NP классы, сведения, различные другие классы алгоритмов

«Будет вообще уморительно, если кто-то сядет и скажет: «Ох ё, поиск Гамильтонова цикла это просто динамика за квадрат». И полстраницы кода...»

Глеб

В прошлый раз мы поговорили о просто задаче SAT. Теперь у нас есть мощный инструмент к сведению сложных на данное время задач. Первая из них — 3-SAT — булева формула в конъюктивно нормальной форме, где каждый дизъюнкт содержит не более 3 литералов (ну или ровно 3, мы докажем эквивалентность чуть позже).

Сведение SAT к 3-SAT

Теорема 1. *$SAT \leqslant_p$ 3-SAT.*

Доказательство. Возьмём один дизъюнкт и сделаем из него много дизъюнктов с количеством литералов не более 3.

Действительно, пусть у нас будет дизъюнкт $(x_1 \vee \ldots \vee x_k)$ — возможно с отрицаниями, нам не важно. Разобьём на 2 примерно равных множества этот дизъюнкт — в одном $\left\lfloor \frac{k}{2} \right\rfloor$ литералов, в другом $\left\lceil \frac{k}{2} \right\rceil$. Добавим новую переменную x_{n+1}, тогда покажем эквивалентность $(x_1 \vee \ldots \vee x_{\lfloor k/2 \rfloor} \vee x_{n+1}) \wedge (x_{\lfloor k/2 \rfloor + 1} \vee \ldots \vee x_k \vee \overline{x_{n+1}})$. Действительно, если эти 2 дизъюнкта выполнены, то в одном из них литерал с x_{n+1} равен 0, значит один из литералов в множестве $x_1, \ldots, x_k$ равен 1 и изначальный дизъюнкт выполнен.

В другую сторону — если изначальный дизъюнкт выполнен, тогда существует какой-то литерал из $x_1, \ldots, x_k$, который равен 1, тогда поставим x_{n+1} так, что оно равно 0, там где литерал из $x_1, \ldots, x_k$ равен 1. Тогда обе скобки будут равны единице (там, где есть литерал, который равен 1, будет всегда 1, а в другой скобке литерал с x_{n+1} равен 1).

Будем так делать для каждого дизъюнкта, добавляя новую (обязательно не совпадающую с предыдущими созданными) переменными. И будем останавливаться, когда дизъюнкт содержит не более 3 литералов. Заметим, что из дизъюнкта, состоящего из 3 литералов нельзя сделать 2 дизъюнкта со строго меньшим количеством литералов. Легко проверить, что такая процедура работает только при $k \geqslant 4$.

Осталось совсем немного — доказать, что 3-$SAT \in \mathsf{NP}$ и сведение действительно полиномиально. Первое совсем очевидно, так как 3-SAT это частный случай SAT, а про SAT мы точно знаем, что эта задача из класса NP.

Заметим, что количество уровней при процедуре с одним дизъюнктом будет не более, чем $O(\log n)$ (это легко показать, сказав и проверив, что если проделать 2 уровня, то максимальная длина дизъюнкта уменьшается хотя бы в 2 раза). И на каждом уровне мы создаём не более 2^k новых литералов. Если всё просуммировать, получим линейное сведение одного дизъюнкта. Для остальных проделаем то же самое. $\square$

Также стоит отметить по лемме в прошлой лекции следует, что 3-$SAT \in \mathsf{NPC}$.

Пока никто не умеет сводить SAT к 2-SAT, так как последняя решается за линейное время.

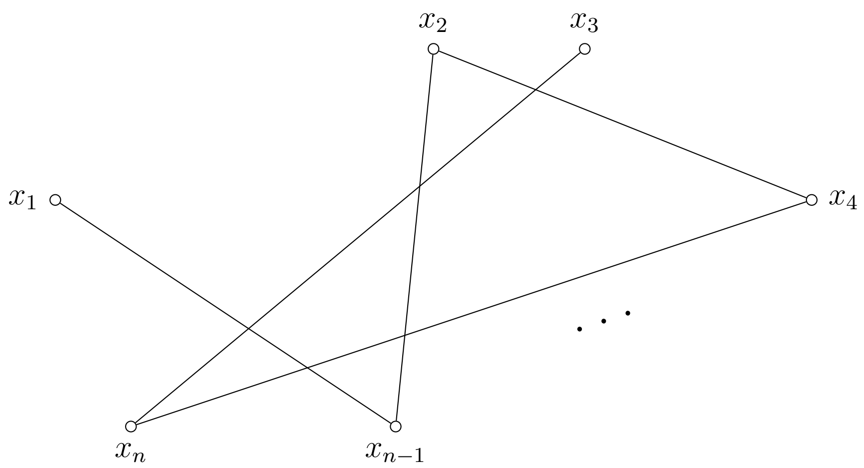
Байка от Глебаса: Только испанские составители контестов могут это делать. Писали испанский контест. Читаешь условие — вот просто дана задача о рюкзаке. $n \leqslant 50000, a_i \leqslant 10^9$, TL 2 секунды. Зашли в админку, увидели супер-эвристику, долго ржали, возможно, даже отослали решение, но не помню. Ну также они дали задачу на гамильтонов путь на 200 вершинах.

NP-полнота задач клики, доминирующего множества и вершинного покрытия

Ну теперь мы займёмся сведениями. Мы много говорили, что некоторые задачи на графах очень сложны. Теперь надо ответить за базар:

Теорема 2. *Задача «существует ли клика размера k в графе $G(V, E)$» является* NPC.

Доказательство. Сведем эту задачу к SAT (просто красивое рассуждение). Потом сделаем в другую сторону.



Переменные будут отвечать за вершины. Соорудим нашу формулу:

Любые 2 вершины, между которыми нет ребра, не могут быть взяты обе.

- $(\overline{x_v}, \overline{x_u})$ при всех $(v, u) \notin E$.

Введём такую величину — d_{ij}, отвечающую на вопрос, верно ли, что среди первых i вершин есть клика размера j. Заметим, что $d_{ij} = d_{i-1,j} \vee (d_{i-1,j-1} \wedge x_i)$, то есть либо есть клика размера k среди первых $i - 1$ вершины, либо среди первых $i - 1$ есть клика размера $j - 1$ и взята вершина i.

- $(\overline{d_{ij}} \vee d_{i-1,j} \vee d_{i-1,j-1}) \wedge (\overline{d_{ij}} \vee d_{i-1,j} \vee x_i)$ — если $d_{i-1,j} = 0$ и $x_i = 0$, тогда точно $d_{ij} = 0$ и выполняются оба дизъюнкта. Если $d_{i-1,j} = 0$ и $d_{i-1,j-1} = 0$, то $d_{ij} = 0$ в обоих случаях. Это делаем при $1 \leqslant i \leqslant n; 1 \leqslant j \leqslant k$.
- $(d_{ij} \vee \overline{d_{i-1,j}}) \wedge (d_{ij} \vee \overline{d_{i-1,j-1}} \vee \overline{x_i})$. Это те же условия, только мы здесь хотим сделать $d_{ij} = 1$, если выполнено хотя бы одно условие. Это делаем при $1 \leqslant i \leqslant n; 1 \leqslant j \leqslant k$.

Начальные условия (среди первых):

- $d_{0j} = 0$ при $1 \leqslant j \leqslant k$ — среди нуля вершин нет клики размера хотя бы 1.
- $d_{i0} = 1$ при $0 \leqslant i \leqslant n$ — среди первых i вершин есть клика размера 0.

Финальное состояние:

- (d_{nk}) — ответ на задачу.

Легко видеть, что это и есть SAT, притом формула выполняется тогда и только тогда, когда есть клика размера k. Причём сведение, очевидно, полиномиально.

Теперь в другую сторону:

Рассмотрим любую SAT формулу. Пусть у нас есть k дизъюнктов. Выпишем всех их (каждый литерал — отдельная вершина) в виде графа. Внутри дизъюнкта вершины не будем соединять, а также не будем соединять вершины в различных дизъюнктах, которые отвечают сразу за x_i и $\overline{x_i}$. Теперь запустим решение задачи о поиске клики размера k. Если решение нашлось, то в каждой части графа, отвечающего за отдельный «дизъюнкт», выбрана ровно 1 вершина. Иначе в каком-то дизъюнкте выбрано 2, а мы не соединяли вершины в одном и том же дизъюнкте. Поэтому в каждом дизъюнкте выбран ровно 1 литерал. Сделаем эти литералы равными единице. Те переменные, которые мы не выбрали, положим единице. Заметим, что мы не сделали одновременно x_i и $\overline{x_i}$ равными единице, так как иначе между ними было бы ребро. А мы договорились, что такого ребра нет.

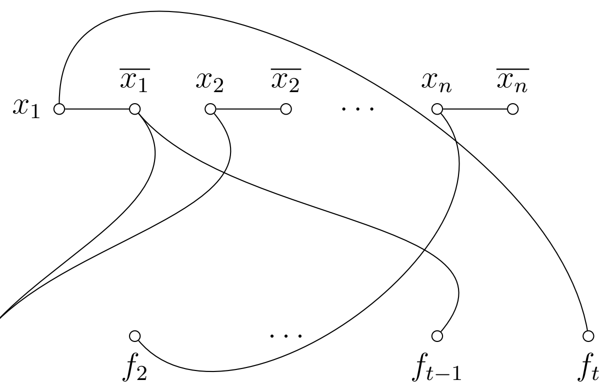
То, что любому решению SAT формулы соответствует какая-то клика в построенном графе следует из почти дословных рассуждений выше, что завершает док-во, что задача про поиск клики данного размера лежит в NPC. $\square$

Теперь поговорим про задачу о доминирующем множестве размера k. Напомним, что мы хотим в данной задаче найти k вершин так, что оставшиеся вершины соединены хотя бы с одной из выбранных вершин. Докажем следующее сведение:

Теорема 3. *$Dominant\_set \in \mathsf{NPC}$.*

Доказательство. Сведём SAT к этой задаче.

Пусть $f_1, \ldots, f_k$ — дизъюнкты в формуле SAT. Тогда построим следующий граф:



Соединим литералы с теми дизъюнктами, куда он входит, а также соединим рёбрами x_i и $\overline{x_i}$.

Обязательно надо сделать $n+1$ копию каждого f_i и соединить их также, как я показал на графе.

Теперь докажем, что доминирующее множество размера n в графе будет соответствовать решению и наоборот.

Если есть решение булевой формулы в КНФ, то если $x_i = 1$, возьмём в доминирующее множество вершину x_i, иначе $\overline{x_i}$. Заметим, что если формула выполняется, то в каждом дизъюнкте хотя бы один литерал равен 1. Заметим, что мы его взяли в доминирующее множество. Также все $x_i, \overline{x_i}$ будут покрыты какой-то вершиной, так как мы берем точно одно из двух.

В обратную сторону. Пусть нам было сказано, что доминирующее множество размера n существует. Заметим, что если при фиксированном i ни одна вершина из пары вершин $x_i, \overline{x_i}$ не взята в доминирующее множество, то в доминирующее множество взято какое-то из f_k (пусть оно соединено с x_i). Все f_k сразу быть взяты не могли (их $n+1$), значит существует f_k, которое не взяли в доминирующее множество. Тогда существует x_k или $\overline{x_k}$, которое соединено с этим «не взятым» f_k, а значит оно соединено со всеми f_k, значит мы можем не брать никакое f_k в доминирующее множество, а взять x_i. Тогда не нарушится свойство доминирующего множества и элементов мы возьмём ровно n. Так можно избавиться от всех вершин в доминирующем множестве, которые являются вершинами f_k. Значит существует доминирующее множество на n вершинах только из x_i и $\overline{x_i}$. Если обе вершины при фиксированном i взяты, то по принципу Дирихле существует пара $x_j, \overline{x_j}$, из которой не взяли ни одну вершину в доминирующее множество. А этот случай см. выше.

Теперь тем литералам, которые мы взяли, поставим единицу. Ясно, что они друг другу не противоречат (см. выше) и каждый дизъюнкт будет выполнен, так как эти n вершин образуют доминирующее множество, то есть каждый дизъюнкт соединен с какой-то вершиной из доминирующего множества.

Понятно, что мы провели полиномиальное сведение от размера КНФ.

Ясно, что по множеству вершин легко проверить, является ли оно доминирующим (например, если взять матрицу смежности и проверить каждую вершину за полином). Поэтому мы свели NPC задачу к этой, а эта задача оказалась из NP, значит эта задача является NP-трудной. □

Теперь поговорим о вершинном покрытии графа размера k. Вспомним, что вершинное покрытие это множество вершин такое, что любое ребро инцидентное хотя бы одной вершине из множества. Неудивительно, эта задача тоже NPC.

Теорема 4. $Vertex\_cover \in \mathsf{NPC}$.

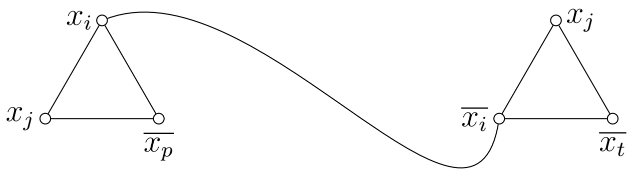
Доказательство. Сведём 3-SAT к этой задаче.

Ну здесь как раз нам и понадобится факт, что в любой КНФ, где дизъюнкты имеют размер не более 3, можно сделать ровно 3.

Добавим 3 переменных x, y, z. Мы хотим их сделать всегда $false$. Давайте напишем 7 дизъюнктов длины 3 с этими переменными, кроме одного — $(x \vee y \vee z)$. Если хотя бы одна переменная равна 1, то несложно убедиться, что формула будет равна 0 (просто перебор). Значит все равны 0. То есть решения существуют тогда и только тогда, когда $x = y = z = 0$. Поэтому их не жалко добавлять в те дизъюнкты, где не хватает литералов до количества 3.

Теперь каждый дизъюнкт содержит ровно 3 переменных.

Теперь построим такой граф: каждый дизъюнкт отвечает треугольнику, причем в различных треугольниках мы соединяем вершины, соответствующие x_i и $\overline{x_i}$. Проиллюстрируем это рисунком для любых 2 различных треугольников:



Теперь, если есть решение формулы, давайте докажем, что у нас есть решение задачи о вершинном покрытии размера $2k$, где k — количество дизъюнктов.

По решению выберем в каждом треугольнике, где значение литерала равно 1. И отметим 2 другие вершины в качестве покрытия. Докажем, что это действительно вершинное покрытие. Ясно, что в каждом треугольнике все рёбра будут инцидентные какой-то вершине, так как выбраны ровно 2 вершины. Осталось разобраться с ребрами между x_i и $\overline{x_i}$. Если ни одна вершина не покрывает это ребро, тогда и x_i, и $\overline{x_i}$ были равны 1, что невозможно. Значит мы нашли вершинное покрытие размера $2k$.

Обратно. Пусть у нас есть покрытие размера $2k$, тогда в каждом треугольнике выбрано не меньше 2 вершин, иначе какое-то ребро не будет покрыто вершиной. С другой стороны их не больше 2, так как иначе по принципу Дирихле найдётся треугольник, в котором покрыто меньше 2 вершин.

Теперь возьмём во всех треугольниках и сделаем литерал равным единице, который не лежит в этом покрытии. Остальным переменным, которые мы не использовали, выставим 1 (с ними всё корректно, мы их не использовали). Осталось понять, что мы корректно выставили всем переменным значения. Если мы вдруг захотели выставить $x_i = 1$ и $\overline{x_i} = 1$, то эти обе вершины не лежали в вершинном покрытии, а между ними есть ребро, поэтому вершинное покрытие было некорректным.

Осталось проверить, что эта задача из NP. Действительно, легко по множеству вершин определить, является ли это множество вершинным покрытием (надо просто просмотреть все рёбра). Значит эта задача является NPC. □

Другие классы алгоритмов

Все мы знаем, что проблема останова невычислима. Все классы алгоритмов, которые считают, что проблема останова невычислима обозначают за H_0. Вот начинают рассматривать некоторые классы алгоритмов, при условии, что мы умеем решать проблему остановки. Этот класс обозначают за H_1, но это ещё не всё! Проблема остановки с оракулом проблемы остановки на обычной МТ тоже невычислима. И если уметь решать и эту проблему, то такие классы алгоритмов обозначают за H_2 и так далее. Нужно ли это кому-то? Вряд ли. Какая разница, если мы уже не умеем решать проблему остановки, то зачем рассматривать случаи, когда мы умеем это делать? -Непонятно, но знать об этом стоит.

Определение 1. *Класс L — это те МТ, которые работают с $\mathcal{O}(\log n)$ дополнительной памятью. Легко показать, что $L \subseteq P$ (оставим, как упражнение читателю).*

Пример задачи — узнать длину строки на входе. Нам нужно все $\mathcal{O}(\log n)$ бит, чтобы закодировать длину на входе длины n.

Определение 2. *Класс* BPP *(от англ. Bounded-Error Probabilistic Polynomial) те языки, для которых существует недетерминированная МТ (использующая генератор случайных чисел, МТ выбирает переход по таблице переходов с некоторой равной вероятностью), которые ошибаются с вероятностью не более* $\frac{1}{3}$*.*

Почему $\frac{1}{3}$? По схеме Бернулли, если $p < 1/2$, то мы можем быть сильно уверены после многократного запуска (например, $\mathcal{O}(n)$ вероятность будет сравнима с экспонентной от входа). Про $\frac{1}{3}$ просто договорились. Примеры таких алгоритмов — хэши.

Определение 3. *Класс* RP *(от англ. Randomized Polynomial) это те языки, для которых, если слово не принадлежит языку, то вероятность, что МТ допустит это слово равна 0. Если принадлежит, то вероятность не меньше* $\frac{1}{2}$*, что МТ допустит (мы опять рассматриваем недетерминированные МТ с генератором случайных чисел).*

Пример такого алгоритма может являться алгоритм Каргера-Штайна, который мы рассматривали в прошлом году.

Класс coRP определяется также, только поменяны местами выражения «принимает» и «не принимает».

Определение 4. *Класс* ZPP *(от англ. Zero-Error Probabilistic Polynomial) это языки, для которых существует вероятностная МТ, которая всегда отвечает правильно и математическое ожидание времени работы полиномиально.*

Упражнение читателю: $\mathsf{ZPP} = \mathsf{RP} \cap \mathsf{coRP}$.

Зачем мы приводим здесь все эти классы? При приёме в аспирантуру всегда есть вопрос про классы алгоритмов, поэтому просто полезно об этом знать.

На этом наш курс подошёл к концу.