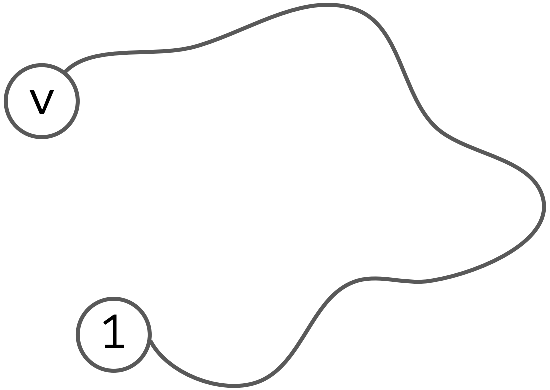
Оптимальная подструктура (TSP)



Допустим, что у нас кратчайший Гамильтонов цикл из вершины 1 в вершину 1. И пусть v - последняя вершина в этом пути.

Тогда наша задача очевидно сводится к поиску кратчайшего пути из 1 в v, но при этом: на этом пути не должно быть повторяющихся вершин!

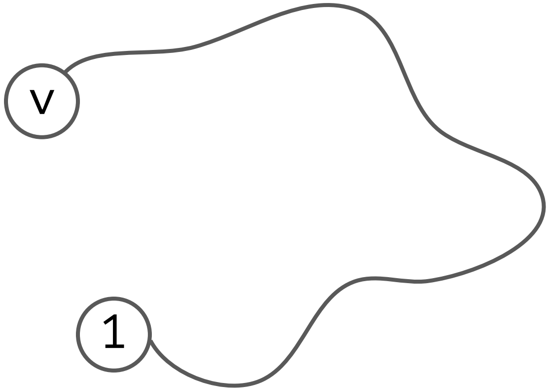
Оптимальная подструктура (TSP)



Допустим, что у нас кратчайший Гамильтонов цикл из вершины 1 в вершину 1. И пусть v - последняя вершина в этом пути.

Тогда наша задача очевидно сводится к поиску кратчайшего пути из 1 в v, но при этом: на этом пути не должно быть повторяющихся вершин + он должен содержать все оставшиеся вершины.

Оптимальная подструктура (TSP)



Допустим, что у нас кратчайший Гамильтонов цикл из вершины 1 в вершину 1. И пусть v - последняя вершина в этом пути.

Тогда наша задача очевидно сводится к поиску кратчайшего пути из 1 в v, но при этом: на этом пути не должно быть повторяющихся вершин + он должен содержать все оставшиеся вершины.

С этими ограничениями попробуем применить оптимальную подструктуру из Беллмана-Форда.

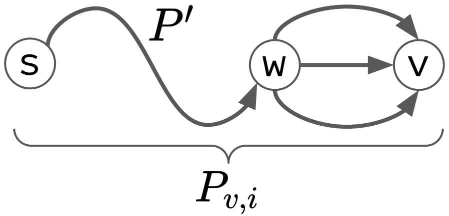
Оптимальная подструктура (Беллман-Форд)

Пусть дан взвешенный граф $G = \langle V, E \rangle$ и вершина **s**, из которой будем искать кратчайшие пути в остальные вершины.

Для каждой вершины $v \in V$ и $i \in \{1, 2, 3, \ldots\}$ введем $P_{v,i}$ - кратчайший путь от s до v, который использует не больше, чем i ребер.

1-ый случай: в $P_{v,i}$ оказалось $\leq (i-1)$ ребер. Тогда: $P_{v,i} = P_{v,i-1}$ 🎉

2-ой случай: в $P_{v,i}$ используется ровно **i** ребер. Тогда $P' = P_{w,i-1}$.



$P_{v,i} = P_{w,i-1} + (w, v)$, где (w, v) - самое короткое ребро между w и v 🎉

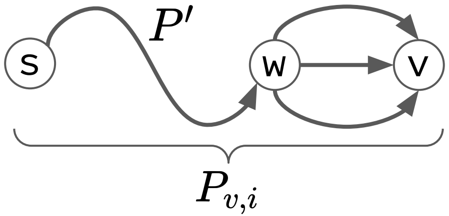
Оптимальная подструктура (Беллман-Форд)

Пусть дан взвешенный граф $G = \langle V, E \rangle$ и вершина **s**, из которой будем искать кратчайшие пути в остальные вершины.

Для каждой вершины $v \in V$ и $i \in \{1, 2, 3, \ldots\}$ введем $P_{v,i}$ - кратчайший путь от s до v, который использует не больше, чем i ребер.

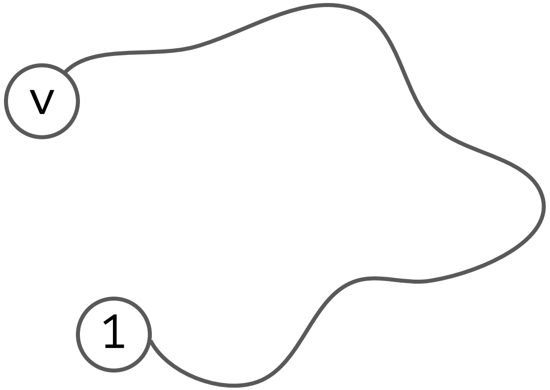
~~1-ый случай: в $P_{v,i}$ оказалось $\leq (i-1)$ ребер. Тогда: $P_{v,i} = P_{v,i-1}$~~ 🎉

2-ой случай: в $P_{v,i}$ используется ровно **i** ребер. Тогда $P' = P_{w,i-1}$.



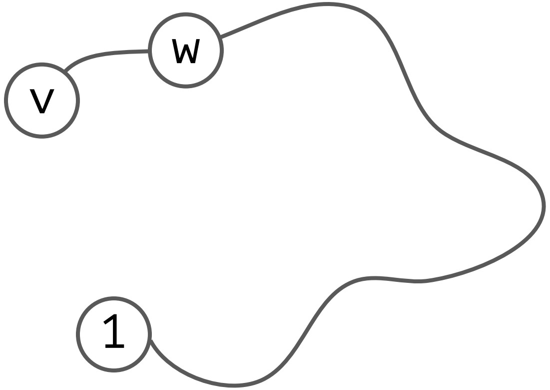
$P_{v,i} = P_{w,i-1} + (w, v)$, где (w, v) - самое короткое ребро между w и v 🎉

Оптимальная подструктура (TSP)



Для каждой вершины $v \in V$ и $i \in \{2, 3, \dots\}$ введем $P_{v,i}$ - кратчайший путь от 1 до v, который использует ровно i ребер и не содержит повторяющихся вершин.

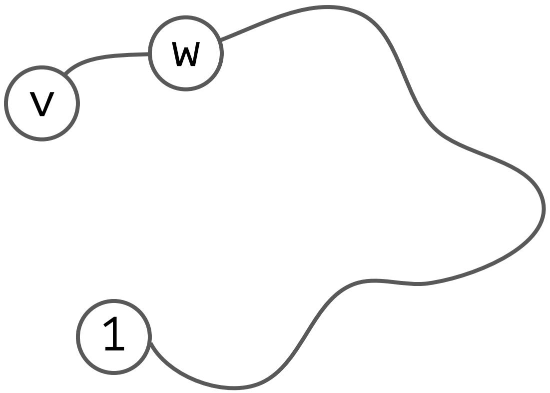
Оптимальная подструктура (TSP)



Для каждой вершины $v \in V$ и $i \in \{2, 3, \ldots\}$ введем $P_{v,i}$ - кратчайший путь от 1 до v, который использует ровно i ребер и не содержит повторяющихся вершин.

Как соотносится с $P_{w,i-1}$?

Оптимальная подструктура (TSP)

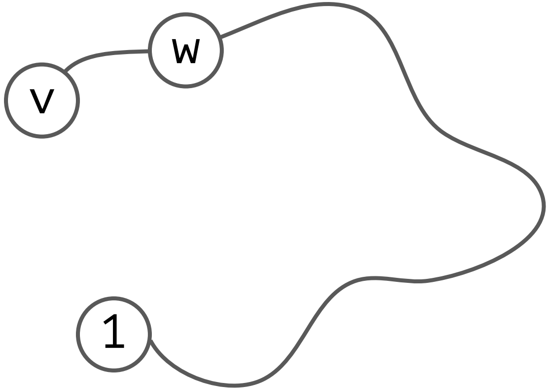


Для каждой вершины $v \in V$ и $i \in \{2, 3, \dots\}$ введем $P_{v,i}$ - кратчайший путь от 1 до v, который использует ровно i ребер и не содержит повторяющихся вершин.

Как соотносится с $P_{w,i-1}$?

Справедливо ли, что $P_{v,i} = P_{w,i-1} + (w, v)$? (где (w, v) - кратчайшее ребро м/у w и v)

Оптимальная подструктура (TSP)



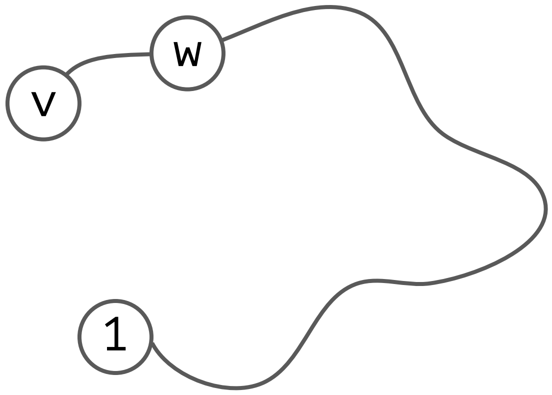
Для каждой вершины $v \in V$ и $i \in \{2, 3, \ldots\}$ введем $P_{v,i}$ - кратчайший путь от 1 до v, который использует ровно i ребер и не содержит повторяющихся вершин.

Как соотносится с $P_{w,i-1}$?

Справедливо ли, что $P_{v,i} = P_{w,i-1} + (w, v)$? (где (w, v) - кратчайшее ребро м/у w и v)

Если да, то мы победили, сейчас выпишем рекуррентное соотношение и ответ!

Оптимальная подструктура (TSP)



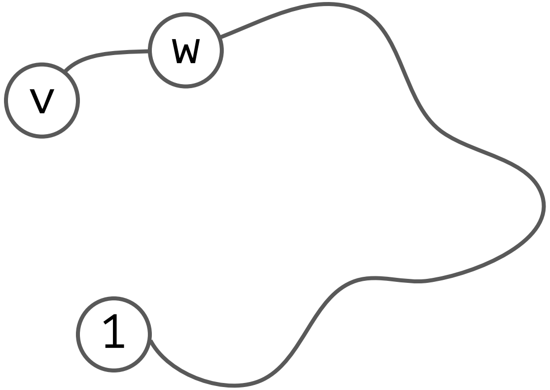
Для каждой вершины $v \in V$ и $i \in \{2, 3, \ldots\}$ введем $P_{v,i}$ - кратчайший путь от 1 до v, который использует ровно i ребер и не содержит повторяющихся вершин.

Как соотносится с $P_{w,i-1}$?

Справедливо ли, что $P_{v,i} = P_{w,i-1} + (w, v)$? (где (w, v) - кратчайшее ребро м/у w и v)

Если да, то мы победили, сейчас выпишем рекуррентное соотношение и ответ! А заодно решим одну из задач тысячелетия

Оптимальная подструктура (TSP)

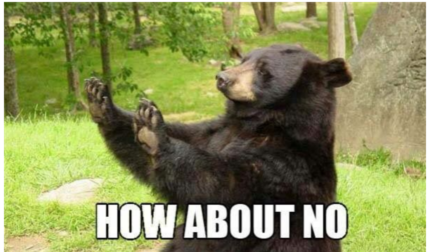


Для каждой вершины $v \in V$ и $i \in \{2, 3, \ldots\}$ введем $P_{v,i}$ - кратчайший путь от 1 до v, который использует ровно i ребер и не содержит повторяющихся вершин.

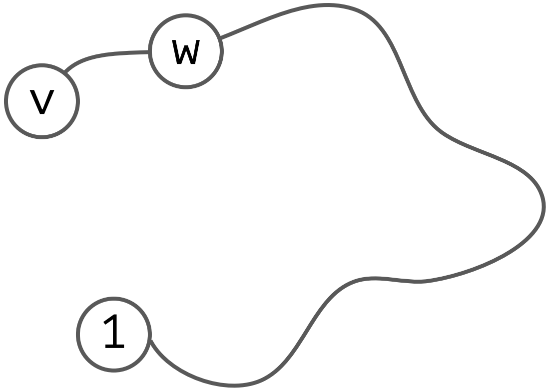
Как соотносится с $P_{w,i-1}$?

Справедливо ли, что $P_{v,i} = P_{w,i-1} + (w, v)$? (где (w, v) - кратчайшее ребро м/у w и v)

НЕТ.



Оптимальная подструктура (TSP)

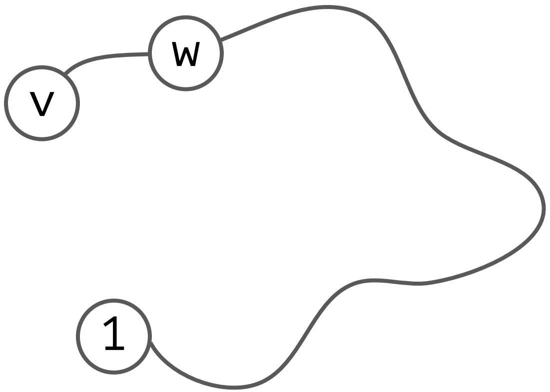


Для каждой вершины $v \in V$ и $i \in \{2, 3, \dots\}$ введем $P_{v,i}$ - кратчайший путь от 1 до v, который использует ровно i ребер и не содержит повторяющихся вершин.

Как соотносится с $P_{w,i-1}$?
Справедливо ли, что $P_{v,i} = P_{w,i-1} + (w, v)$?
(где (w, v) - кратчайшее ребро м/у w и v)

Нет. Проблема в том, что $P_{w,i-1}$ "кратчайший путь из 1 в w, содержащий ровно i - 1 ребер и не содержащий повторяющихся вершин" вполне может содержать вершину v!

Оптимальная подструктура (TSP)

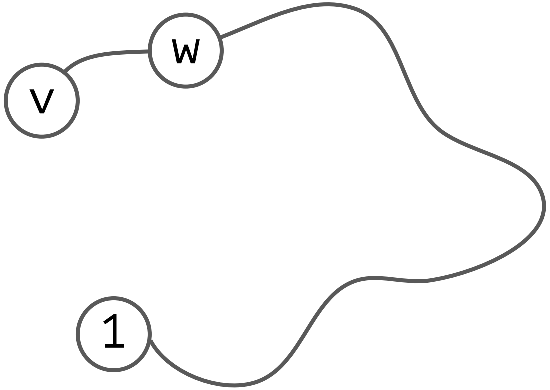


Для каждой вершины $v \in V$ и $i \in \{2, 3, \dots\}$ введем $P_{v,i}$ - кратчайший путь от 1 до v, который использует ровно i ребер и не содержит повторяющихся вершин.

Как соотносится с $P_{w,i-1}$?
Справедливо ли, что $P_{v,i} = P_{w,i-1} + (w, v)$?
(где (w, v) - кратчайшее ребро м/у w и v)

Нет. Проблема в том, что $P_{w,i-1}$ "кратчайший путь из 1 в w, содержащий ровно i - 1 ребер и не содержащий повторяющихся вершин" вполне может содержать вершину v! А тогда мы не имеем права строить $P_{v,i}$ через добавление $P_{w,i-1}$, ведь мы потеряем свойство уникальности вершин!

Оптимальная подструктура (TSP)



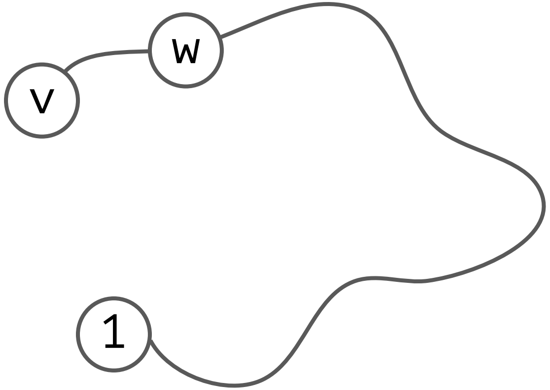
Для каждой вершины $v \in V$ и $i \in \{2, 3, \dots\}$ введем $P_{v,i}$ - кратчайший путь от 1 до v, который использует ровно i ребер и не содержит повторяющихся вершин.

Как соотносится с $P_{w,i-1}$?
Справедливо ли, что $P_{v,i} = P_{w,i-1} + (w, v)$?
(где (w, v) - кратчайшее ребро м/у w и v)

Нет. Проблема в том, что $P_{w,i-1}$ "кратчайший путь из 1 в w, содержащий ровно i - 1 ребер и не содержащий повторяющихся вершин" вполне может содержать вершину v! А тогда мы не имеем права строить $P_{v,i}$ через добавление $P_{w,i-1}$, ведь мы потеряем свойство уникальности вершин!

Как это исправить?

Оптимальная подструктура (TSP)



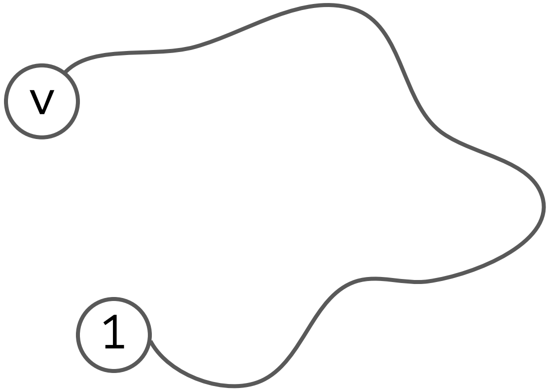
Для каждой вершины $v \in V$ и $i \in \{2, 3, \dots\}$ введем $P_{v,i}$ - кратчайший путь от 1 до v, который использует ровно i ребер и не содержит повторяющихся вершин.

Как соотносится с $P_{w,i-1}$?
Справедливо ли, что $P_{v,i} = P_{w,i-1} + (w, v)$?
(где (w, v) - кратчайшее ребро м/у w и v)

Нет. Проблема в том, что $P_{w,i-1}$ "кратчайший путь из 1 в w, содержащий ровно i - 1 ребер и не содержащий повторяющихся вершин" вполне может содержать вершину v! А тогда мы не имеем права строить $P_{v,i}$ через добавление $P_{w,i-1}$, ведь мы потеряем свойство уникальности вершин!

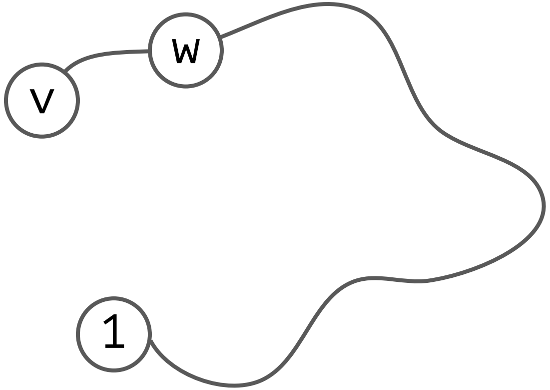
Как это исправить? Длины недостаточно, нужно думать про весь путь.

Оптимальная подструктура (TSP)



Для каждой вершины $v \in V$ и $S \subseteq \{1, 2, \dots, n\}$ причем $1 \in S$ и $v \in S$, введем $P_{v,S}$ - кратчайший путь из 1 в v, содержащий в точности все вершины из S (ровно по одному разу)

Оптимальная подструктура (TSP)

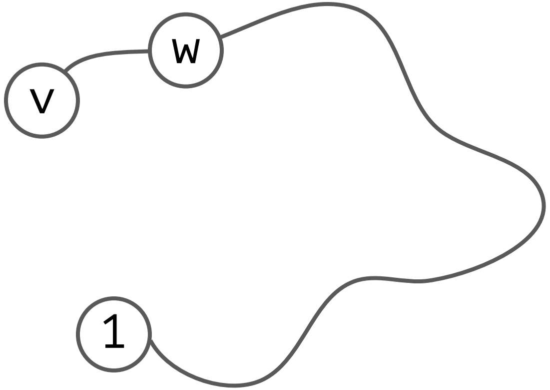


Для каждой вершины $v \in V$ и $S \subseteq \{1, 2, \ldots, n\}$ причем $1 \in S$ и $v \in S$, введем $P_{v,S}$ - кратчайший путь из 1 в v, содержащий в точности все вершины из S (ровно по одному разу)

Тогда пусть в таком пути последняя вершина до v была w.

Как выразить $P_{v,S}$ через меньшую размерность?

Оптимальная подструктура (TSP)



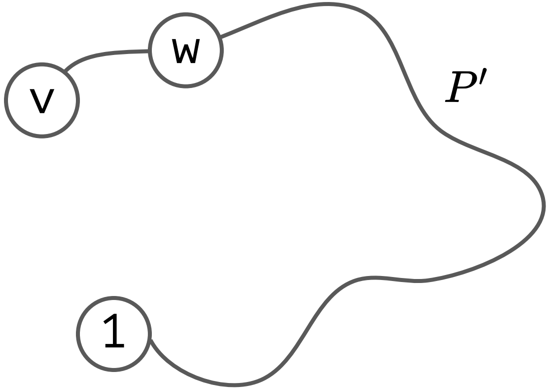
Для каждой вершины $v \in V$ и $S \subseteq \{1, 2, \ldots, n\}$ причем $1 \in S$ и $v \in S$, введем $P_{v,S}$ - кратчайший путь из 1 в v, содержащий в точности все вершины из S (ровно по одному разу)

Тогда пусть в таком пути последняя вершина до v была w.

Как выразить $P_{v,S}$ через меньшую размерность?

$$P_{v,S} = P_{w,S-\{v\}} + (w, v)$$

Оптимальная подструктура (TSP)



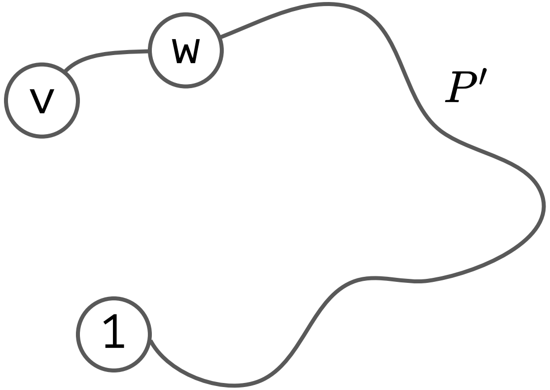
Для каждой вершины $v \in V$ и $S \subseteq \{1, 2, \ldots, n\}$ причем $1 \in S$ и $v \in S$, введем $P_{v,S}$ - кратчайший путь из 1 в v, содержащий в точности все вершины из S (ровно по одному разу)

Тогда пусть в таком пути последняя вершина до v была w.

Как выразить $P_{v,S}$ через меньшую размерность?

$P_{v,S} = P_{w,S-\{v\}} + (w, v)$ => больше не имеем права использовать v в P' 🎉

Оптимальная подструктура (TSP)



Для каждой вершины $v \in V$ и $S \subseteq \{1, 2, \ldots, n\}$ причем $1 \in S$ и $v \in S$, введем $P_{v,S}$ - кратчайший путь из 1 в v, содержащий в точности все вершины из S (ровно по одному разу)

Тогда пусть в таком пути последняя вершина до v была w.

Как выразить $P_{v,S}$ через меньшую размерность?

$P_{v,S} = P_{w,S-\{v\}} + (w, v)$ => больше не имеем права использовать v в P' 🎉

Т.е. выражаем ответ для |S| == i через множества |S'| == i - 1

Алгоритм динамики для TSP

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Алгоритм динамики для TSP

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Размерность такого массива?

Алгоритм динамики для TSP

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Размерность такого массива? $2^{n-1} * n$

Алгоритм динамики для TSP

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

Алгоритм динамики для TSP

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

for m in [2, n]:
  foreach $S \subseteq \{1, 2, \ldots, n\} : |S| = m, 1 \in S$ do:

Алгоритм динамики для TSP

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

```
for m in [2, n]:
  foreach S ⊆ {1, 2, ..., n} : |S| = m, 1 ∈ S do:
    for v in S, v != 1:
```

Алгоритм динамики для TSP

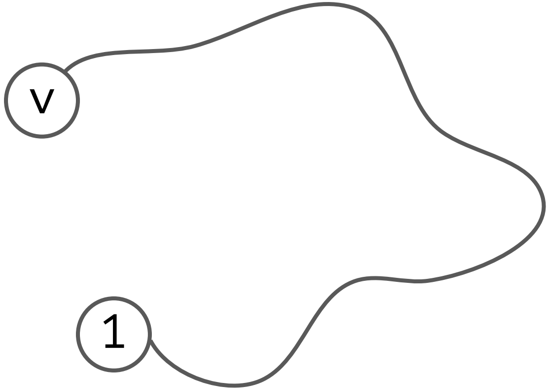
Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \dots, n\}$ по x и множествами $S \subseteq \{1, 2, \dots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

for m in [2, n]:
  foreach $S \subseteq \{1, 2, \dots, n\} : |S| = m, 1 \in S$ do:
    for v in S, v != 1:
      $A[v, S] = \min\limits_{w \in S, w \neq v} \{A[w, S - \{v\}] + c(v, w)\}$

c(v, w) - длина ребра (v, w)

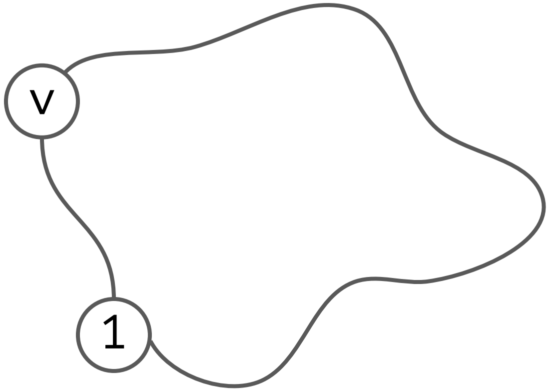
Оптимальная подструктура (TSP)



Допустим, что у нас кратчайший гамильтонов цикл из вершины 1 в вершину 1. И пусть v - последняя вершина в этом пути.

Тогда наша задача очевидно сводится к поиску кратчайшего пути из 1 в v…

Оптимальная подструктура (TSP)



Допустим, что у нас кратчайший гамильтонов цикл из вершины 1 в вершину 1. И пусть v - последняя вершина в этом пути.

Алгоритм динамики для TSP

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

for m in [2, n]:
  foreach $S \subseteq \{1, 2, \ldots, n\} : |S| = m, 1 \in S$ do:
    for v in S, v != 1:
      $$A[v, S] = \min_{w \in S, w \neq v} \{A[w, S - \{v\}] + c(v, w)\}$$

c(v, w) - длина ребра (v, w)

Алгоритм динамики для TSP

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

for m in [2, n]:
  foreach $S \subseteq \{1, 2, \ldots, n\} : |S| = m, 1 \in S$ do:
    for v in S, v != 1:

$$A[v, S] = \min_{w \in S, w \neq v} \{A[w, S - \{v\}] + c(v, w)\}$$

c(v, w) - длина ребра (v, w)

result = $\min_{v \in [2, n]} \{A[v, \{1, 2, \ldots, n\}] + c(v, 1)\}$

Алгоритм Беллмана - Хелда - Карпа

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

```
for m in [2, n]:
  foreach S ⊆ {1, 2, ..., n} : |S| = m, 1 ∈ S do:
    for v in S, v != 1:
```

$$A[v, S] = \min_{w \in S, w \neq v} \{A[w, S - \{v\}] + c(v, w)\}$$

c(v, w) - длина ребра (v, w)

result = $\min_{v \in [2, n]} \{A[v, \{1, 2, \ldots, n\}] + c(v, 1)\}$

Алгоритм Беллмана - Хелда - Карпа

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

for m in [2, n]:
  foreach $S \subseteq \{1, 2, \ldots, n\} : |S| = m, 1 \in S$ do:
    for v in S, v != 1:

$$A[v, S] = \min_{w \in S, w \neq v} \{A[w, S - \{v\}] + c(v, w)\}$$

result = $\min_{v \in [2, n]} \{A[v, \{1, 2, \ldots, n\}] + c(v, 1)\}$

Сложность?

Алгоритм Беллмана - Хелда - Карпа

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

for m in [2, n]:
  foreach $S \subseteq \{1, 2, \ldots, n\} : |S| = m, 1 \in S$ do:
    for v in S, v != 1:

$$A[v, S] = \min_{w \in S, w \neq v} \{A[w, S - \{v\}] + c(v, w)\}$$

result = $\min_{v \in [2, n]} \{A[v, \{1, 2, \ldots, n\}] + c(v, 1)\}$

Сложность?

Подзадач $O(n * 2^n)$

Алгоритм Беллмана - Хелда - Карпа

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \dots, n\}$ по x и множествами $S \subseteq \{1, 2, \dots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

for m in [2, n]:
  foreach $S \subseteq \{1, 2, \dots, n\} : |S| = m, 1 \in S$ do:
    for v in S, v != 1:
      $A[v, S] = \min\limits_{w \in S, w \neq v} \{A[w, S - \{v\}] + c(v, w)\}$

result = $\min\limits_{v \in [2, n]} \{A[v, \{1, 2, \dots, n\}] + c(v, 1)\}$

Сложность?

Подзадач $O(n * \boxed{2^n})$

Алгоритм Беллмана - Хелда - Карпа

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

for m in [2, n]:
  foreach $S \subseteq \{1, 2, \ldots, n\} : |S| = m, 1 \in S$ do:
    for v in S, v != 1:
      $A[v, S] = \min_{w \in S, w \neq v} \{A[w, S - \{v\}] + c(v, w)\}$

result = $\min_{v \in [2, n]} \{A[v, \{1, 2, \ldots, n\}] + c(v, 1)\}$

Сложность?

Подзадач $O(\boxed{n} * 2^n)$

Алгоритм Беллмана - Хелда - Карпа

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \dots, n\}$ по x и множествами $S \subseteq \{1, 2, \dots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

for m in [2, n]:
  foreach $S \subseteq \{1, 2, \dots, n\} : |S| = m, 1 \in S$ do:
    for v in S, v != 1:
      $A[v, S] = \min_{w \in S, w \neq v} \{A[w, S - \{v\}] + c(v, w)\}$

result = $\min_{v \in [2, n]} \{A[v, \{1, 2, \dots, n\}] + c(v, 1)\}$

Сложность?

Подзадач $O(n * 2^n)$

В каждой подзадаче $O(n)$

Алгоритм Беллмана - Хелда - Карпа

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

for m in [2, n]:
  foreach $S \subseteq \{1, 2, \ldots, n\} : |S| = m, 1 \in S$ do:
    for v in S, v != 1:
      $A[v, S] = \boxed{\min_{w \in S, w \neq v} \{A[w, S - \{v\}] + c(v, w)\}}$

result = $\min_{v \in [2, n]} \{A[v, \{1, 2, \ldots, n\}] + c(v, 1)\}$

Сложность?

Подзадач $O(n * 2^n)$

В каждой подзадаче $O(\boxed{n})$

Алгоритм Беллмана - Хелда - Карпа

Храним двумерный массив A, индексированного номерами вершин $v \in \{1, 2, \ldots, n\}$ по x и множествами $S \subseteq \{1, 2, \ldots, n\}$, включающими вершину 1 по y.

Базовая разметка: $A[1, S] = \begin{cases} 0, \text{ если } S = \{1\}, \\ +\infty, \text{ иначе} \end{cases}$

for m **in** [2, n]:
  foreach $S \subseteq \{1, 2, \ldots, n\} : |S| = m, 1 \in S$ **do**:
    for v **in** S, v != 1:
      $$A[v, S] = \min_{w \in S, w \neq v} \{A[w, S - \{v\}] + c(v, w)\}$$

result = $\min_{v \in [2, n]} \{A[v, \{1, 2, \ldots, n\}] + c(v, 1)\}$

Сложность?

Подзадач $O(n * 2^n)$

В каждой подзадаче $O(n)$

Итого: $O(n^2 * 2^n)$

Мини-задача #48 (2 балла, дополнительная)

Реализовать наивный перебор для TSP. Проверить, на каком размере графа он все еще работает.

Реализовать алгоритм Беллмана-Хелда-Карпа для решения задачи TSP.

Подумайте об удобном кодировании множеств S (и используйте его в решении).

Сравнить скорость решения с наивным перебором (на графах, где наивный еще работает за разумное время).

Алгоритм Беллмана - Хелда - Карпа

$O(n^2 * 2^n)$ конечно же лучше, чем $O(n!)$

Алгоритм Беллмана - Хелда - Карпа

$O(n^2 * 2^n)$ конечно же лучше, чем $O(n!)$

Но это все еще экспоненциальная сложность!

Алгоритм Беллмана - Хелда - Карпа

$O(n^2 * 2^n)$ конечно же лучше, чем $O(n!)$

Но это все еще экспоненциальная сложность!

Снова вопрос: можем ли мы лучше?
Например, за полиномиальное время?

Алгоритм Беллмана - Хелда - Карпа

$O(n^2 * 2^n)$ конечно же лучше, чем $O(n!)$

Но это все еще экспоненциальная сложность!

Снова вопрос: можем ли мы лучше?
Например, за полиномиальное время?

Ответ: **МЫ ТОЧНО НЕ ЗНАЕМ**

Алгоритм Беллмана - Хелда - Карпа

$O(n^2 * 2^n)$ конечно же лучше, чем $O(n!)$

Но это все еще экспоненциальная сложность!

Снова вопрос: можем ли мы лучше?
Например, за полиномиальное время?

Ответ: **МЫ ТОЧНО НЕ ЗНАЕМ**

Хотя думаем об этом по крайней мере последние 60 лет.

Алгоритм Беллмана - Хелда - Карпа

$O(n^2 * 2^n)$ конечно же лучше, чем $O(n!)$

Но это все еще экспоненциальная сложность!

Снова вопрос: можем ли мы лучше?
Например, за полиномиальное время?

Ответ: **МЫ ТОЧНО НЕ ЗНАЕМ**

Хотя думаем об этом по крайней мере последние 60 лет.

Но кое-что про такие задачи мы знаем.

DISCLAIMER:

Дальнейшие рассуждения – небольшой спойлер к тому, что будет на будущих курсах по алгоритмам.

Наша задача сейчас – получить базовое понимание. Детальнее вы будете это рассматривать позже.

Класс задач P

Что является хорошим алгоритмом для решения задачи?

Класс задач P

Что является хорошим алгоритмом для решения задачи?

Алгоритм, имеющий полиномиальную (или лучше) сложность.

Класс задач P

Что является хорошим алгоритмом для решения задачи?

Алгоритм, имеющий полиномиальную (или лучше) сложность.

Т.е. сложность $O(n^k)$, где n - размер входных данных, а k - некоторая константа.

Класс задач P

Что является хорошим алгоритмом для решения задачи?

Алгоритм, имеющий полиномиальную (или лучше) сложность.

Т.е. сложность $O(n^k)$, где n - размер входных данных, а k - некоторая константа.

Поправка на реальность: не всегда полиномиальный алгоритм так уж применим, представьте себе сложность $O(n^{1000})$. Но это хорошее приближение к понятию хорошего алгоритма.

Класс задач P

Определение: множество задач, для которых существует полиномиальный алгоритм, называется **P**.

Класс задач P

Определение: множество задач, для которых существует полиномиальный алгоритм, называется **P**.

Примеры?

Класс задач P

Определение: множество задач, для которых существует полиномиальный алгоритм, называется **P**.

Примеры: почти все алгоритмы, которые мы обсудили за год.

Класс задач P

Определение: множество задач, для которых существует полиномиальный алгоритм, называется **P**.

Примеры: почти все алгоритмы, которые мы обсудили за год.

Примеры задач, для которых мы не знаем полиномиальный алгоритм (и поэтому не уверены, что они в P)?

Класс задач P

Определение: множество задач, для которых существует полиномиальный алгоритм, называется **P**.

Примеры: почти все алгоритмы, которые мы обсудили за год.

Примеры задач, для которых мы не знаем полиномиальный алгоритм (и поэтому не уверены, что они в P)?

- TSP (только что разбирали)

- Найти кратчайшие пути, без циклов отрицательного веса

Класс задач P

Определение: множество задач, для которых существует полиномиальный алгоритм, называется **P**.

Примеры: почти все алгоритмы, которые мы обсудили за год.

Примеры задач, для которых мы не знаем полиномиальный алгоритм (и поэтому не уверены, что они в P)?

- TSP (только что разбирали)
- Найти кратчайшие пути, без циклов отрицательного веса
- Рюкзак!

Класс задач P

Определение: множество задач, для которых существует полиномиальный алгоритм, называется **P**.

Примеры: почти все алгоритмы, которые мы обсудили за год.

Примеры задач, для которых мы не знаем полиномиальный алгоритм (и поэтому не уверены, что они в P)?

- TSP (только что разбирали)
- Найти кратчайшие пути, без циклов отрицательного веса
- Рюкзак!?

Дискретная задача о рюкзаке

Сложность алгоритма?

Всего-то O(n*W)!

Дискретная задача о рюкзаке

Сложность алгоритма?

Всего-то O(n*W)!

Здесь n - количество вещей, а W - размер рюкзака.

Дискретная задача о рюкзаке

Сложность алгоритма?

Всего-то O(n*W)!

Здесь n - количество вещей, а W - размер рюкзака.

Но что такое "входные данные размер рюкзака"? Что такое размер этих входных данных?

Дискретная задача о рюкзаке

Сложность алгоритма?

Всего-то O(n*W)!

Здесь n - количество вещей, а W - размер рюкзака.

Но что такое "входные данные размер рюкзака"? Что такое размер этих входных данных?

Например, количество бит, которое в которое записано это число. Какая тогда длина входных данных?

Дискретная задача о рюкзаке

Сложность алгоритма?

Всего-то O(n*W)!

Здесь n - количество вещей, а W - размер рюкзака.

Но что такое "входные данные размер рюкзака"? Что такое размер этих входных данных?

Например, количество бит, которое в которое записано это число. Какая тогда длина входных данных? Это **logW**.

Дискретная задача о рюкзаке

Сложность алгоритма?

Всего-то O(n*W) = O(n*(2^k))

Здесь n - количество вещей, а W - размер рюкзака.

Но что такое "входные данные размер рюкзака"? Что такое размер этих входных данных?

Например, количество бит, которое в которое записано это число. Какая тогда длина входных данных? Это **logW** = k.

Дискретная задача о рюкзаке

Сложность алгоритма?

Всего-то O(n*W) = O(n*(2^k))

Здесь n - количество вещей, а W - размер рюкзака.

Но что такое "входные данные размер рюкзака"? Что такое размер этих входных данных?

Например, количество бит, которое в которое записано это число. Какая тогда длина входных данных? Это **logW** = k.

Т.е. сложность здесь экспоненциальная!!

Класс задач P

Определение: множество задач, для которых существует полиномиальный алгоритм, называется **P**.

Примеры: почти все алгоритмы, которые мы обсудили за год.

Примеры задач, для которых мы не знаем полиномиальный алгоритм (и поэтому не уверены, что они в P)?

- TSP (только что разбирали)
- Найти кратчайшие пути, без циклов отрицательного веса
- Рюкзак.

Класс задач NP

А что является, возможно, не таким хорошим алгоритмом но все же хоть как-то применимым?

Класс задач NP

А что является, возможно, не таким хорошим алгоритмом но все же хоть как-то применимым?

С чего мы всегда начинали обсуждение задачи?
С самого простого и прямолинейного решения: с brute-force.

Класс задач NP

А что является, возможно, не таким хорошим алгоритмом но все же хоть как-то применимым?

С чего мы всегда начинали обсуждение задачи?
С самого простого и прямолинейного решения: с brute-force.

Давайте опишем класс задач, которые можно решить с помощью перебора и brute-force.

Класс задач NP

Определение: будем говорить, что задача находится в классе NP, если для ее решения существует алгоритм, для которого исполняется 2 условия:

Класс задач NP

Определение: будем говорить, что задача находится в классе NP, если для ее решения существует алгоритм, для которого исполняется 2 условия:

1. Размер ответа всегда ограничен сверху полиномиальной функцией от размера входных данных.

Класс задач NP

Определение: будем говорить, что задача находится в классе NP, если для ее решения существует алгоритм, для которого исполняется 2 условия:

1. Размер ответа всегда ограничен сверху полиномиальной функцией от размера входных данных.

2. По некоторому кандидату на ответ, вы можете за полиномиальное время сказать: является ли он ответом на задачу или нет (т.е. проверифицировать его).

Класс задач NP

Определение: будем говорить, что задача находится в классе NP, если для ее решения существует алгоритм, для которого исполняется 2 условия:

1. Размер ответа всегда ограничен сверху полиномиальной функцией от размера входных данных.

2. По некоторому кандидату на ответ, вы можете за полиномиальное время сказать: является ли он ответом на задачу или нет (т.е. проверифицировать его).

Связь с полным перебором:

Класс задач NP

Определение: будем говорить, что задача находится в классе NP, если для ее решения существует алгоритм, для которого исполняется 2 условия:

1. Размер ответа всегда ограничен сверху полиномиальной функцией от размера входных данных.

2. По некоторому кандидату на ответ, вы можете за полиномиальное время сказать: является ли он ответом на задачу или нет (т.е. проверифицировать его).

Связь с полным перебором: размер любого кандидата $O(n^d)$ =>

Класс задач NP

Определение: будем говорить, что задача находится в классе NP, если для ее решения существует алгоритм, для которого исполняется 2 условия:

1. Размер ответа всегда ограничен сверху полиномиальной функцией от размера входных данных.

2. По некоторому кандидату на ответ, вы можете за полиномиальное время сказать: является ли он ответом на задачу или нет (т.е. проверифицировать его).

Связь с полным перебором: размер любого кандидата $O(n^d)$ => всего их $2^{O(n^d)}$ => каждый проверяем за полином => получили экспоненциальный алгоритм перебора

Класс задач NP

Примеры:

1. Любая задача из P

Класс задач NP

Примеры:

1. Любая задача из P

   Действительно: если уж вы решение **построили** за полином, то оно явно ограничено сверху по длине полиномом, и любого кандидата вы можете просто сравнить с правильным ответом.

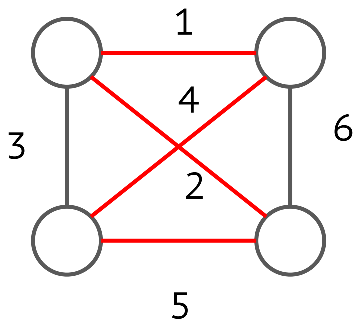
Класс задач NP

Примеры:

1. Любая задача из P
2. TSP

Задача коммивояжера a.k.a TSP

Задача: пусть дан неориентированный взвешенный граф G. Найти гамильтонов цикл (т.е. замкнутый путь, проходящий через каждую вершину ровно один раз) минимального веса.



Изначальная легенда - поиск кратчайшего маршрута для странствующего торговца (travelling salesman, коммивояжера)

Класс задач NP

Примеры:

2. TSP

Не совсем очевидно, как применить определения, поэтому переформулируем задачу:

Пусть дан граф из N вершин и константа k. Найти такой гамильтонов цикл, что его суммарный вес будет <= k.

Класс задач NP

Примеры:

2. TSP

Не совсем очевидно, как применить определения, поэтому переформулируем задачу:

Пусть дан граф из N вершин и константа k. Найти такой гамильтонов цикл, что его суммарный вес будет <= k.

Изначальная задача сводится к этой: почему?

Класс задач NP

Примеры:

2. TSP

Не совсем очевидно, как применить определения, поэтому переформулируем задачу:

Пусть дан граф из N вершин и константа k. Найти такой гамильтонов цикл, что его суммарный вес будет <= k.

Изначальная задача сводится к этой: если умеем решать эту, то обычный TSP решим с помощью бинарного поиска.

Класс задач NP

Примеры:

2. TSP

Не совсем очевидно, как применить определения, поэтому переформулируем задачу:

Пусть дан граф из N вершин и константа k. Найти такой гамильтонов цикл, что его суммарный вес будет <= k.

Изначальная задача сводится к этой: если умеем решать эту, то обычный TSP решим с помощью бинарного поиска.

Эта сводится к изначальной: если умеем искать вообще кратчайший => будем всегда брать его и проверять на k.

Класс задач NP

Примеры:

2. TSP

Не совсем очевидно, как применить определения, поэтому переформулируем задачу:

Пусть дан граф из N вершин и константа k. Найти такой гамильтонов цикл, что его суммарный вес будет <= k.

Ответ - это гамильтонов цикл, т.е. n вершин - полиномиально зависит от размера графа.

Класс задач NP

Примеры:

2. TSP

Не совсем очевидно, как применить определения, поэтому переформулируем задачу:

Пусть дан граф из N вершин и константа k. Найти такой гамильтонов цикл, что его суммарный вес будет <= k.

Ответ - это гамильтонов цикл, т.е. n вершин - полиномиально зависит от размера графа.

Пусть вам дали путь, как проверить, что это гамильтонов цикл суммарного веса <= k?

Класс задач NP

Примеры:

2. TSP

Не совсем очевидно, как применить определения, поэтому переформулируем задачу:

Пусть дан граф из N вершин и константа k. Найти такой гамильтонов цикл, что его суммарный вес будет <= k.

Ответ - это гамильтонов цикл, т.е. n вершин - полиномиально зависит от размера графа.

Пусть вам дали путь, как проверить, что это гамильтонов цикл суммарного веса <= k? За линейный проход по нему!

Класс задач NP

Определение: будем говорить, что задача находится в классе NP, если для ее решения существует алгоритм, для которого исполняется 2 условия:

1. Размер ответа всегда ограничен сверху полиномиальной функцией от размера входных данных.
2. По некоторому кандидату на ответ, вы можете за полиномиальное время сказать: является ли он ответом на задачу или нет (т.е. проверифицировать его).

Класс задач NP

Примеры:

1. Любая задача из P
2. TSP
3. Задача о сумме подмножеств: дан набор из N элементов найти хотя бы одно подмножество такое, чтобы сумма его элементов была равно нулю.

Класс задач NP

Примеры:

1. Любая задача из P
2. TSP
3. Задача о сумме подмножеств: дан набор из N элементов найти хотя бы одно подмножество такое, чтобы сумма его элементов была равно нулю.

   {3, 54, -2, 1, 19, -5, 4} -> {1, -5, 4}

Класс задач NP

Примеры:

1. Любая задача из P

2. TSP

3. Задача о сумме подмножеств: дан набор из N элементов найти хотя бы одно подмножество такое, чтобы сумма его элементов была равно нулю.

   {3, 54, -2, 1, 19, -5, 4} -> {1, -5, 4}

   - ответ точно не больше N
   - проверить можно за линию

Класс задач NP

Определение: будем говорить, что задача находится в классе NP, если для ее решения существует алгоритм, для которого исполняется 2 условия:

1. Размер ответа всегда ограничен сверху полиномиальной функцией от размера входных данных.
2. По некоторому кандидату на ответ, вы можете за полиномиальное время сказать: является ли он ответом на задачу или нет (т.е. проверифицировать его).

Класс задач NP

Примеры:

1. Любая задача из P
2. TSP
3. Задача о сумме подмножеств
4. Раскраска графа в три цвета

Класс задач NP

Примеры:

1. Любая задача из P
2. TSP
3. Задача о сумме подмножеств
4. Раскраска графа в три цвета
5. 3-SAT: есть набор булевых переменных x1, x2, …, xn, и набор формул, каждая из которых является дизъюнкцией не более трех литералов.

Класс задач NP

Примеры:

2. TSP
3. Задача о сумме подмножеств
4. Раскраска графа в три цвета
5. 3-SAT: есть набор булевых переменных x1, x2, …, xn, и набор формул, каждая из которых является дизъюнкцией не более трех литералов.

   Найти такое означивание переменных, чтобы все формулы были истинны.

Класс задач NP

Примеры:

1. Любая задача из P
2. TSP
3. Задача о сумме подмножеств
4. Раскраска графа в три цвета
5. 3-SAT
6. …

Класс задач NP

А бывают ли вообще задачи, которые не входят в NP?

Класс задач NP

А бывают ли вообще задачи, которые не входят в NP?

Да! Например, проблема остановки.

Класс задач NP

А бывают ли вообще задачи, которые не входят в NP?

Да! Например, проблема остановки.

Задача: по заданной функции и входным данным понять, остановится ли исполнение этой функции с заданными входными данными или нет.

Класс задач NP

А бывают ли вообще задачи, которые не входят в NP?

Да! Например, проблема остановки.

Задача: по заданной функции и входным данным понять, остановится ли исполнение этой функции с заданными входными данными или нет.

(как верифицировать ответ? запустить программу и подождать?)

Класс задач NP

Замечание #1: есть и альтернативное определение класса NP, связанное с исполнением на недетерминированной машине Тьюринга. В будущих курсах в с ними обязательно столкнетесь.

Класс задач NP

Замечание #1: есть и альтернативное определение класса NP, связанное с исполнением на недетерминированной машине Тьюринга. В будущих курсах в с ними обязательно столкнетесь.

Замечание #2: Что значит NP?

Класс задач NP

Замечание #1: есть и альтернативное определение класса NP, связанное с исполнением на недетерминированной машине Тьюринга. В будущих курсах в с ними обязательно столкнетесь.

Замечание #2: NP от "nondeterministic polynomial time" как раз из-за того, что их можно решить за полиномиальное время на недетерминированной машине Тьюринга.

Связь P и NP

Очевидно, что $P \subseteq NP$

Связь P и NP

Очевидно, что $P \subseteq NP$

Но существует ли такая задача из NP, которая при этом не входит в класс P?

Связь P и NP

Очевидно, что $P \subseteq NP$

Но существует ли такая задача из NP, которая при этом не входит в класс P?

P = NP? P != NP?

Связь P и NP

Очевидно, что $P \subseteq NP$

Но существует ли такая задача из NP, которая при этом не входит в класс P?

P = NP? P != NP?

Можно ли решить TSP, рюкзак, раскраску в три цвета за полином?

Связь P и NP

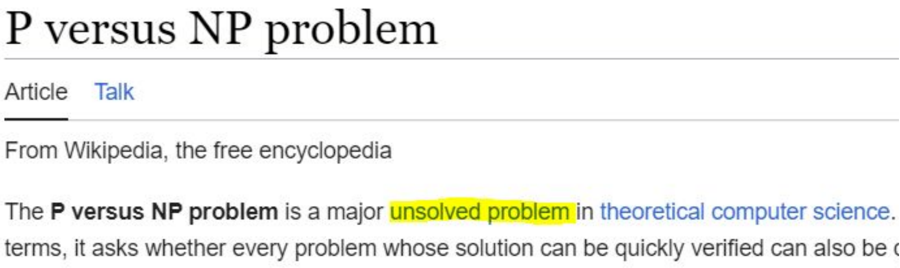
Очевидно, что $P \subseteq NP$

Но существует ли такая задача из NP, которая при этом не входит в класс P?

P = NP? P != NP?

Можно ли решить TSP, рюкзак, раскраску в три цвета за полином?

НЕИЗВЕСТНО.



P versus NP problem

Article Talk

From Wikipedia, the free encyclopedia

The **P versus NP problem** is a major unsolved problem in theoretical computer science.
terms, it asks whether every problem whose solution can be quickly verified can also be

Связь P и NP

Очевидно, что $P \subseteq NP$

Но существует ли такая задача из NP, которая при этом не входит в класс P? P = NP? P != NP?

Интуитивно кажется, что такая задача есть, т.е. P != NP.

Связь P и NP

Очевидно, что $P \subseteq NP$

Но существует ли такая задача из NP, которая при этом не входит в класс P? P = NP? P != NP?

Интуитивно кажется, что такая задача есть, т.е. P != NP.

Причины:

1) За 60 лет научных изысканий не удалось найти полиномиальные решения для TSP, 3-SAT и т.д.

Связь P и NP

Очевидно, что $P \subseteq NP$

Но существует ли такая задача из NP, которая при этом не входит в класс P? P = NP? P != NP?

Интуитивно кажется, что такая задача есть, т.е. P != NP.

Причины:

1) За 60 лет научных изысканий не удалось найти полиномиальные решения для TSP, 3-SAT и т.д.

2) Кажется, наш мир работает не так:
решить задачу сложнее, чем проверить ответ. Так ведь?

Связь P и NP

Очевидно, что $P \subseteq NP$

Но существует ли такая задача из NP, которая при этом не входит в класс P? P = NP? P != NP?

Интуитивно кажется, что такая задача есть, т.е. P != NP.

Доказательства нет. Как и доказательства обратного.

Связь P и NP

Очевидно, что $P \subseteq NP$

Но существует ли такая задача из NP, которая при этом не входит в класс P? P = NP? P != NP?

Интуитивно кажется, что такая задача есть, т.е. P != NP.

Доказательства нет. Как и доказательства обратного.

И на самом деле, ситуацию даже интереснее!

NP полнота

Определение: будем говорить, что проблема A сводится к B, если может быть решена с использованием полиномиального (по размеру входных данных) числа вызова подпрограммы B.

NP полнота

Определение: будем говорить, что проблема A сводится к B, если может быть решена с использованием полиномиального (по размеру входных данных) числа вызова подпрограммы B.

Хороший пример: общая TSP сводится к TSP с ограничением на суммарный вес тура.

NP полнота

Определение: будем говорить, что проблема A сводится к B, если может быть решена с использованием полиномиального (по размеру входных данных) числа вызова подпрограммы B.

Следствие #1: $B \in P \Rightarrow A \in P$ 😊

NP полнота

Определение: будем говорить, что проблема A сводится к B, если может быть решена с использованием полиномиального (по размеру входных данных) числа вызова подпрограммы B.

Следствие #1: $B \in P \Rightarrow A \in P$ 😊

Следствие #2: $A \notin P \Rightarrow B \notin P$ 🥺

NP полнота

Определение: будем говорить, что проблема A сводится к B, если может быть решена с использованием полиномиального (по размеру входных данных) числа вызова подпрограммы B.

Следствие #1: $B \in P \Rightarrow A \in P$ 😊

Следствие #2: $A \notin P \Rightarrow B \notin P$ 🥺

Еще говорят, что B по крайней мере такая же сложная, как A.

NP полнота

Определение: будем говорить, что проблема A сводится к B, если может быть решена с использованием полиномиального (по размеру входных данных) числа вызова подпрограммы B.

Определение: пусть C - класс задач. Про задачу A говорят, что она C-полна, если $A \in C$ и все задачи из C сводятся к A.

NP полнота

Определение: будем говорить, что проблема A сводится к B, если может быть решена с использованием полиномиального (по размеру входных данных) числа вызова подпрограммы B.

Определение: пусть C - класс задач. Про задачу A говорят, что она C-полна, если $A \in C$ и все задачи из C сводятся к A.

Утверждение: TSP - NP-полная задача.

NP полнота

Определение: будем говорить, что проблема A сводится к B, если может быть решена с использованием полиномиального (по размеру входных данных) числа вызова подпрограммы B.

Определение: пусть C - класс задач. Про задачу A говорят, что она C-полна, если $A \in C$ и все задачи из C сводятся к A.

Утверждение: TSP - NP-полная задача. Как и 3-SAT.

NP полнота

Определение: будем говорить, что проблема A сводится к B, если может быть решена с использованием полиномиального (по размеру входных данных) числа вызова подпрограммы B.

Определение: пусть C - класс задач. Про задачу A говорят, что она C-полна, если $A \in C$ и все задачи из C сводятся к A.

Утверждение: TSP - NP-полная задача. Как и 3-SAT. Как и раскраска графа в три цвета. Как и огромное количество других задач.

NP полнота


SAT
3-SAT
Пригодность
к трехцветной раскраске
MIP
Ориентированный
гамильтонов путь
Независимое
множество
Клика
Неориентированный
гамильтонов путь
Бесцикловый
кратчайший путь
Вершинное
покрытие
Сумма
подмножеств
Задача
коммивояжера
Рюкзак
Минимальная
производственная
продолжительность
Покрытие
множества
Задача о пути
коммивояжера
Самый дешевый
k-вершинный путь
Максимальный
охват
Максимизация
влияния

NP полнота

Определение: будем говорить, что проблема A сводится к B, если может быть решена с использованием полиномиального (по размеру входных данных) числа вызова подпрограммы B.

Определение: пусть C - класс задач. Про задачу A говорят, что она C-полна, если $A \in C$ и все задачи из C сводятся к A.

Утверждение: TSP - NP-полная задача. Как и 3-SAT. Как и раскраска графа в три цвета. Как и огромное количество других задач.

Следствие: достаточно для **одной** единственной NP-полной задачи найти полиномиальное решение, как ~~мир рухнет~~ окажется, что P = NP.

Связь P и NP

Очевидно, что $P \subseteq NP$

Но существует ли такая задача из NP, которая при этом не входит в класс P?

P = NP? P != NP?

Можно ли решить TSP, рюкзак, раскраску в три цвета за полином?

НЕИЗВЕСТНО.

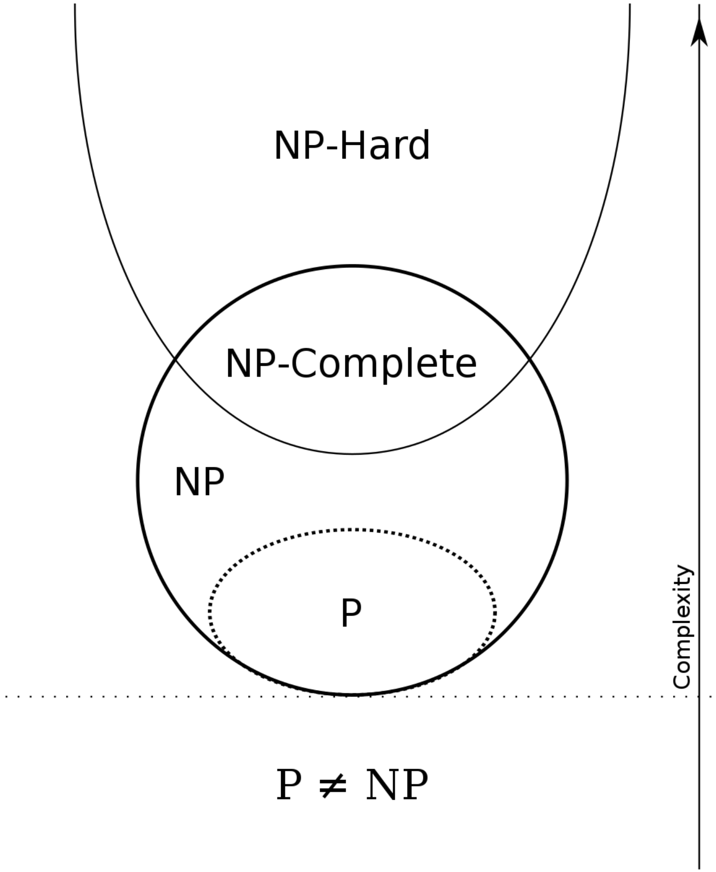
P versus NP problem

Article Talk

From Wikipedia, the free encyclopedia

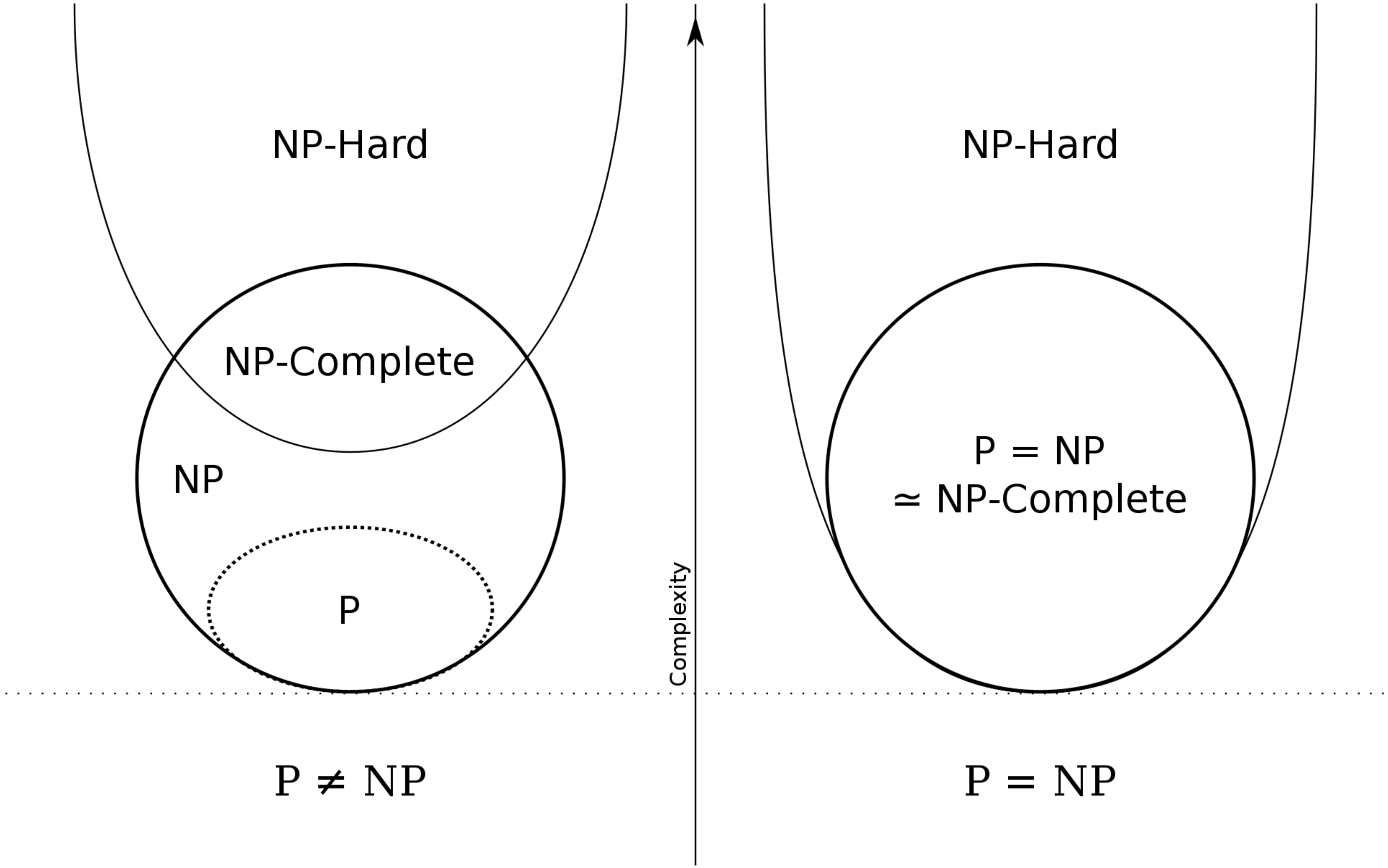
The **P versus NP problem** is a major unsolved problem in theoretical computer science.
terms, it asks whether every problem whose solution can be quickly verified can also be

Связь P и NP



P ≠ NP

Связь P и NP


NP-Hard
NP-Complete
NP
P
Complexity
NP-Hard
P = NP
≃ NP-Complete
P ≠ NP
P = NP

NP полнота: ближе к практике

Пусть на практике перед вами стоит некоторая задача.

NP полнота: ближе к практике

Пусть на практике перед вами стоит некоторая задача.

Если она вдруг окажется NP-полной, а вы потратите время на поиск ее полиномиального решения…

NP полнота: ближе к практике

Пусть на практике перед вами стоит некоторая задача.

Если она вдруг окажется NP-полной, а вы потратите время на поиск ее полиномиального решения… есть два варианта: вы случайно докажите, что P = NP (и заработаете много денег),

NP полнота: ближе к практике

Пусть на практике перед вами стоит некоторая задача.

Если она вдруг окажется NP-полной, а вы потратите время на поиск ее полиномиального решения… есть два варианта: вы случайно докажите, что P = NP (и заработаете много денег), либо вы потратите очень много времени впустую.

NP полнота: ближе к практике

Пусть на практике перед вами стоит некоторая задача.

Если она вдруг окажется NP-полной, а вы потратите время на поиск ее полиномиального решения… есть два варианта: вы случайно докажите, что P = NP (и заработаете много денег), либо вы потратите очень много времени впустую.

Поэтому, в первую очередь ищите NP-полную задачу, которая сводится к вашей.

NP полнота

Определение: будем говорить, что проблема A сводится к B, если может быть решена с использованием полиномиального (по размеру входных данных) числа вызова подпрограммы B.

Следствие #1: $B \in P \Rightarrow A \in P$ 😊

Следствие #2: $A \notin P \Rightarrow B \notin P$ 🥺

Еще говорят, что B по крайней мере такая же сложная, как A.

NP полнота

Определение: будем говорить, что проблема A сводится к B, если может быть решена с использованием полиномиального (по размеру входных данных) числа вызова подпрограммы B.

Следствие #1: $B \in P \Rightarrow A \in P$ 😊

Следствие #2: $A \notin P \Rightarrow B \notin P$ 🥺

Еще говорят, что B по крайней мере такая же сложная, как A.

Следствие #3: A - NP-полная => B тоже NP-полная.

NP полнота: ближе к практике

Пусть вы поняли, что ваша задача NP-полная.

Что дальше? Сдаемся?

NP полнота: ближе к практике

Пусть вы поняли, что ваша задача NP-полная.

Что дальше? Сдаемся?

Конечно нет! Как полиномиальное решение не является гарантией применимости на практике, так и NP-полнота - не приговор.

NP полнота: ближе к практике

Пусть вы поняли, что ваша задача NP-полная.

Что дальше? Сдаемся?

Конечно нет! Как полиномиальное решение не является гарантией применимости на практике, так и NP-полнота - не приговор.

Примеры реальных NP-полных задач для системных программистов:

- распределение регистров в компиляторе (это же раскраска графа в k цветов!)

NP полнота: ближе к практике

Пусть вы поняли, что ваша задача NP-полная.

Что дальше? Сдаемся?

Конечно нет! Как полиномиальное решение не является гарантией применимости на практике, так и NP-полнота - не приговор.

Примеры реальных NP-полных задач для системных программистов:

- распределение регистров в компиляторе (это же раскраска графа в k цветов!)
- оптимальное планирование задач планировщиком в виртуальной машине

NP полнота: ближе к практике

Как полиномиальное решение не является гарантией применимости на практике, так и NP-полнота - не приговор.

Стратегии решения NP-полной задачи:

NP полнота: ближе к практике

Как полиномиальное решение не является гарантией применимости на практике, так и NP-полнота - не приговор.

Стратегии решения NP-полной задачи:

1) Пожертвовать универсальностью: алгоритм будет работать эффективно, но не на всех входных данных.

NP полнота: ближе к практике

Как полиномиальное решение не является гарантией применимости на практике, так и NP-полнота - не приговор.

Стратегии решения NP-полной задачи:

1) Пожертвовать универсальностью: алгоритм будет работать эффективно, но не на всех входных данных.

   Пример: задача о рюкзаке. Работает хорошо только при маленьких W.

NP полнота: ближе к практике

Как полиномиальное решение не является гарантией применимости на практике, так и NP-полнота - не приговор.

Стратегии решения NP-полной задачи:

1) Пожертвовать универсальностью: алгоритм будет работать эффективно, но не на всех входных данных.

2) Смириться с не полиномиальной сложностью, но улучшать время работы по сравнению с полным перебором.

NP полнота: ближе к практике

Как полиномиальное решение не является гарантией применимости на практике, так и NP-полнота - не приговор.

Стратегии решения NP-полной задачи:

1) Пожертвовать универсальностью: алгоритм будет работать эффективно, но не на всех входных данных.

2) Смириться с не полиномиальной сложностью, но улучшать время работы по сравнению с полным перебором.

   Пример: решение TSP через динамику.

NP полнота: ближе к практике

Как полиномиальное решение не является гарантией применимости на практике, так и NP-полнота - не приговор.

Стратегии решения NP-полной задачи:

1) Пожертвовать универсальностью: алгоритм будет работать эффективно, но не на всех входных данных.

2) Смириться с не полиномиальной сложностью, но улучшать время работы по сравнению с полным перебором.

3) Пожертвовать точностью: алгоритм будет работать правильно и эффективно для большинства входных данных, такие алгоритмы называются эвристическими.

NP полнота: ближе к практике

На будущих курсах вы научитесь:

1. Использовать редукции, чтобы распознавать NP-полноту задачи,

NP полнота: ближе к практике

На будущих курсах вы научитесь:

1. Использовать редукции, чтобы распознавать NP-полноту задачи,

2. Докажите, что 3-SAT - NP-полная задача, и узнаете, к какому огромному множеству других задач она сводится,

NP полнота: ближе к практике

На будущих курсах вы научитесь:

1. Использовать редукции, чтобы распознавать NP-полноту задачи,

2. Докажите, что 3-SAT - NP-полная задача, и узнаете, к какому огромному множеству других задач она сводится,

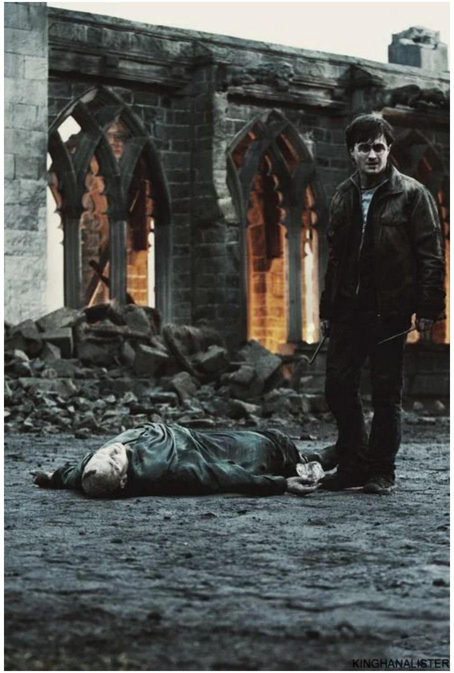
3. Рассмотрите многие практические решения NP-полных задач (в теории расписаний, в теории сложности, в методах трансляции и оптимизирующей компиляции и т.д.)

Takeaways

- Классы сложности задач: P и NP
- NP-полнота - не приговор, а указание, как именно к таким задачам подходить
- TSP, как яркий пример NP-полной задачи (для которой есть относительно быстрое решение)

Takeaways по всему курсу

- Сложность алгоритмов
    1. Временная и емкостная
    2. В худшем и среднем
    3. Амортизационная
    4. Классы сложности

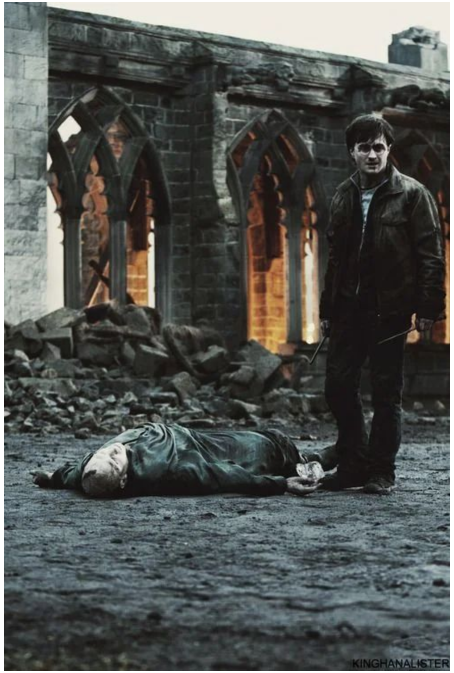


Takeaways по всему курсу

- Сложность алгоритмов

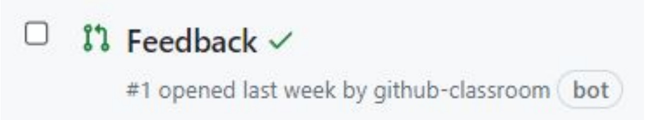
- Инструменты для разработки и анализа алгоритмов:

  1. Divide and Conquer
  2. Рандомизированные алгоритмы
  3. Жадность и динамика
  4. Различные деревья
  5. Первичное знание о NP-полноте



Takeaways по всему курсу

- Сложность алгоритмов
- Инструменты для разработки и анализа алгоритмов
- Практика на литкоде и на гитхабе



Алгоритмы и структуры данных продолжатся
в других курсах и спецкурсах (если вы этого захотите)