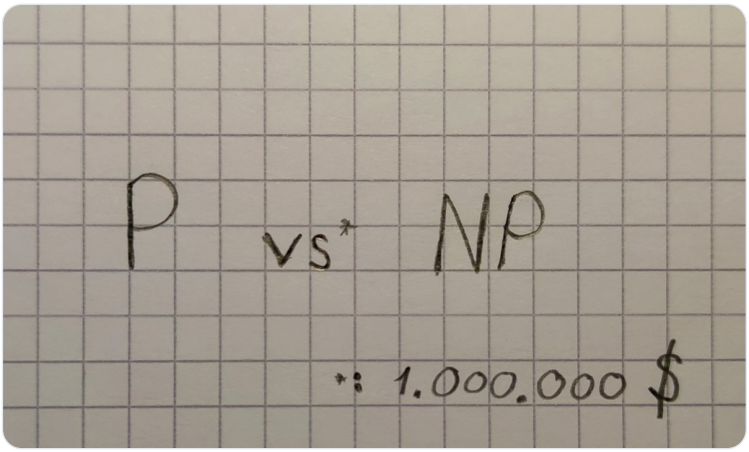
Валерий Жила @ValeriiZhyla
Nov 2, 2021 • 78 tweets • ValeriiZhyla/status/1455462831420215296

Я расскажу простым языком об одном из самых важных вопросов науки, который ставит под сомнение все, на чем основывается сегодняшнее IT.

Встречайте - вопрос о равенстве классов P и NP!



Мы поговорим про эти два класса, про смысл самого вопроса об их равенстве, и чем чреват для нас ответ на него.
Я продемонстрирую две очень похожие задачи, принадлежащие этим двум классам, и покажу ключевую разницу между ними.
Будет интересно)

Примечание автора: ретвит треда вознаграждается Вселенной в этой жизни и всех последующих бытиях

P vs NP – одна из шести нерешенных задач тысячелетия.

Что это вообще за задачи такие?
Если коротко - особенно сложные и важные математические проблемы, за которые одним американским университетом назначена премия по миллиону долларов за штуку плюс мировая известность бонусом.

Что важно знать – об эту проблему ломает зубы уже третье поколение очень мотивированных ученых.

Какая преследуется цель?
Нужно доказать их равенство или неравенство?

Науку интересует оба варианта!
Тем более, что они взаимоисключающие.
Так что цель состоит в том, чтоб доказать, что P = NP, или же P != NP

Начнем с простого вопроса - что такое эти ваши классы сложности? Что вообще обозначают эти названия?

Буквы какие стрёмные, мутные какие-то, ещё и проблемы науки загадочные, напрочь оторванные от реальности...

Итак, класс сложности P.
Класс P (от слова Polynomial), он же класс PTIME - это класс задач, требующих полиномиального времени.

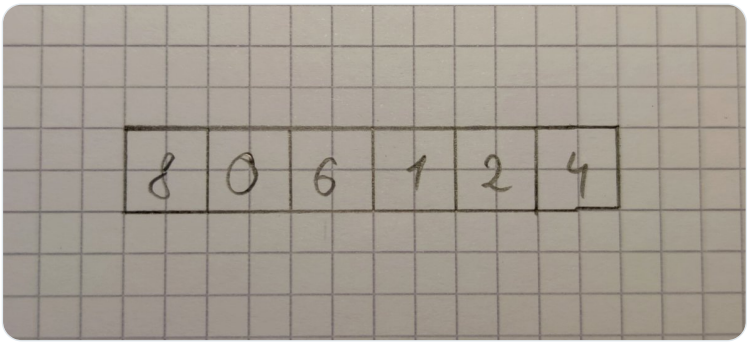
- Что такое полиномиальное время, дед, ты таблетки забыл?

Предлагаю начать с минутки ностальгии.
Полином это то, что мы в школе называли многочленом. Шутки про члены и Вовочку с ножичком, припоминаете?
Например, x^2 + 1 - многочлен.

Когда говорят про полиномиальное время, обычно имеют в виду, что на выполнение программы с N объектами нужно N^(какое-то число) операций.
На самом деле, практически все алгоритмы, которые мы широко используем в своих программах - полиномиальные.

Например, если мы хотим самое большое число из массива чисел, нам нужно их всех перебрать, по одному и сравнить с каким-то числом. Тобишь на N чисел нам нужно N операций доступа и N операций сравнения - получаем 2N операций, оно же 2*N^1.

Например, программе поиска самого большого числа нужно будет пройтись по всем ячейкам ровно одни раз, даже если число это стоит в самом начале.



Константы в умножении обычно отбрасываются, и в таком случае обычно пишут просто N.

Простые алгоритмы сортировки сравнивают каждое число со всеми остальными. На каждое из N чисел у нас будет N-1 сравнений.
Их сложность составляет N*(N-1), что практически равно N*N = N^2.

В примере с массивом сверху, восьмерку нужно сравнить с нулем, с шестрекой, с единицей, двойкой и четверкой. Ноль нужно сравнить с шестеркой, единицей, двойкой, четверкой. И так далее.

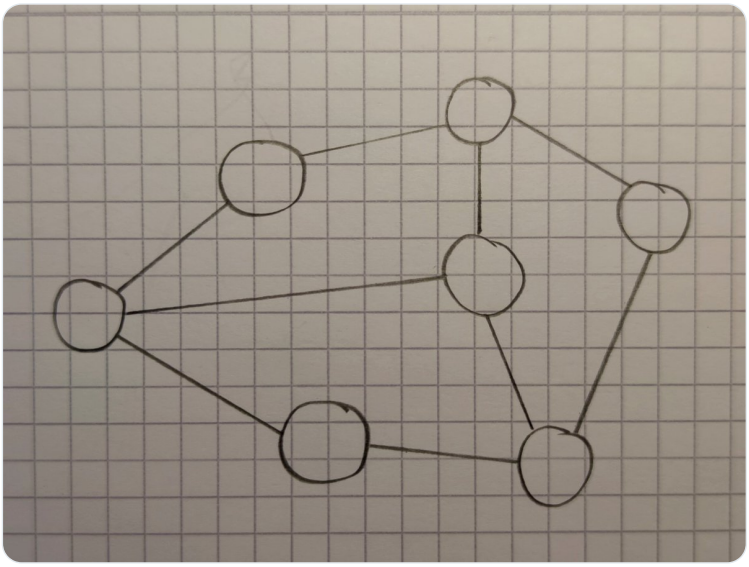
Сортировки разного пошиба, поиск в массиве, поиск по графу, поиск пути в графе, проверки строк - все это алгоритмы, которые входят в класс P.
Всё, что вы видели на литкоде, все, что спросят на собесах - однозначно алгоритмы из класса P.
(Я не гуру литкода, это предположение)

Предлагаю разобраться с одной конкретной задачей из класса P, а то я заладил про элементарные поиски в массиве - скучно, тухло, душно.

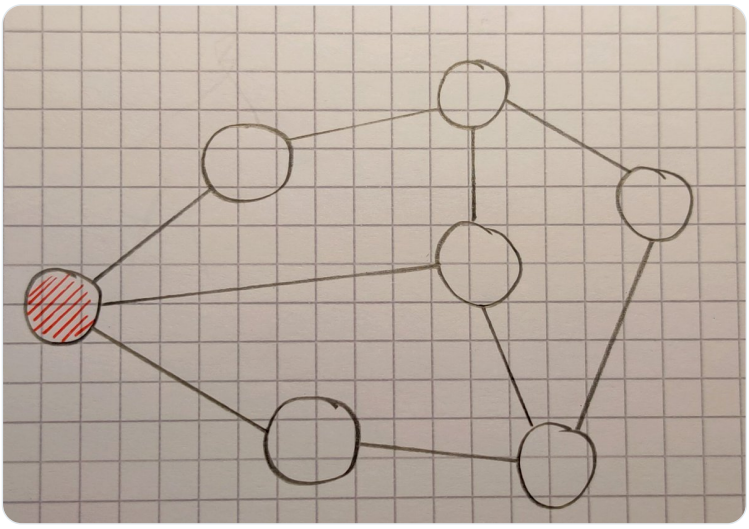
Это довольно простая задача об окраске графа в два цвета так, чтоб каждое ребро соединяло вершины с разными цветами.

Если что - граф это стуктура из кружочков, соединенных между собой.
Кружочек называется вершиной, а соединения - ребрами

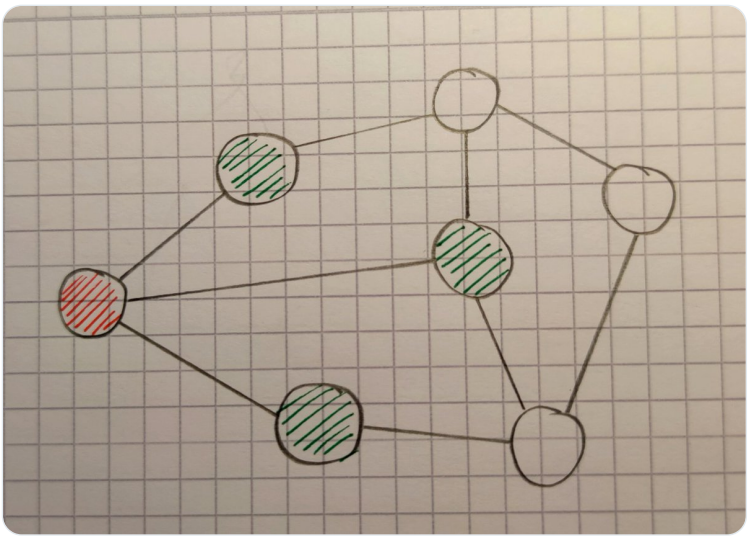
Задача про окраску в два цвета, называется довольно логично - 2COLOR.
Давайте красить вот этот милый граф в красный и зеленый.



Алгоритм очень прост.
Для начала, в произвольном месте графа красим вершину в красный:



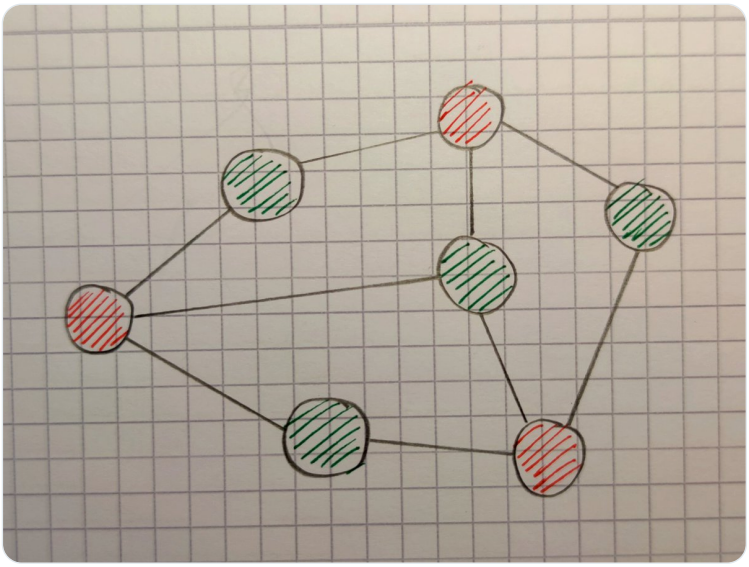
И всех ее неокрашенных соседей – в зеленый



Как видно по моей окраске, маляр из меня так себе. Но не в этом суть!

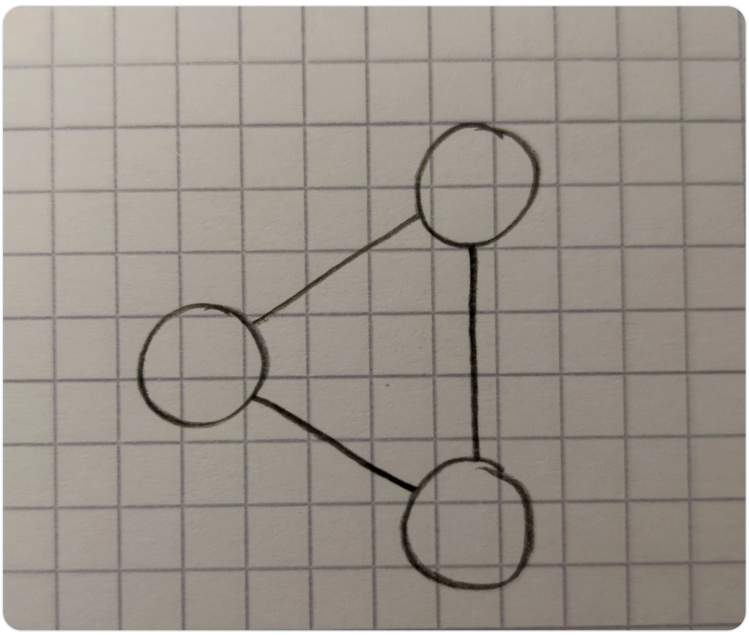
Двигаемся по графу, окрашивая всех неокрашенных соседей зелёных в красный. Если один сосед уже красный – всё в порядке, а если зеленый – ошибка, всё, граф неокрашиваем в два цвета нужным нам образом.

Выглядеть результат будет вот так:



Неокрашенных вершин не осталось – поздравляю, 2COLOR для этого графа решен. Прелесть ещё и в том, что есть только два возможных решения – если бы мы начали с зелёного цвета, мы бы получили второе из них. А можно просто взять первое решение и "перевернуть" все цвета

Вот простой граф, который нельзя окрасить в два цвета так, чтоб каждое ребро соединяло разные цвета. Просто, правда?



Компьютер легко справится даже со сложным и запутанным графом! На глаз сложность будет около N^2 для графа с N вершинами и с очень большой связностью (когда каждая вершина соединена с кучей других).

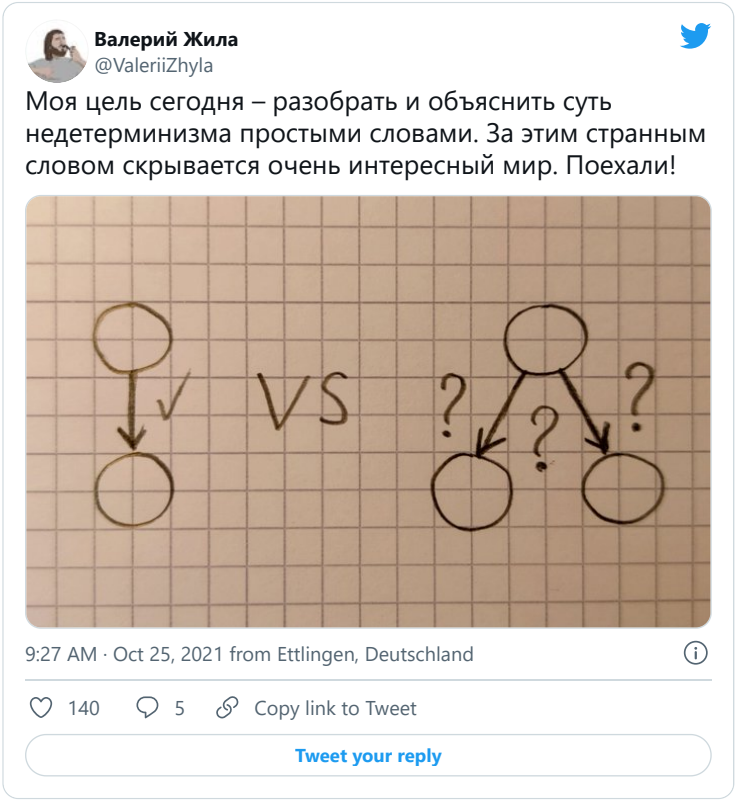
Где бы мы не начали считать, мы получим решение, если оно существует, а набор правил очень прост. Это классика класса P, и этот пример отлично показывает свойства этого класса.
Конечно, для сортировки на сервере сложность N^10 была бы катастрофой, но она все еще была бы в P!

Переходим к более серьезному бойцу - класс сложности NP.
Угадайте, что значит NP! Non-Polynomial? Мимо.
Notyourfucking Problem? Снова мимо!

NP - это Nondeterministic Polynomial, задачи, требующие полиномиального времени на недетерминистичном вычислителе.

- Дед, ты явно не пил сегодня таблетки.

Секунду, секунду, сейчас всё будет.
В классе P подразумевался обычный, детерменистичный вычислитель. Все наши компьютеры детерменистичны. Что такое недетерминизм?
Ну, на эту тему у меня был целый тред!



Но давайте всё немножечко упростим.
Наш обычный компьютер, в свою очередь, за одну операцию может сделать...то, что мы представляем под одной операцией.
Умножить, сложить, сравнить, и так далее.

Очень-очень упрощенно, и очень грубо, и вообще некорректно, но давайте назовем воображаемый квантовый компьютер недетерминистичным вычислителем.
И очень упрощенно опишем его, как компьютер, который одновременно просчитывает все возможные пути решения проблемы.

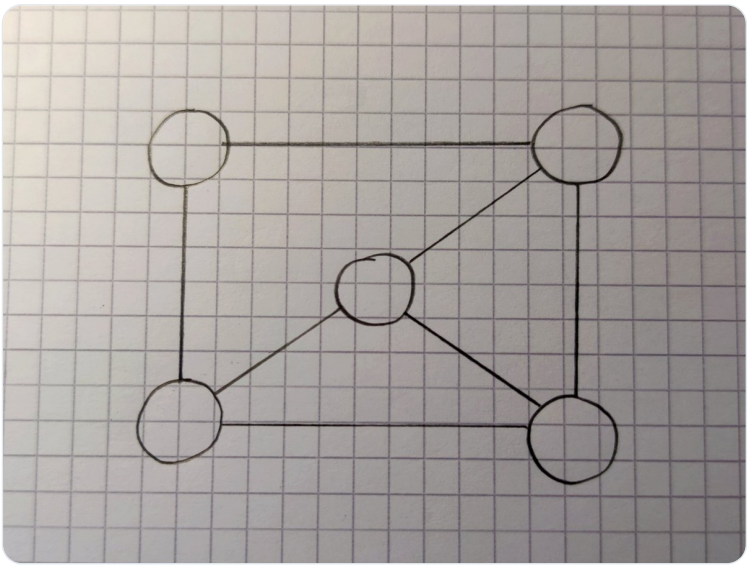
В отличии от обычного компьютера, который параллельно просчитывает только один путь решения (или несколько, по количеству ядер, но это не важно в данном контексте).

И алгоритмы, требующие полиномиального времени на таком квантовом компьютере и составляют из себя класс NP.
Самое важное тут - заметить, что задачи из класса NP ощущаются заметно сложнее чем задачи из класса P.

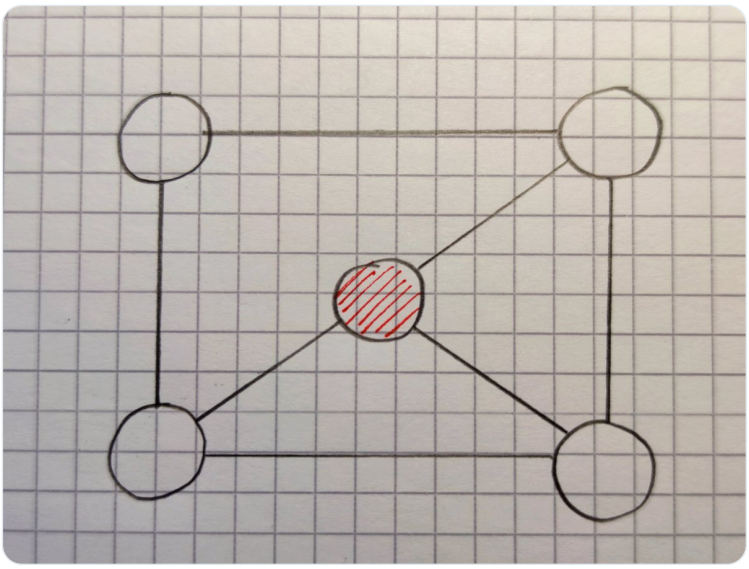
Предлагаю рассмотреть задачу из класса NP. Помните наш недавний 2COLOR? На сцену выходит старший брат, 3COLOR!
Задача 3COLOR крайне похожа: вершины графа нужно раскрасить тремя цветами так, чтоб каждое ребро соединяло вершины с разными цветами.

Был зеленый и красный, а теперь зеленый, красный и синий. Казалось бы, проблема не выглядит сложнее оригинала.
Сейчас вы все увидите!

Вот такой милаха у нас в наличии. Что ж, давайте покрасим одну из вершин в красный!



Теперь мы в сложной ситуации. Нам нужно покрасить соседей, но теперь у нас нет четкого правила, потому что у нас две возможности. Что мы можем сделать?



Во-первых, мы можем проверять перед покраской, не вызовет ли наш выбор цвета конфликтов в будущем, но тогда время выполнения улетит в экпоненциальный космос, и мы получим что-то в стиле N^N.

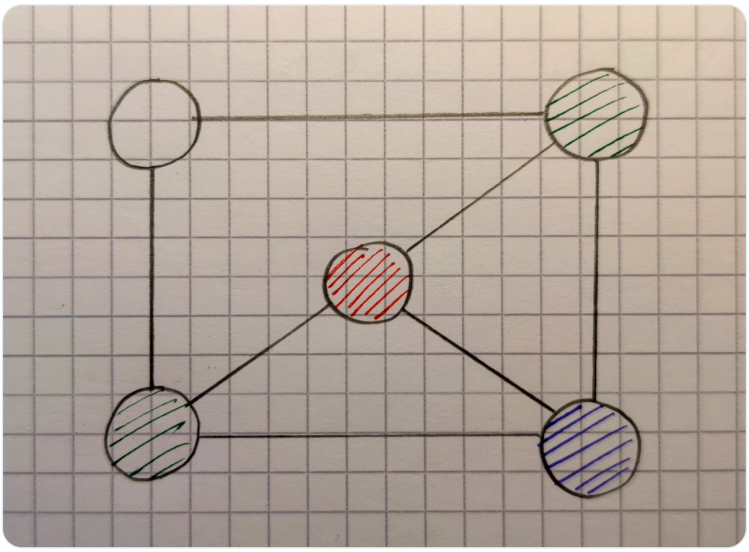
Если степень зависит от размера входных данных – это верный признак того, что мы пытаемся решить NP задачу детерминистическим путем.

Хорошо, что тут можно сделать? Неплохо выглядит решение, где программа запоминает то, что она делает по шагам, и в случае конфликта – откатывает изменения, пробуя другие комбинации. Если все комбинации не дали результата – отказываемся ещё дальше в прошлое, и пробуем другие

Если вам показалось, что такое решение перебором всех возможных комбинаций взорвется по сложности, то вам не показалось)
Оно именно так и поступит.

Этот алгоритм называется 3COLOR с бэктрекингом, и его на граф с N вершинами нужно до 3^N операций, по количеству всех возможных комбинаций из тред цветов. Помните, что я говорил про присутствие количества объектов не в основании, а в степени? Вот-вот.

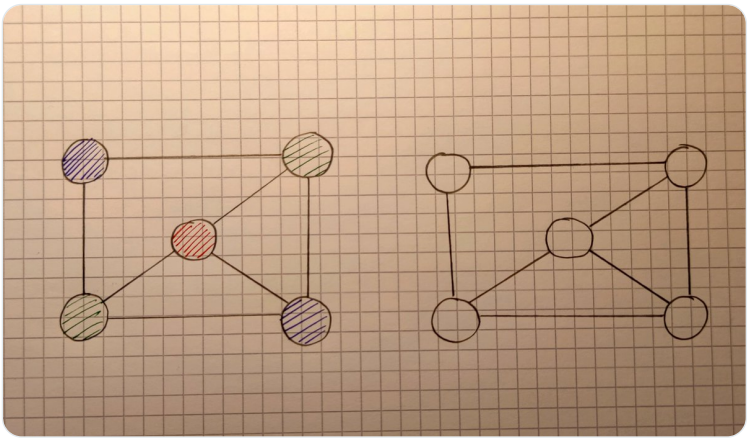
Как вся эта чехарда может выглядеть с нашим милым графом?
Например, можно сделать вот так:



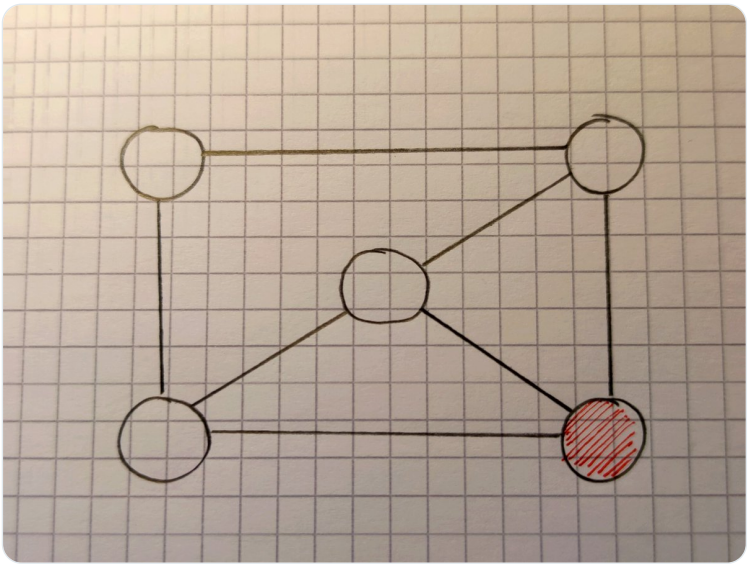
Этот граф довольно прост, и нам даже не нужно прибегать к бэктрекингу – уж в больно удачном месте мы начали. Вопрос – в какой цвет покрасить последнюю вершину?
Как вы догадались, и красный, и синий цвет будут верны. Даже на этом маленьком моменте возникает два правильных ответа!

Ещё одна характеристика NP проблем – очень часто у них есть много решений, которые вообще не зависят друг от друга (в отличие от решений 2COLOR, например)

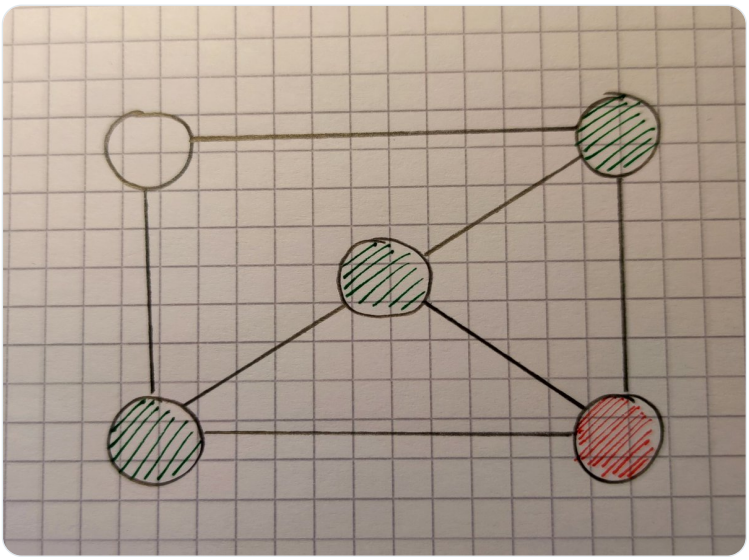
Покажу пример с бэктрекингом. Нарисуем рядышком такой же граф, и начнем с начала.



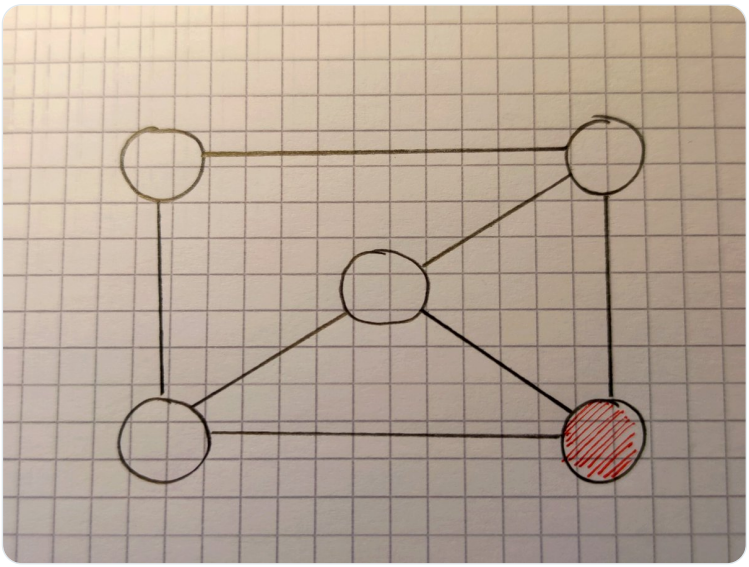
Начнем в другом месте



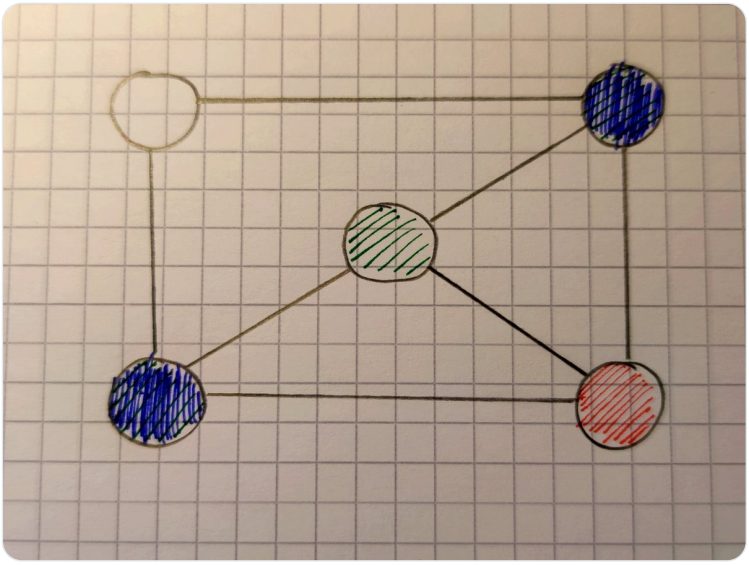
И примем решение покрасить всех соседей в зеленый! Наш алгоритм не проверяет наперед, и не знает, что они связаны между собой.



На этапе следующей проверки алгоритм увидит, что создал конфликт. Сделает шаг назад:



И попробует какую-то другую покраску.



Что мы видим? Мы знаем (из первой попытки, когда начали с центра), что правильное решение есть. Но у нас нет быстрого алгоритма его нахождения! И приходится все комбинации перебирать. Выйдет 3^N или за счет оптимизаций дерева кандидатов какой-то 3^logN – все равно кошмар.

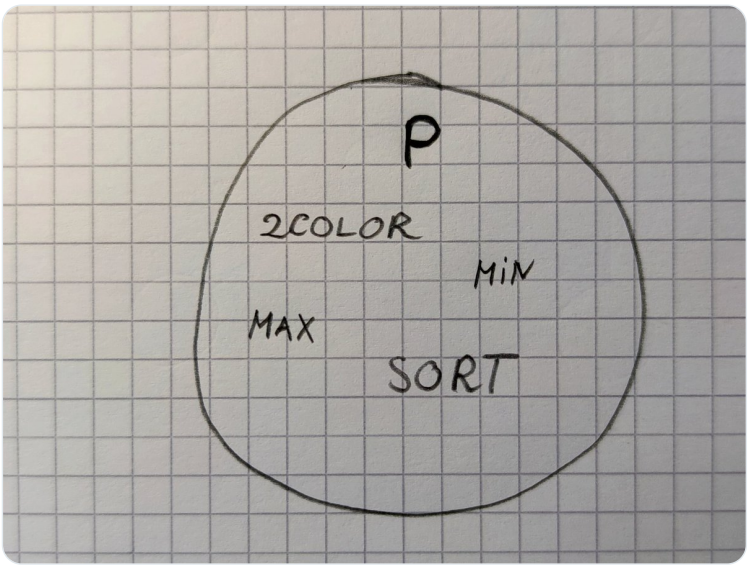
С 3COLOR вроде бы разобрались. Наш воображаемый квантовый копьютер, который как раз перебирает все возможные комбинации одновременно, справится с ней за N шагов, может за N^2, в любом случае, за человеческий полином.

Надеюсь, что это объяснение самих классов прошло без особых сложностей. Если они возникли - сейчас хорошая возможность задать вопрос!

Двигаемся дальше. Сложная часть позади)

Итак, хорошо, у нас есть класс P и класс NP. Если P=NP, то их связь ясна. А как они связаны, если допустить, что таки NP намного сложнее P?

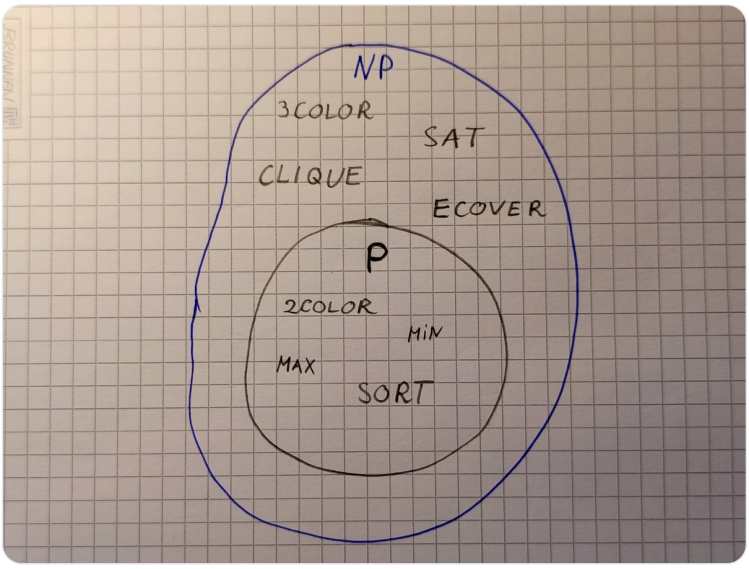
Давайте представим P как множество, содержащеё все проблемы из класса P. Вот такой вот мешочек, и несколько P проблем в придачу.



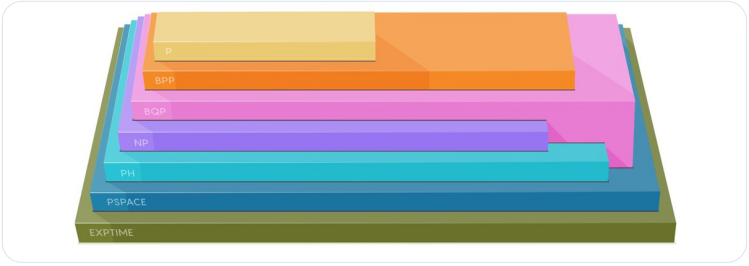
NP выглядит намного сложнее P. Но при этом, наш абстрактный квантовый компьютер может и массив отсортировать за N^2 и быстрее, ему достаточно просто не использовать своих супер-возможности. Машина, которая может решить 3COLOR, уж точно справится с примитивным 2COLOR.

Гроссмейстер уж точно справится со школьным конкурсом по шахматам среди младших классов. И так далее, поток аналогий можно прикрутить, пойду выпью таблетки, секунду.

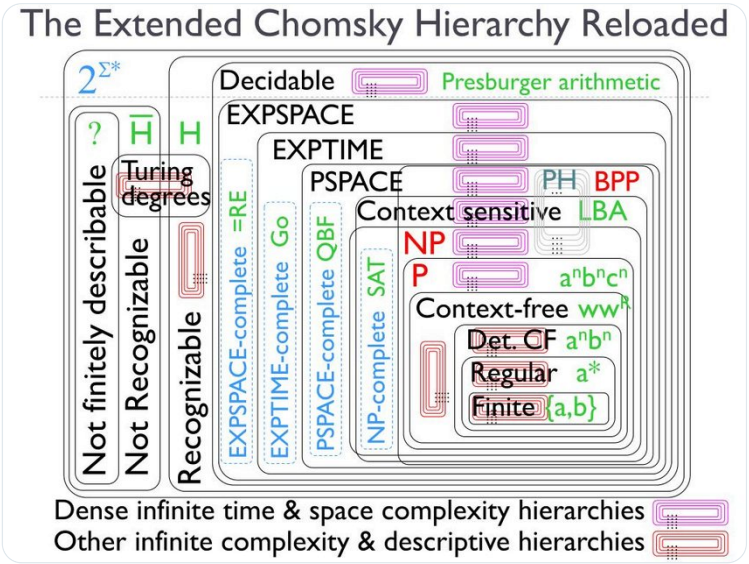
Из этого становится очевидно, что P при таком раскладе содержится в NP. Выглядит это примерно так:



Сделаю маленькое отступление и скажу, что NP это только начало классов сложности, но далеко не предел. Вот так выглядит кусочек иерархии классов сложности:



А вот его более полная версия:



Позовите экзорциста!

Но это было маленькое отступление. Теперь поговорим о том, что значит для нас решение этой проблемы.

Если на хотя бы одну задачу из NP (вроде 3COLOR) получится найти решение из класса P, то и разницы между P и NP никакой нет, и оба класса станут одним целым.

Как это работает? Для ответа на этот вопрос мне придется рассказать про NP-полные задачи и сведение задач друг к другу. Я думаю, что лучше это будет запихнуть в отдельный тред, иначе этот выйдет совсем убийственным.

Что важно принять - если найти, подчеркиваю, решение из P для хотя бы одной NP-полной задачи (а 3COLOR как раз из таких), то из этого автоматом следует равенство этих классов.

Изменится ли наш мир, если это произойдет? Штош...
Мир изменится. И есть две новости:

Хорошая новость: многие расчеты, требующие сегодня миллионов лет, станут решаемы за пару минут/часов/дней

Плохая новость: многие расчеты, требующие сегодня миллионов лет, станут решаемы за пару минут/часов/дней

Почему это плохая новость?
Ну, самый поверхностный ответ - криптография помашет нам ручкой и умчится в закат.
Никакой защиты транзакций, никаких сертификатов, RSA, блокчейна, защиты банков, все это из непробиваемой брони станет листом бумаги, который не порвет только ленивый.

Потому что взлом этих вещей - NP задачи. И их не взламывают потому, что это нерентабельно. Вряд ли кто-то возьмется тратить триллионы долларов аренду датацентров на взлом моей кредитки, правда?

Вся классическая криптография основывается на том, что P != NP
Ядерная война, крах экономики, банковской системы, возможно, конец нашей информационной эре. А может и нет, но точно будет очень-очень больно.

Если выйдет доказать, что сведение невозможно, и P таки не равно NP, то P останется подмножеством класса NP.
В целом, мир никак не изменится, потому что сегодня все и так из этого исходят.

Наш медленный, скучный мир останется при нас, по крайней мере, до появления распространения новых видов вычислительных моделей, а-ля квантового компьютера с миллионами кубитов по цене сегодняшних пк.

Поговорим о том, куда двигается наука. Наука пытается придумать, как разделить эти классы окончательно, тобишь доказать, что P != NP.
Я видел попытки энтузиастов доказывать P=NP, но они упирались в математическое мошенничество, см Псевдополиномиальные алгоритмы.

Мне, конечно, хотелось бы увидеть дивный новый мир, где P окажется равным NP, но жить я предпочту в мире, где они не равны.

Есть ли какое-то движение по этому вопросу?
Не то, что бы сильно, но время от времени доказывают равенство или неравенство классов сложности более высокого порядка.

Не кажется ли мне, что эта проблема никогда так и не будет решена? Никогда не говори никогда в науке. Для математики и Computer Science это особенно актуально.

В следующих тредах я планирую обсудить интересные проблемы и парадоксы, поговрить про сведение NP-полных задач, а так же думаю начать линейку тредов по основам алгоритмов в качестве бессовестного инфоцыганства!

Надеюсь, что этот рассказ достаточно хорошо и просто осветил проблему P vs NP. Если что-нибудь не ясно, или я где-то что-то перепутал или напорол - отвечу на все возникшие вопросы)

@threadreaderapp unroll

@threadreaderapp unroll

• • •