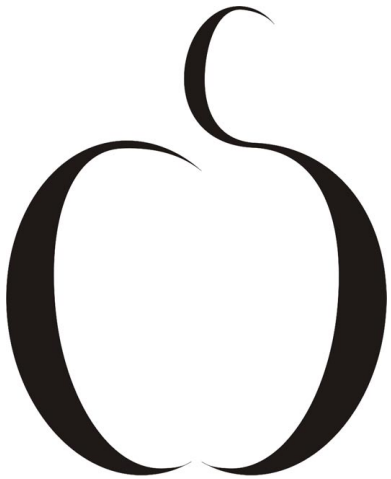
Алгоритмы для NP-трудных задач

Лекция 3: NP-полные задачи

А. Куликов

Computer Science клуб при ПОМИ
http://logic.pdmi.ras.ru/~infclub/



План лекции

1. Задачи поиска

План лекции

1. Задачи поиска
2. NP-полные задачи

План лекции

1. Задачи поиска
2. NP-полные задачи
3. Сведения

План лекции

Эффективные алгоритмы

Эффективные алгоритмы

- Алгоритмы нахождения кратчайших путей и минимальных покрывающих деревьев в графах, паросочетаний в двудольных графах, наибольшей возрастающей подпоследовательности, максимального потока в сети являются эффективными, поскольку время работы каждого из них растет полиномиально (как n, n^2, n^3) с ростом размера входа.

Эффективные алгоритмы

- Алгоритмы нахождения кратчайших путей и минимальных покрывающих деревьев в графах, паросочетаний в двудольных графах, наибольшей возрастающей подпоследовательности, максимального потока в сети являются эффективными, поскольку время работы каждого из них растет полиномиально (как n, n^2, n^3) с ростом размера входа.
- В каждой из решаемых ими задач мы ищем решение (путь, дерево, паросочетание) среди экспоненциально большого множества кандидатов: существует $n!$ различных паросочетаний в двудольном $n \times n$ графе, у полного графа на n вершинах есть n^{n-2} покрывающих деревьев, в графе, как правило, есть экспоненциальное число путей из s в t.

Эффективные алгоритмы

Эффективные алгоритмы

- Все эти задачи могут, в принципе, быть решены за экспоненциальное время перебором всех кандидатов. Однако алгоритм, время работы которого есть 2^n или хуже, достаточно бесполезен на практике.

Эффективные алгоритмы

- Все эти задачи могут, в принципе, быть решены за экспоненциальное время перебором всех кандидатов. Однако алгоритм, время работы которого есть 2^n или хуже, достаточно бесполезен на практике.
- Задачей эффективного алгоритма, таким образом, является нахождение способов избежать полного перебора всех кандидатов.

Задача выполнимости

Задача выполнимости

- Вход задачи пропозициональной выполнимости (satisfiability problem, SAT) выглядит следующим образом:

$$(x \vee y \vee z)(x \vee \neg y)(y \vee \neg z)(z \vee \neg x)(\neg x \vee \neg y \vee \neg z)\,.$$

Задача выполнимости

- Вход задачи пропозициональной выполнимости (satisfiability problem, SAT) выглядит следующим образом:

$$(x \vee y \vee z)(x \vee \neg y)(y \vee \neg z)(z \vee \neg x)(\neg x \vee \neg y \vee \neg z)\,.$$

- Задача пропозициональной выполнимости заключается в проверке существования выполняющего набора у заданной формулы.

Задача поиска

Задача поиска

- Задача выполнимости является типичной задачей поиска (search problem).

Задача поиска

- Задача выполнимости является типичной задачей поиска (search problem).
- Дано условие (instance) I (то есть некоторые входные данные, специфицирующие рассматриваемую задачу; в нашем случае — булева формула в КНФ) и требуется найти решение (solution) S (объект, удовлетворяющий некоторым условиям; в рассматриваемом нами случае — выполняющий набор). Если решения не существует, необходимо сообщить об этом.

Задача поиска

- Задача выполнимости является типичной задачей поиска (search problem).
- Дано условие (instance) I (то есть некоторые входные данные, специфицирующие рассматриваемую задачу; в нашем случае — булева формула в КНФ) и требуется найти решение (solution) S (объект, удовлетворяющий некоторым условиям; в рассматриваемом нами случае — выполняющий набор). Если решения не существует, необходимо сообщить об этом.
- Необходимым свойством задачи поиска является является наличие быстрой проверки корректности данного решения S для данного условия I.

Задача поиска

Будем говорить, что задача поиска задается алгоритмом $\mathcal{C}$, который получает на вход условие I и кандидата на решение S и работает полиномиальное от $|I|$ время. Мы говорим, что S является решением для I тогда и только тогда, когда $\mathcal{C}(I, S) = \texttt{true}$.

Задача о коммивояжере

Задача о коммивояжере

- В задаче о коммивояжере (traveling salesman problem, TSP) дано n вершин с номерами $1, \ldots, n$ и все $n(n-1)/2$ попарных расстояний между ними, а также бюджет b.

Задача о коммивояжере

- В задаче о коммивояжере (traveling salesman problem, TSP) дано n вершин с номерами $1, \ldots, n$ и все $n(n-1)/2$ попарных расстояний между ними, а также бюджет b.
- Найти необходимо цикл, проходящий через все вершины ровно по одному разу (то есть гамильтонов цикл) и имеющий длину не более b, или же сообщить, что такого цикла нет.

Задача о коммивояжере

- В задаче о коммивояжере (traveling salesman problem, TSP) дано n вершин с номерами $1, \ldots, n$ и все $n(n-1)/2$ попарных расстояний между ними, а также бюджет b.
- Найти необходимо цикл, проходящий через все вершины ровно по одному разу (то есть гамильтонов цикл) и имеющий длину не более b, или же сообщить, что такого цикла нет.
- Другими словами, необходимо найти перестановку $\tau(1), \ldots, \tau(n)$ вершин, такую что

$$d_{\tau(1),\tau(2)} + d_{\tau(2),\tau(3)} + \cdots + d_{\tau(n),\tau(1)} \leq b\,.$$

Оптимизационные задачи и задачи поиска

Оптимизационные задачи и задачи поиска

- Почему мы сформулировали задачу о коммивояжере как задачу поиска, когда на самом деле это оптимизационная задача?

Оптимизационные задачи и задачи поиска

- Почему мы сформулировали задачу о коммивояжере как задачу поиска, когда на самом деле это оптимизационная задача?
- Переформулировка оптимизационной задачи как задачи поиска не меняет ее сложности, поскольку обе версии сводятся друг к другу.

Оптимизационные задачи и задачи поиска

- Почему мы сформулировали задачу о коммивояжере как задачу поиска, когда на самом деле это оптимизационная задача?
- Переформулировка оптимизационной задачи как задачи поиска не меняет ее сложности, поскольку обе версии сводятся друг к другу.
- Имея потенциальное решение для задачи TSP, легко проверить свойства “является маршрутом” и “имеет длину не более b”, но как проверить свойство “является оптимальным”?

Задача о коммивояжере и задача о покрывающем дереве

Задача о коммивояжере и задача о покрывающем дереве

- Задача о минимальном покрывающем дереве: дана матрица попарных расстояний и бюджет b и требуется найти дерево T, такое что

$$\sum_{(i,j)\in T} d_{ij} \leq b\,.$$

Задача о коммивояжере и задача о покрывающем дереве

- Задача о минимальном покрывающем дереве: дана матрица попарных расстояний и бюджет b и требуется найти дерево T, такое что
$$\sum_{(i,j)\in T} d_{ij} \leq b\,.$$
- Задача о коммивояжере может рассматриваться как вариант задачи о покрывающем дереве, где искомому дереву не разрешается ветвиться и оно, таким образом, должно быть просто путем.

Задача о коммивояжере и задача о покрывающем дереве

- Задача о минимальном покрывающем дереве: дана матрица попарных расстояний и бюджет b и требуется найти дерево T, такое что
$$\sum_{(i,j)\in T} d_{ij} \leq b\,.$$
- Задача о коммивояжере может рассматриваться как вариант задачи о покрывающем дереве, где искомому дереву не разрешается ветвиться и оно, таким образом, должно быть просто путем.
- Это дополнительное условие на структуру дерева делает задачу гораздо более сложной.

Незавсисмое множество, вершинное покрытие и клика

Независмое множество, вершинное покрытие и клика

- В задаче о независимом множестве (independent set problem) дан граф и число g и требуется найти g независимых вершин, то есть таких, что никакие две из них не соединены ребром.

Независмое множество, вершинное покрытие и клика

- В задаче о независимом множестве (independent set problem) дан граф и число g и требуется найти g независимых вершин, то есть таких, что никакие две из них не соединены ребром.
- В задаче о вершинном покрытии (vertex cover problem) дан граф и число b, и требуется найти b вершин, покрывающих все ребра (то есть такое множество из b вершин, что для любого ребра хотя бы один из его концов содержится в этом множестве).

Независмое множество, вершинное покрытие и клика

- В задаче о независимом множестве (independent set problem) дан граф и число g и требуется найти g независимых вершин, то есть таких, что никакие две из них не соединены ребром.
- В задаче о вершинном покрытии (vertex cover problem) дан граф и число b, и требуется найти b вершин, покрывающих все ребра (то есть такое множество из b вершин, что для любого ребра хотя бы один из его концов содержится в этом множестве).
- Задача о клике (clique problem) заключается в нахождении по графу и числу g таких g вершин, что любые две из них соединены ребром.

Длиннейший путь

Длиннейший путь

- Задача о нахождении кратчайшего пути может быть решена эффективно, но так ли это для задачи о длиннейшем пути (longest path problem)?

Длиннейший путь

- Задача о нахождении кратчайшего пути может быть решена эффективно, но так ли это для задачи о длиннейшем пути (longest path problem)?
- В данной задаче нам дан граф с неотрицательными весами на ребрах, две его вершины s и t, а также число g.

Длиннейший путь

- Задача о нахождении кратчайшего пути может быть решена эффективно, но так ли это для задачи о длиннейшем пути (longest path problem)?
- В данной задаче нам дан граф с неотрицательными весами на ребрах, две его вершины s и t, а также число g.
- Найти необходимо простой путь из s в t веса хотя бы g.

Задача о рюкзаке

Задача о рюкзаке

- Задача о рюкзаке (knapsack problem): даны объемы $w_1, \ldots, w_n \in \mathbb{N}$ и стоимости $v_1, \ldots, v_n \in \mathbb{N}$ набора из n предметов. Также даны общий объем W и число g.

Задача о рюкзаке

- Задача о рюкзаке (knapsack problem): даны объемы $w_1, \ldots, w_n \in \mathbb{N}$ и стоимости $v_1, \ldots, v_n \in \mathbb{N}$ набора из n предметов. Также даны общий объем W и число g.
- Найти необходимо множество предметов из набора, общий объем которых не превышает W, а общая стоимость не менее g.

Задача о рюкзаке

- Задача о рюкзаке (knapsack problem): даны объемы $w_1, \ldots, w_n \in \mathbb{N}$ и стоимости $v_1, \ldots, v_n \in \mathbb{N}$ набора из n предметов. Также даны общий объем W и число g.
- Найти необходимо множество предметов из набора, общий объем которых не превышает W, а общая стоимость не менее g.
- Существует простой алгоритм для задачи о рюкзаке, основанный на методе динамического программирования и имеющий время работы $O(nW)$. Данное время работы является экспоненциальным от размера входа, поскольку оно включает в себя W, а не $\log W$.

Задача об унарном рюкзаке

Задача об унарном рюкзаке

- Допустим, однако, что мы решаем задачу о рюкзаке, в которой все числа даются на вход в унарной системе счисления.

Задача об унарном рюкзаке

- Допустим, однако, что мы решаем задачу о рюкзаке, в которой все числа даются на вход в унарной системе счисления.
- Например, число 12 записывается как ////////////.

Задача об унарном рюкзаке

- Допустим, однако, что мы решаем задачу о рюкзаке, в которой все числа даются на вход в унарной системе счисления.
- Например, число 12 записывается как ////////////.
- Конечно же, это неэкономный способ представления целых чисел, однако он все же позволяет определить законную задачу поиска, которую мы будем называть задачей об унарном рюкзаке (unary knapsack).

Задача об унарном рюкзаке

- Допустим, однако, что мы решаем задачу о рюкзаке, в которой все числа даются на вход в унарной системе счисления.
- Например, число 12 записывается как ////////////.
- Конечно же, это неэкономный способ представления целых чисел, однако он все же позволяет определить законную задачу поиска, которую мы будем называть задачей об унарном рюкзаке (unary knapsack).
- Как видно, данная слегка искусственная задача имеет полиномиальный алгоритм.

Задача о сумме подмножества

Задача о сумме подмножества

- Рассмотрим теперь другую вариацию. Пусть стоимость каждого предмета равняется его объему, а общий вес W равен числу g.

Задача о сумме подмножества

- Рассмотрим теперь другую вариацию. Пусть стоимость каждого предмета равняется его объему, а общий вес W равен числу g.
- Получившаяся задача, известная как задача о сумме подмножества (subset sum problem), эквивалентна выбору подмножества чисел с общей суммой W из заданного множества чисел. Поскольку это частный случай задачи о рюкзаке, данная задача не может быть сложнее. Но может ли она решаться за полиномиальное время?

План лекции

Сложные и простые задачи

трудные задачи (NP-полные)	простые задачи (из P)
3-SAT	2-SAT, Horn SAT
задача о коммивояжере	задача о покрывающем дереве
длиннейший путь	кратчайший путь
задача о рюкзаке	задача об унарном рюкзаке
задача о независимом множестве	задача о независимом множестве для деревьев

Сложные и простые задачи

Сложные и простые задачи

- Задачи из правого столбца могут быть решены таких методами, как динамическое программирование, потоки в сетях, поиск в графе, жадные алгоритмы. Эти задачи просты по многим различным причинам.

Сложные и простые задачи

- Задачи из правого столбца могут быть решены таких методами, как динамическое программирование, потоки в сетях, поиск в графе, жадные алгоритмы. Эти задачи просты по многим различным причинам.
- В свою очередь, все задачи из левого столбца сложны по одной и той же причине.

Сложные и простые задачи

- Задачи из правого столбца могут быть решены таких методами, как динамическое программирование, потоки в сетях, поиск в графе, жадные алгоритмы. Эти задачи просты по многим различным причинам.
- В свою очередь, все задачи из левого столбца сложны по одной и той же причине.
- Все они эквивалентны: как мы увидим, любая из них может быть сведена к любой другой из них и обратно.

Классы P и NP

Классы P и NP

- NP — класс всех задач поиска.

Классы P и NP

- NP — класс всех задач поиска.
- P — класс всех задач поиска, которые могут быть решены за полиномиальное время.

Сведения

Сведения

- Даже если допустить, что P≠NP, что можно сказать о задачах из левого столбца таблицы? Основываясь на чем, мы полагаем, что эти конкретные задачи не имеют эффективных алгоритмов (не считая того исторического факта, что многие сильные математики безуспешно пытались построить такие алгоритмы)?

Сведения

- Даже если допустить, что P≠NP, что можно сказать о задачах из левого столбца таблицы? Основываясь на чем, мы полагаем, что эти конкретные задачи не имеют эффективных алгоритмов (не считая того исторического факта, что многие сильные математики безуспешно пытались построить такие алгоритмы)?
- Это объясняется сведéниями (reductions), которые переводят одну задачу поиска в другую. Они показывают, что все задачи левого столбца в действительности являются в некотором смысле одной и той же задачей, за исключением лишь того, что они сформулированы в разных терминах.

Сведения

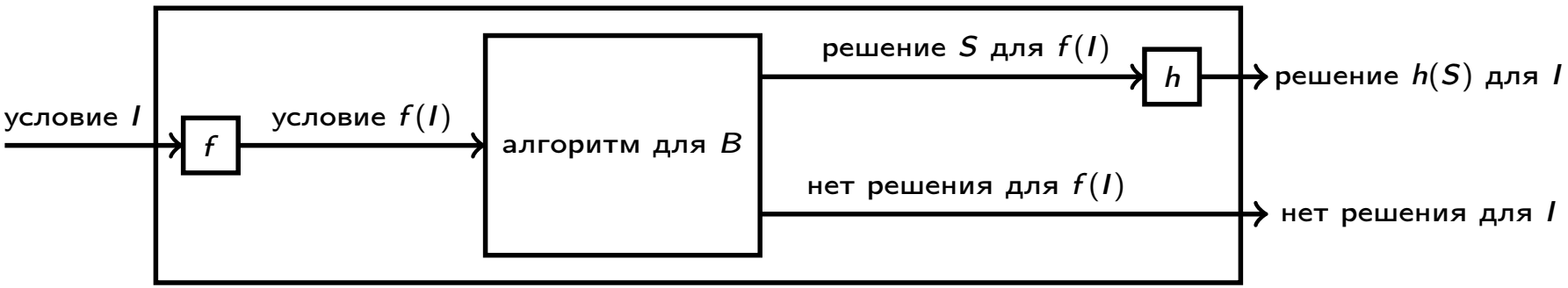
- Даже если допустить, что P≠NP, что можно сказать о задачах из левого столбца таблицы? Основываясь на чем, мы полагаем, что эти конкретные задачи не имеют эффективных алгоритмов (не считая того исторического факта, что многие сильные математики безуспешно пытались построить такие алгоритмы)?
- Это объясняется сведéниями (reductions), которые переводят одну задачу поиска в другую. Они показывают, что все задачи левого столбца в действительности являются в некотором смысле одной и той же задачей, за исключением лишь того, что они сформулированы в разных терминах.
- Более того, используя сведения, мы покажем, что эти задачи являются самыми трудными задачами поиска класса NP: если для хотя бы одной из них существует полиномиальный алгоритм, то полиномиальный алгоритм существует для каждой задачи класса NP. Значит, если мы верим в то, что P≠NP, то все эти задачи сложны.

Сведения

Сведением задачи поиска A к задаче поиска B называется пара полиномиальных по времени алгоритмов (f, h), где f по любому условию I задачи A выдает условие $f(I)$ задачи B, а h по любому решению S для $f(I)$ выдает решение $h(S)$ для I, см. диаграмму ниже. Если у $f(I)$ нет решения, то его нет и у I. Данные два алгоритма f и h могут быть использованы для того, чтобы любой алгоритм для задачи B переделать в алгоритм для задачи A.

Сведения

Сведением задачи поиска A к задаче поиска B называется пара полиномиальных по времени алгоритмов (f, h), где f по любому условию I задачи A выдает условие $f(I)$ задачи B, а h по любому решению S для $f(I)$ выдает решение $h(S)$ для I, см. диаграмму ниже. Если у $f(I)$ нет решения, то его нет и у I. Данные два алгоритма f и h могут быть использованы для того, чтобы любой алгоритм для задачи B переделать в алгоритм для задачи A.



NP-полные задачи

NP-полные задачи

- Задача поиска называется NP-полной (NP-complete), если все другие задачи поиска сводятся к ней.

NP-полные задачи

- Задача поиска называется NP-полной (NP-complete), если все другие задачи поиска сводятся к ней.
- Чтобы задача была NP-полной, с ее помощью должны решаться все задачи поиска в мире!

NP-полные задачи

- Задача поиска называется NP-полной (NP-complete), если все другие задачи поиска сводятся к ней.
- Чтобы задача была NP-полной, с ее помощью должны решаться все задачи поиска в мире!
- Поразительно, что такие задачи существуют вообще. И они действительно существуют, и правый столбец нашей таблицы заполнен наиболее известными такими примерами.

Два способа использования сведений

Задача поиска A сводится к задаче поиска B:

Два способа использования сведений

Задача поиска A сводится к задаче поиска B:

- Мы знаем, как эффективно решать B и хотим использовать это знание для решения A.

Два способа использования сведений

Задача поиска A сводится к задаче поиска B:

- Мы знаем, как эффективно решать B и хотим использовать это знание для решения A.
- Мы знаем, что A трудна, и используем сведение, чтобы показать, что и B трудна.

Два способа использования сведений

Задача поиска A сводится к задаче поиска B:

- Мы знаем, как эффективно решать B и хотим использовать это знание для решения A.
- Мы знаем, что A трудна, и используем сведение, чтобы показать, что и B трудна.

Если обозначить сведение от A к B через $A \longrightarrow B$, то можно сказать, что трудность движется в направлении стрелки, в то время как эффективные алгоритмы двигаются в обратном направлении. Через это распространение мы знаем, что NP-полные задачи трудны: все задачи поиска сводятся к ним, и, таким образом, каждая NP-полная задача содержит трудность всех задач поиска. Если хотя бы одна NP-полная задача принадлежит классу P, то P=NP.

План лекции

Сведения

Мы построим следующие сведения:

любая задача из NP $\longrightarrow$ SAT $\longrightarrow$ 3SAT $\longrightarrow$ независимое множество $\longrightarrow$ вершинное покрытие, клика.

3-SAT $\longrightarrow$ задача о независимом множестве

В задаче 3-SAT входом является множество клозов, каждый из которых состоит из не более чем трех литералов, например,

$$(\neg x \vee y \vee \neg z)(x \vee \neg y \vee z)(x \vee y \vee z)(\neg x \vee \neg y)\,,$$

и найти необходимо выполняющий набор. В задаче о независимом множестве даны граф и число g и найти необходимо g попарно не соединенных ребрами вершин. Мы, таким образом, должны как-то связать булеву логику с графами.

Конструкция

По данной формуле I задачи 3-SAT мы строим вход (G, g) задачи о независимом множестве следующим образом.

Конструкция

По данной формуле I задачи 3-SAT мы строим вход (G, g) задачи о независимом множестве следующим образом.

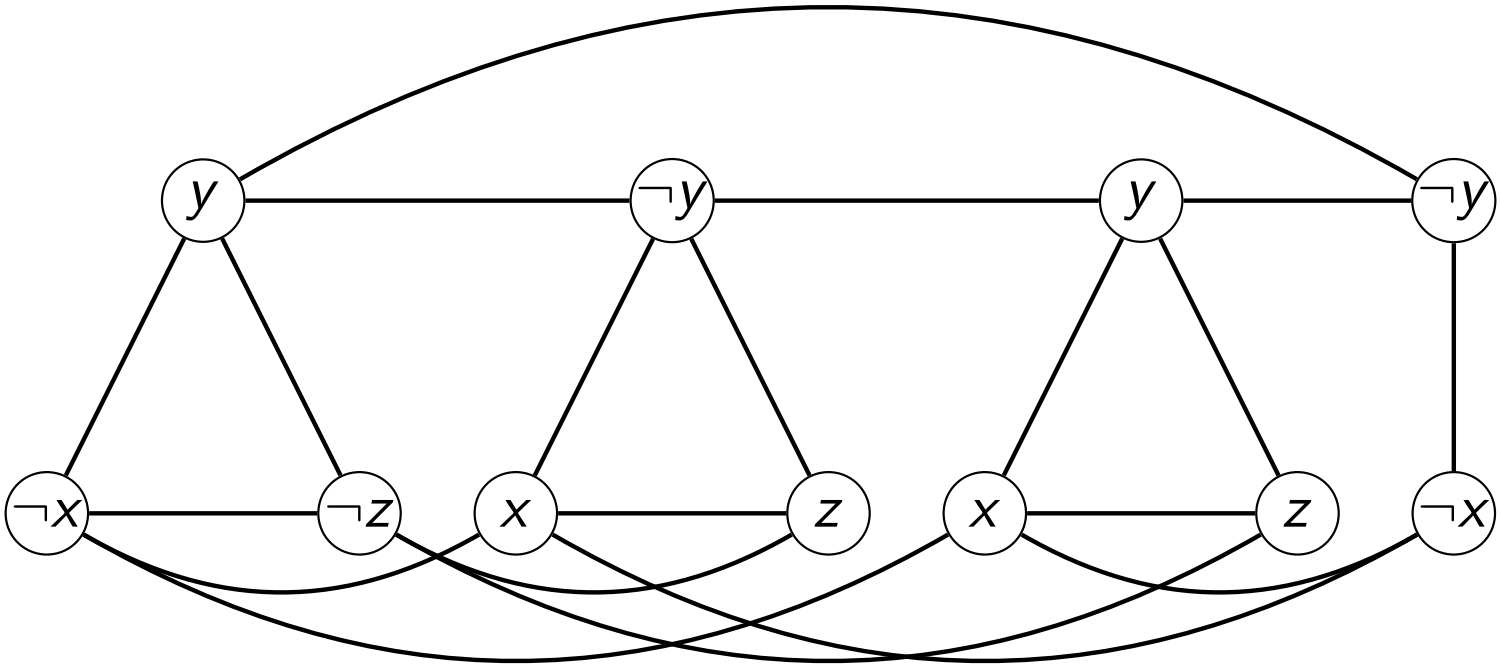
- В графе G каждому клозу соответствует треугольник (или просто ребро, если клоз состоит из всего двух литералов) с вершинами, помеченными литералами из клоза; все пары вершин, соответствующих противоположным литералам, соединены.

Конструкция

По данной формуле I задачи 3-SAT мы строим вход (G, g) задачи о независимом множестве следующим образом.

- В графе G каждому клозу соответствует треугольник (или просто ребро, если клоз состоит из всего двух литералов) с вершинами, помеченными литералами из клоза; все пары вершин, соответствующих противоположным литералам, соединены.
- Число g равно количеству клозов.

Конструкция



$$(\neg x \vee y \vee \neg z)(x \vee \neg y \vee z)(x \vee y \vee z)(\neg x \vee \neg y)$$

Анализ

Анализ

- Построение занимает полиномиальное время.

Анализ

- Построение занимает полиномиальное время.
- По данному независимому множеству S из g вершин графа G можно эффективно восстановить выполняющий истинностный набор для I.

Анализ

- Построение занимает полиномиальное время.
- По данному независимому множеству S из g вершин графа G можно эффективно восстановить выполняющий истинностный набор для I.
- Если в G нет независимого множества размера g, то формула I невыполнима.

SAT $\longrightarrow$ 3-SAT

SAT $\longrightarrow$ 3-SAT

- Это интересное и типичное сведение задачи к своему частному случаю.

SAT → 3-SAT

- Это интересное и типичное сведение задачи к своему частному случаю.
- Мы хотим показать, что задача остается сложной, даже если ее входы как-то ограничены — в нашем случае, если все клозы содержат не более трех литералов.

SAT → 3-SAT

- Это интересное и типичное сведение задачи к своему частному случаю.
- Мы хотим показать, что задача остается сложной, даже если ее входы как-то ограничены — в нашем случае, если все клозы содержат не более трех литералов.
- Такие сведения модифицируют входное условие так, чтобы избавиться от запрещенных конфигураций (клозов длины хотя бы четыре), не меняя при этом условия в том смысле, что решение для исходного условия можно построить из решения для полученного условия.

Конструкция

Конструкция

- По данной формуле I в КНФ рассмотрим формулу I, в которой каждый клоз $(a_1 \vee a_2 \vee \cdots \vee a_k)$ длины более 3 заменяется на множество клозов

$$(a_1 \vee a_2 \vee y_1)(\neg y_1 \vee a_3 \vee y_2)(\neg y_2 \vee a_4 \vee y_3) \ldots (\neg y_{k-3} \vee a_{k-1} \vee a_k),$$

где $\{y_i\}$ суть новые переменные.

Конструкция

- По данной формуле I в КНФ рассмотрим формулу I, в которой каждый клоз $(a_1 \vee a_2 \vee \cdots \vee a_k)$ длины более 3 заменяется на множество клозов

  $$(a_1 \vee a_2 \vee y_1)(\neg y_1 \vee a_3 \vee y_2)(\neg y_2 \vee a_4 \vee y_3) \ldots (\neg y_{k-3} \vee a_{k-1} \vee a_k),$$

  где $\{y_i\}$ суть новые переменные.
- Обозначим полученную формулу в 3-КНФ через I'.

Конструкция

- По данной формуле I в КНФ рассмотрим формулу I, в которой каждый клоз $(a_1 \vee a_2 \vee \cdots \vee a_k)$ длины более 3 заменяется на множество клозов

  $$(a_1 \vee a_2 \vee y_1)(\neg y_1 \vee a_3 \vee y_2)(\neg y_2 \vee a_4 \vee y_3)\ldots(\neg y_{k-3} \vee a_{k-1} \vee a_k),$$

  где $\{y_i\}$ суть новые переменные.
- Обозначим полученную формулу в 3-КНФ через I'.
- Ясно, что формула I' строится по I за полиномиальное время.

Анализ

Анализ

- I' эквивалентно I в смысле выполнимости, потому что для каждого набора переменным $\{a_i\}$ клоз $(a_1 \vee a_2 \vee \cdots \vee a_k)$ выполнен тогда и только тогда, когда существует набор значений переменным $\{y_i\}$, выполняющий все соответствующие клозы.

Анализ

- I' эквивалентно I в смысле выполнимости, потому что для каждого набора переменным $\{a_i\}$ клоз $(a_1 \vee a_2 \vee \cdots \vee a_k)$ выполнен тогда и только тогда, когда существует набор значений переменным $\{y_i\}$, выполняющий все соответствующие клозы.
- Чтобы увидеть это, предположим сперва, что все эти клозы выполнены. Тогда должен быть выполнен и хотя бы один из литералов $a_1, \ldots, a_k$, ибо в противном случае переменной y_1 должно было бы присвоено значение `true`, чтобы выполнить первый клоз, аналогично с переменной y_2 для выполнения второго клоза и всеми остальными переменными, и последний клоз был бы не выполнен.

Анализ

- I' эквивалентно I в смысле выполнимости, потому что для каждого набора переменным $\{a_i\}$ клоз $(a_1 \vee a_2 \vee \cdots \vee a_k)$ выполнен тогда и только тогда, когда существует набор значений переменным $\{y_i\}$, выполняющий все соответствующие клозы.
- Чтобы увидеть это, предположим сперва, что все эти клозы выполнены. Тогда должен быть выполнен и хотя бы один из литералов $a_1, \ldots, a_k$, ибо в противном случае переменной y_1 должно было бы присвоено значение `true`, чтобы выполнить первый клоз, аналогично с переменной y_2 для выполнения второго клоза и всеми остальными переменными, и последний клоз был бы не выполнен.
- Обратно, если клоз $(a_1 \vee a_2 \vee \cdots \vee a_k)$ выполнен, то для некоторого i выполнен литерал a_i. Присвоим тогда значение `true` переменным $y_1, \ldots, y_{i-2}$, а оставшимся присвоим значение `false`. Несложно видеть, что все клозы будут выполнены.

Задача о независимом множестве $\longrightarrow$ задача о вершинном покрытии

Задача о независимом множестве $\longrightarrow$ задача о вершинном покрытии

- Некоторые сведения основаны на способе связать две непохожие друг на друга задачи, а некоторые просто отражают тот факт, что одна задача является переформулировкой другой.

Задача о независимом множестве $\longrightarrow$ задача о вершинном покрытии

- Некоторые сведения основаны на способе связать две непохожие друг на друга задачи, а некоторые просто отражают тот факт, что одна задача является переформулировкой другой.
- Чтобы свести задачу о независимом множестве к задаче о вершинном покрытии, нужно просто заметить, что множество S является вершинным покрытием графа $G(V, E)$ тогда и только тогда, когда оставшиеся вершины, то есть множество $S \setminus V$, независимы в G.

Задача о независимом множестве $\longrightarrow$ задача о вершинном покрытии

- Некоторые сведения основаны на способе связать две непохожие друг на друга задачи, а некоторые просто отражают тот факт, что одна задача является переформулировкой другой.
- Чтобы свести задачу о независимом множестве к задаче о вершинном покрытии, нужно просто заметить, что множество S является вершинным покрытием графа $G(V, E)$ тогда и только тогда, когда оставшиеся вершины, то есть множество $S \setminus V$, независимы в G.
- Таким образом, чтобы решить задачу о независимом множестве для входа (G, g), нужно просто решить задачу о вершинном покрытии для входа $(G, |V| - g)$. Если такое покрытие существует, возьмем все вершины, не принадлежащие ему. Если такого покрытия не существует, то у G не существует и независимого множества размера g.

Задача о независимом множестве $\longrightarrow$ задача о клике

Задача о независимом множестве $\longrightarrow$ задача о клике

- Определим дополнение (complement) графа $G(V, E)$ как граф $\bar{G} = (V, \bar{E})$, где $\bar{E}$ содержит ровно те неупорядоченные пары вершин, которые не принадлежат E.

Задача о независимом множестве $\longrightarrow$ задача о клике

- Определим дополнение (complement) графа $G(V, E)$ как граф $\bar{G} = (V, \bar{E})$, где $\bar{E}$ содержит ровно те неупорядоченные пары вершин, которые не принадлежат E.
- Тогда подмножество вершин S является независимым в G тогда и только тогда, когда S является кликой в $\bar{G}$. Другими словами, эти вершины попарно не соединены в G тогда и только тогда, когда любые две из них соединены ребром в $\bar{G}$.

Задача о независимом множестве $\longrightarrow$ задача о клике

- Определим дополнение (complement) графа $G(V, E)$ как граф $\bar{G} = (V, \bar{E})$, где $\bar{E}$ содержит ровно те неупорядоченные пары вершин, которые не принадлежат E.
- Тогда подмножество вершин S является независимым в G тогда и только тогда, когда S является кликой в $\bar{G}$. Другими словами, эти вершины попарно не соединены в G тогда и только тогда, когда любые две из них соединены ребром в $\bar{G}$.
- Таким образом, можно свести задачу о независимом множестве к задаче о клике, сопоставив каждому входу (G, g) вход $(\bar{G}, g)$.

Любая задача класса NP $\longrightarrow$ SAT

Любая задача класса NP $\longrightarrow$ SAT

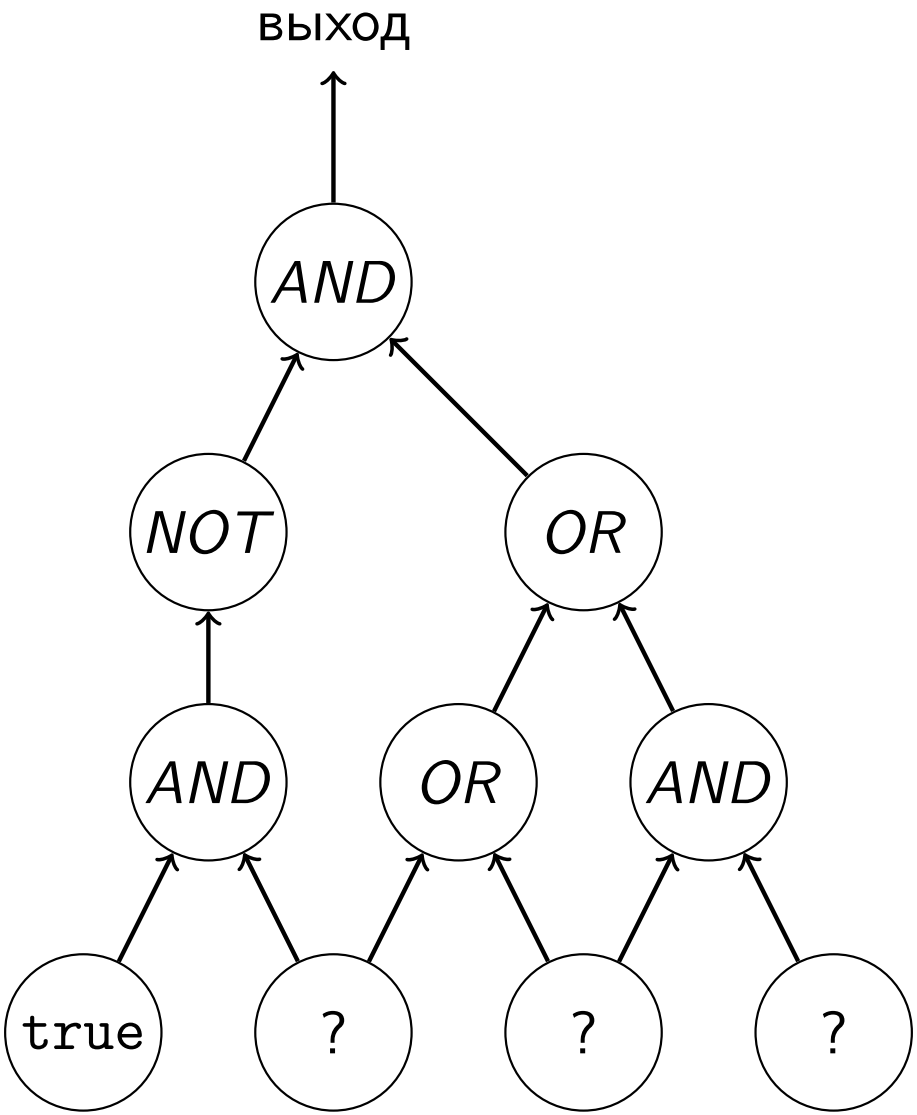
- Мы уже свели задачу SAT к нескольким задачам. Теперь мы замыкаем круг и показываем, что все эти задачи — в действительности все задачи класса NP — сводятся к SAT. Именно, мы покажем, что все задачи класса NP сводятся к обобщению задачи SAT, называемому задачей выполнимости булевой схемы (CIRCUIT-SAT).

Любая задача класса NP → SAT

- Мы уже свели задачу SAT к нескольким задачам. Теперь мы замыкаем круг и показываем, что все эти задачи — в действительности все задачи класса NP — сводятся к SAT. Именно, мы покажем, что все задачи класса NP сводятся к обобщению задачи SAT, называемому задачей выполнимости булевой схемы (CIRCUIT-SAT).
- В данной задаче на вход дается булева схема (Boolean circuit), то есть ориентированный ациклический граф, вершинами которого являются гейты (gates) пяти следующих типов:
  - гейты AND и OR имеют входящую степень 2;
  - гейты NOT имеют входящую степень 1;
  - константные входные гейты имеют входящую степень 0 и помечены либо `true`, либо `false`;
  - переменные входные гейты имеют входящую степень 0 и помечены “?”.

  Один из стоков графа называется выходным гейтом.

Пример схемы



Задача выполнимости схемы

Задача выполнимости схемы

- По данному набору значений переменных входных гейтов мы можем вычислять значения в гейтах последовательно в топологическом порядке, используя правила булевой логики (такие, как $\texttt{false} \vee \texttt{true} = \texttt{true}$), пока не получим значение на выходном гейте. Это будет значением схемы на данном наборе значений входным переменным.

Задача выполнимости схемы

- По данному набору значений переменных входных гейтов мы можем вычислять значения в гейтах последовательно в топологическом порядке, используя правила булевой логики (такие, как $\texttt{false} \vee \texttt{true} = \texttt{true}$), пока не получим значение на выходном гейте. Это будет значением схемы на данном наборе значений входным переменным.
- Задача выполнимости булевой схемы заключается в проверке существования значений входных переменных заданной схемы, при которых значение схемы есть `true`.

SAT $\leftrightarrow$ CIRCUIT-SAT

SAT $\leftrightarrow$ CIRCUIT-SAT

- SAT $\longrightarrow$ CIRCUIT-SAT: понятно.

SAT $\leftrightarrow$ CIRCUIT-SAT

- SAT $\rightarrow$ CIRCUIT-SAT: понятно.
- Обратно: запишем схему в виде формулы в КНФ. Для каждого гейта g мы заводим переменную g и моделируем этот гейт добавлением клозов следующим образом.

гейт g	соответствующие клозы
`true`	(g)
`false`	$(\neg g)$
$g = h_1$ OR h_2	$(g \vee \neg h_1)(g \vee \neg h_2)(\neg g \vee h_1 \vee h_2)$
$g = h_1$ AND h_2	$(\neg g \vee h_1)(\neg g \vee h_2)(g \vee \neg h_1 \vee \neg h_2)$
$g =$ NOT h	$(g \vee h)(\neg g \vee \neg h)$

Наконец, для выходного гейта g мы добавляем клоз (g), чтобы учесть тот факт, что нас интересует набор значений переменных, при которых на выходе схемы получается `true`.

Сведение

Сведение

- Теперь, когда мы знаем, что CIRCUIT SAT сводится к SAT, мы возвращаемся к нашей основной задаче — показать, что все задачи поиска сводятся к CIRCUIT SAT.

Сведение

- Теперь, когда мы знаем, что CIRCUIT SAT сводится к SAT, мы возвращаемся к нашей основной задаче — показать, что все задачи поиска сводятся к CIRCUIT SAT.
- Итак, предположим, что задача A принадлежит классу NP. Мы должны построить сведение A к CIRCUIT-SAT. Это кажется довольно сложным, поскольку мы не знаем практически ничего про A.

Сведение

- Теперь, когда мы знаем, что CIRCUIT SAT сводится к SAT, мы возвращаемся к нашей основной задаче — показать, что все задачи поиска сводятся к CIRCUIT SAT.
- Итак, предположим, что задача A принадлежит классу NP. Мы должны построить сведение A к CIRCUIT-SAT. Это кажется довольно сложным, поскольку мы не знаем практически ничего про A.
- Всё, что мы знаем про A, это то, что A является задачей поиска, поэтому мы должны это как-то использовать. Основное свойство задачи поиска заключается в том, что любое решение может быть быстро проверено: существует алгоритм $\mathcal{C}$, который по входу I и кандидату на решение S проверяет, действительно ли S является решением I. Более того, $\mathcal{C}$ делает это за полиномиальное время от длины записи I (можно считать, что S дано как битовая строка длины, ограниченной полиномом от длины I).

Преобразование алгоритма в схему

Преобразование алгоритма в схему

- Каждый алгоритм может быть преобразован в схему (точнее, в семейство схем), входные переменные гейты которой кодируют вход алгоритма.

Преобразование алгоритма в схему

- Каждый алгоритм может быть преобразован в схему (точнее, в семейство схем), входные переменные гейты которой кодируют вход алгоритма.
- Размер каждой такой схемы будет полиномиальным от количества входных переменных гейтов. Если исходный алгоритм решает задачу, ответ на которую есть один бит информации (так и есть в ситуации с $\mathcal{C}$: "является ли S решением для I"), то этот ответ будет выдаваться на выходном гейте схемы.

Преобразование алгоритма в схему

- Каждый алгоритм может быть преобразован в схему (точнее, в семейство схем), входные переменные гейты которой кодируют вход алгоритма.
- Размер каждой такой схемы будет полиномиальным от количества входных переменных гейтов. Если исходный алгоритм решает задачу, ответ на которую есть один бит информации (так и есть в ситуации с $\mathcal{C}$: "является ли S решением для I"), то этот ответ будет выдаваться на выходном гейте схемы.
- Итак, по данному входу I задачи A мы строим за полиномиальное время схему, известными входными гейтами которой являются биты I, а неизвестными — биты S, такую что выход схемы равен `true` тогда и только тогда, когда переменные входные гейты схемы содержат решение S условия I. Другими словами, выполняющие наборы схемы находятся во взаимно-однозначном соответствии с решениями условия I задачи A. Сведение завершено.

Спасибо за внимание!