Вычислительная сложность. Метод ветвей и границ.

Поздняков Сергей Николаевич

запись конспекта: Ковалева Ксения
дата лекции: 13.01.2019

0:00

Введение.

Лекция посвящена вопросу сложности алгоритма.

Мы с вами разбирали ситуации, когда вообще нет алгоритма для решения какой-то задачи. Теперь же стоит задача, когда алгоритмы есть, только они настолько «плохие», что серьезные задачи решить невозможно из-за экспоненциального роста времени.

Возник вопрос: можно ли написать не эффективный алгоритм для решения задач? Пример ответа на такой вопрос: «У вас есть два стозначных числа, вам нужно вычесть из одного другое. Есть ли для вычитания эффективный алгоритм?» Во-первых, вы все делали эти системы компьютерной алгебры, и они работали. Если бы они были не эффективны, то ваши программы бы просто зависли и ничего не складывали, не вычитали, не делили. Алгоритмы, конечно, есть, и вы их реализовывали. И представьте себе, что мы будем по определению (что такое разность: $a-b=c$, если $b+c=a$). Вот у нас есть признак, по которому мы можем проверить правильность ответа.

Допустим, что у нас есть эффективный алгоритм сложения и мы можем на него сослаться. Пишем такой алгоритм: цикл от 1 до 10^100 степени, прибавлять по c от 1 до 10^100, прибавлять c к b. Проверить, равно оно a или не равно. Если равно, значит ответ найден. И получается, что задача стала бессмысленной, потому что перебрать 10^100 чисел, мы с вами год назад считали, что это время жизни галактики.

Между тем вы опирались на хороший алгоритм, просто включили его в переборный, и задача стала бессмысленной. Поэтому, конечно, стоит разделить задачи на два класса: одни — которые по необходимости

переборные, а другие — только по незнанию, то есть что вы не знаете, как сделать эффективный алгоритм и делаете не эффективный.

Мы с вами обсудим так называемую *полиномиальную сводимость*. Возьмем какую-нибудь задачу, для которой никто не знает хорошего, эффективного алгоритма, которую приходится решать перебором, и покажем, что есть другие задачи, которые сводятся к этой и наоборот.

Если бы была решена другая задача эффективно, то и эта тоже была бы решена и наоборот. Это означает, что мы должны найти эффективные алгоритмы, которые одну задачу преобразуют к другой, тогда они будут в этом смысле эквивалентны, что будут решаться либо оба «хорошо», либо оба «плохо».

§3. Вычислительная сложность.

3:57

Сначала мы с вами рассмотрим такой класс, который вызывается *NP-сложные* или *NP-трудные задачи*: это задачи, у которых решение проверяется эффективным алгоритмом.

Пример 1. Проверим выполнимость КНФ. То есть написана какая-то длинная конъюнктивная нормальная форма, и нас просят выяснить, есть ли такой набор значений переменных, чтобы эта функция равнялась единице, или нет.

На данный момент лучшего алгоритма, чем переборный, для решения этой задачи никто не нашел. В то же время, если вам дали набор и спрашивают, выполняется ли в этом наборе функция (то есть единицу она дает или 0). Понятно, что подстановка и вычисление значений делается моментально, то есть алгоритм эффективен. Поэтому вот это можно назвать примером NP-трудной задачи.

А еще среди них мы выделим класс, который будет называться *NP-полные задачи* — это те задачи, которые сводятся друг к другу и обладают тем же свойством NP-трудных.

Задача SAT.

5:56

Рассмотрим сначала задачу, которая называется SAT — это задача выполнимости КНФ. Все такие задачи ставятся в виде предикатов, которые имеют ответ «да» или «нет». Поэтому задача формулируется так: «Существует ли набор значений переменных, на которых заданная КНФ принимает истинное значение?»

Сразу от этой задачи перейдем к так называемой 3SAT задаче. Оказывается, что произвольную КНФ можно свести к такой КНФ, к которой каждый множитель, каждый конъюнкт содержит всего три переменных, и именно это позволяет в дальнейшем удобнее доказывать сводимость одной задачи к другой, в данном случае сводимость этой задачи SAT.

Задача 3SAT.

8:01

Итак, вместо задачи SAT рассмотрим задачу *3SAT*: это то же самое, только каждый конъюнкт или множитель КНФ содержит ровно три, или не более чем три (из двух три всегда можно сделать три, это мы рассмотрим позже) переменных.

Есть такая простая теорема, что выполнимость задачи SAT эквивалентна выполнимости задачи 3SAT: мы можем у любой КНФ построить КНФ с тремя переменными в каждом множителе так, что выполнимость одной будет эквивалентна выполнимости другой.

Покажем это на примере, потому что общая идея достаточно прозрачна.

Пример 2. Возьмем функцию $f(x, y, z, u) = (x \vee \bar{y} \vee \bar{z} \vee \bar{u}) \wedge (\ldots) \wedge \ldots \wedge \ldots$

Рассмотрим, как преобразуется один множитель, остальные преобразуются аналогично. Введем новую переменную: возьмем, например, первые две переменные и обозначим за V.

Тогда вот этот множитель можно заменить, то есть с точки зрения логической функции это эквиваленция, на такой:

$$(x \vee \bar{y} \vee \bar{z} \vee \bar{u}) \equiv (V \vee \bar{z} \vee \bar{u}) \wedge (V \equiv x \vee \bar{y})$$

Вы видите, что в каждой скобке теперь три разных переменных. Единственное: вы можете сказать, что во второй части стоит не дизъюнкция, а эквиваленция, но если мы не будем избавляться, нам же не добавится никаких переменных. Если хотите, можно и избавиться: давайте вспомним про эквиваленцию. Какая у неё КНФ, чтобы заменить эквиваленцию на произведение?

x	y	$x \equiv y$	
0	0	1	
0	1	0 $\wedge$	$x \vee \bar{y}$
1	0	0 $\wedge$	$\bar{x} \vee \bar{y}$
1	1	1	

Так, значит, для КНФ обратим внимание на те строчки, где 0, если 0 - не берем отрицание, если 1 – берем. Тем самым получим вот такую формулу: $x \equiv y = (x \vee \bar{y}) \wedge (\bar{x} \vee \bar{y})$

Поэтому мы можем заметить $(V \equiv x \vee \bar{y})$ на $(V \vee x \vee \bar{y}) \wedge (V \vee (\bar{x} \wedge y))$

Теперь опять получилось не КНФ. Во-первых, мы можем просто раскрыть скобки, пользуясь тем, что у нас симметрия в операции (в свойствах операций дистрибутивность в таком виде тоже существует).

$(V \vee x \vee \bar{y}) \wedge (V \vee (\bar{x} \wedge y)) = (V \vee x \vee \bar{y}) \wedge ((V \vee \bar{x}) \wedge (V \vee y))$

Вы можете сказать, что тут их не три, а две. Но что делать, если вы хотите получить ровно три, покажем на одной скобке:

$V \vee \bar{x} = V \vee \bar{x} \vee (y \wedge \bar{y})$

Чему равна $(y \wedge \bar{y})$ по закону исключенного третьего? Нулю, потому что произведение. Значит добавление нуля ничего не меняет.

Ну и если снова действовать по тому же свойству дистрибутивности, раскроем и получим:

$V \vee \bar{x} \vee (y \wedge \bar{y}) = (V \vee \bar{x} \vee y) \wedge (V \vee \bar{x} \vee \bar{y})$

Итак, мы рассмотрели прием и осталось выяснить, почему выполнимость одной эквивалентна выполнимости другой. Если, допустим, выполняется формула КНФ, то, естественно, можно вычислить все введенные переменные, поставить в расписанный вариант и, естественно, получится тоже единица. Наоборот, если у нас уже 3SAT, мы опять же возвращаемся обратно и из-за того, что у нас V определено именно так, как здесь, то будет тоже давать один.

Теперь более интересно: как показать сводимость других задач к задаче выполнимости, к задаче 3SAT? Мы рассмотрим две задачи на графы, потому что мы с вами проходили отдельно графы и отдельно булевы функции, но оказывается, в рамках этой задачи между ними есть связь.

Сейчас мы научимся строить задачи по выполнимости булевой функции, граф с какими-то свойствами и наоборот.

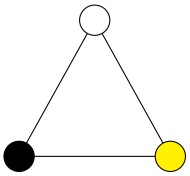
Задача 3-color

16:33

Итак, вторая задача, которую мы рассмотрим это *3-color* – задача раскраски графов в три цвета. У вас есть неориентированный граф, и вам надо раскрасить тремя красками вершины так, чтобы каждое ребро имело концы разного цвета, чтобы если две вершины соединены ребром, то они были раскрашены разными цветами.

Вот это задача *3-color*: раскрасить вершины неориентированного связного графа так, чтобы вершины, соединенные ребром, были раскрашены разными цветами.

Самый простой пример, который в три цвета можно раскрасить, а в два нельзя — это треугольник.



Допустим, одна вершина будет белая, вторая — черная, а третья — какая-нибудь ещё.

Как связать раскраску с булевыми функциями: поскольку красок у нас три, значения 0 и 1 недостаточно. Значит, мы рассмотрим наборы, допустим, 01, 10 и 11 — кодировки цветов.

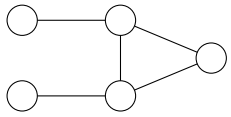
Как теперь эти цвета связать с булевыми значениями? Раз у нас набор из двух цифр, значит, нужно ввести две переменные. Пусть v_i1 — первая компонента, а v_i2 — вторая. И поскольку у нас не четыре цвета, а три, нужно исключить вариант 00. Как мы это сделаем? Напишем такое выражение: $(v_i1 \vee v_i2)$ — оно становится истинным при условии, что по крайней мере одна цифра равна единице, поэтому 00 не будет работать, потому что мы будем рассматривать задачу выполнимости, когда КНФ принимает значение 1.

Теперь как нам поступить, если по условию у нас соседние вершины раскрашены разными цветами? Если есть ребро (i, j), соединяющее вершины i и j, то соответственно надо сказать, что у них по крайней мере один разряд отличается, то есть $(v_i1 \neq v_j1) \vee (v_i2 \neq v_j2)$. Объединяем с $(v_i1 \vee v_i2)$ знаком коньюнкции для всех вершин i и j.

И мы получаем такую большую КНФ. Дальше вы преобразуете по тем правилам и получается, что если мы сумели найти такой набор, который обращает в истинное значение выражение, построенное по графу, значит, мы найдем по нему раскраску, потому что вот у нас эти значения дают нам раскраски вершин. Поэтому в данном случае мы свели задачу раскраски к задаче выполнимости: если задача выполнимости будет решена хорошо, то есть задача раскраски тремя цветами будет решена эффективно. Это принято записывать вот так: $3 - color \geq_p 3SAT$, то есть эта задача полиномиально сводима (или сводима по Карлу) 3SAT. Если есть хороший алгоритм для решения 3SAT, то мы можем решить хорошо и 3-color.

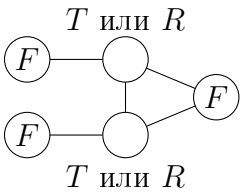
То, что переход туда-сюда — полиномиальный, доказательства не требует, потому что этот алгоритм сформулирован явно. Ясно, что сложность у него линейная и никаких проблем нет. Теперь интересно в обратную сторону: сводится ли наоборот, если мы можем раскрасить граф, следует ли отсюда, что мы сможем решить эту задачу? То есть будут ли они эквивалентны, они попадут в один класс?

Итак, давайте теперь рассмотрим обратную задачу: как получить сводимость задачи 3SAT к задаче 3-color? Здесь такая идея: граф будем строить из маленьких графов.

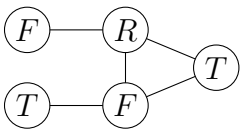


Будем использовать три цвета: F — ложь, T — истина и R (обозначение третьего цвета в работе авторства Мусатова, посвященной данной теме).

Обратите внимание на следующее: если у вас две левые вершины окрашены в цвет F, то какого цвета будет правая вершина? Тоже F, согласны? Тогда граф будет выглядеть следующим образом (причем в центральные вершины нельзя записать одинаковые цвета):



Ну а теперь в остальных случаях: если взять одну левую вершину F, а другую T, или два T (будет та же ситуация, что и в прошлый раз), то всегда можно найти такую раскраску, что в другой вершине будет тоже T. В верхней центральной вершине может быть T или R, в нижней — F или R (причем два R быть не может). Мы строим граф так, чтобы в правой вершине получилось T, то есть мы хотим доказать, что всегда можно покрасить так, что в ней будет T. Тогда в верхней центральной вершине мы T выбирать не будем, а выберем тут R, а в нижней — F.



Вот почему мы будем использовать такую конструкцию, потому что у нас есть возможность так раскрашивать. Теперь давайте возьмем какую-нибудь формулу и на примере этой формулы покажем, как надо по ней построить граф. Мы не будем брать ту формулу, которая была приведена в начале, так как там получилось много множителей. Минимум, который

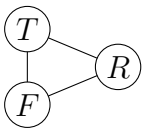
еще можно нарисовать на доске, это три множителя, поэтому мы возьмем четыре переменные и три множителя.

Пример 3. $f(x, y, z, u) = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee u) \wedge (\bar{y} \vee \bar{z} \vee \bar{u})$

Будет ли эта формула выполнима? Допустим, мы взяли $x = 1$, тогда первый множитель выполняется независимо от других. Во втором будет ноль, тогда y тоже нужно взять за единицу, и этот множитель будет выполняться. В третьем множителе $\bar{y}$ будет ноль, значит, нужно взять за единицу $\bar{z}$ или $\bar{u}$. Имеем $x = 1, y = 1, z = 0$, а чему равно u не имеет значения.

Теперь остается изобразить большой граф, который состоит из таких компонент, которые показывают, как строится по формуле граф, который будет в дальнейшем раскрашиваться.

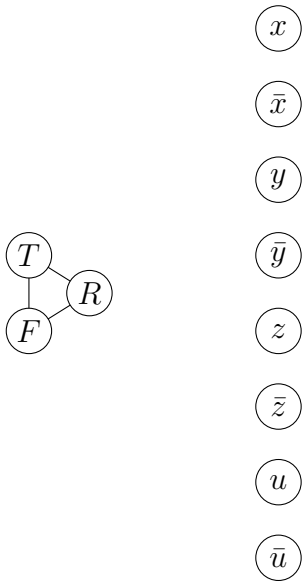
Изобразим *палитру*:



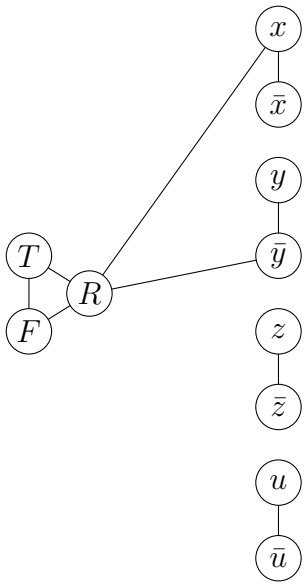
Палитру можно не рисовать, а просто сказать, что крайняя вершина будет окрашена в цвет R.

На самом деле, если вы раскрасили, а у вас цвет крайней вершины получился не R, а T или F. Вы взяли и переименовали краски, то есть сделали так, что вершина окрашена в R, так что эта процедура необязательна.

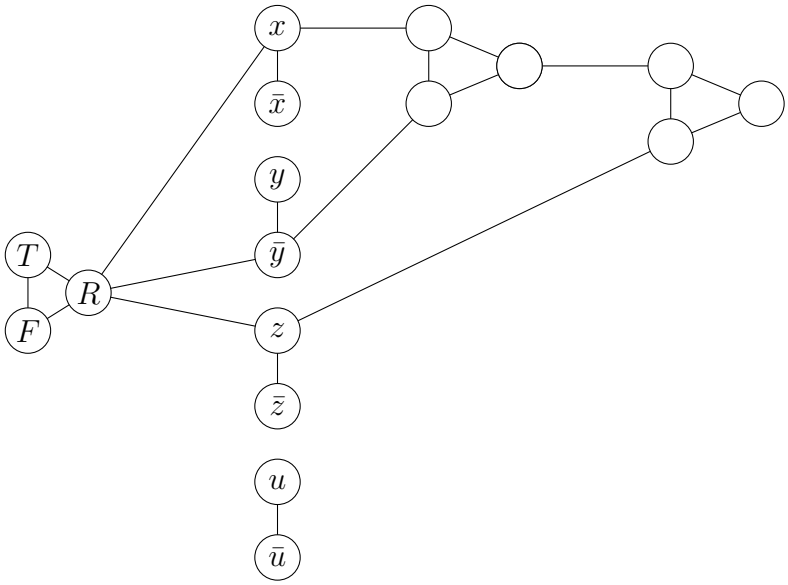
Теперь напишем все переменные, которые входят:



Будем их соединять, во-первых, между собой, потому что раз они — отрицание друг от друга, значит, они должны быть раскрашены в разные цвета. Теперь берем первую формулу: x и отрицание y.



Теперь построим такую конструкцию, добавляем третий z и строим еще одну конструкцию:

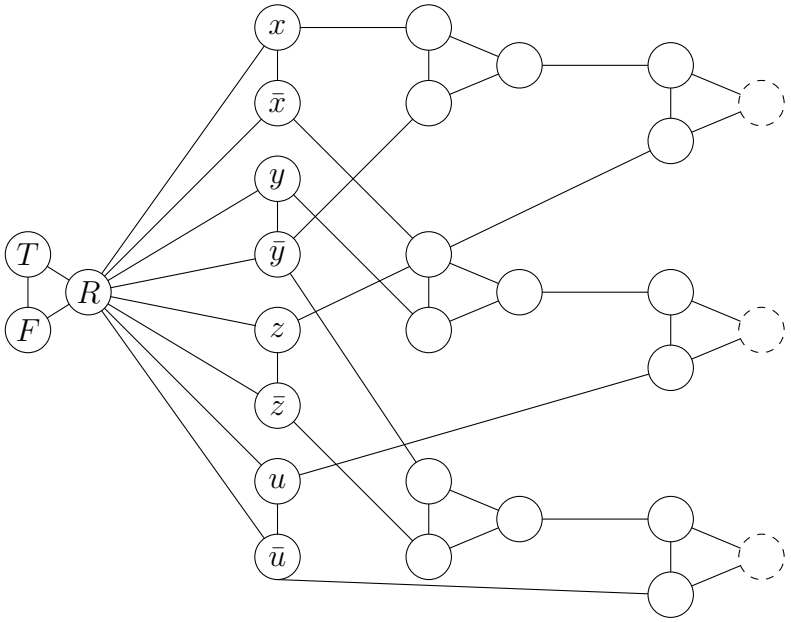


Если какая-то из переменных будет равна единице, то есть если функция выполнима, значит здесь будет где-то единица. Допустим, среди x

и $\bar{y}$, тогда обязательно крайняя вершина первой конструкции будет единицей, то есть иметь цвет T. Если же среди них единице равна только z, то крайняя вершина второй конструкции будет также окрашена в цвет T. Получается, что в любом случае можно раскрасить граф так, что в крайней вершине второй конструкции будет цвет T.

На самом деле, крайняя вершина должна быть одна, но чтобы легче было рисовать, изобразим ее в трех экземплярах.

Переходим ко второму множителю и имеем:



Выделяем крайние вершины пунктиром, потому что, по сути, это одна и та же вершина, и мы объединим их в одну.

Итак, если у нас есть три КНФ, которые выполняются, то мы изобразим граф так, что он будет иметь правильную раскраску.

Теперь в обратную сторону, если нам кто-то даст правильную раскраску, то обязательно пары вершин будут разных цветов, и поскольку в палитре у нас стоит R, цвета мы можем заранее зафиксировать или, как мы уже обговорили, поменять цвета. Поэтому на первом уровне у нас обязательно будут цвета F и T. Ну и раз вершины соединены ребром, не могут стоять два одинаковых цвета одновременно. Получается, то, что мы получим раскраску на этом уровне, и будет набором для выполнимости КНФ.

Таким образом мы показали, что задача 3SAT сводима к задаче 3-color, то есть $3SAT \geq_p 3 - color$.

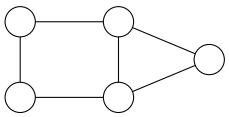
Задача о независимом множестве точек в графе.

39:24

Рассмотрим еще одну красивую задачу: задачу о независимом множестве точек в графе.

Определение. *Независимое множество точек* — это набор точек, между которыми нет ребра. Если рассматривать одну точку, то она всегда будет независимой.

Пример 4 (Задача INDSET)**.** Изобразим граф:

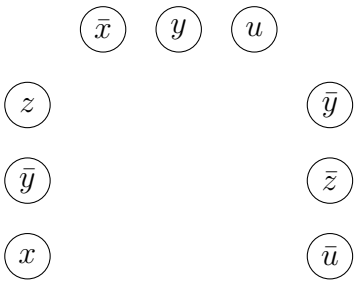


Существуют ли на данном графе два независимых множества, то есть существуют ли две точки, не соединенные ребром? Да, существуют. Существуют три независимых? Как бы мы ни взяли три точки, две из них обязательно будут связаны.

Значит, эта задача заключается в поиске k независимого множества вершин, то есть найти k вершин, из которых никакие две не соединены между собой ребром.

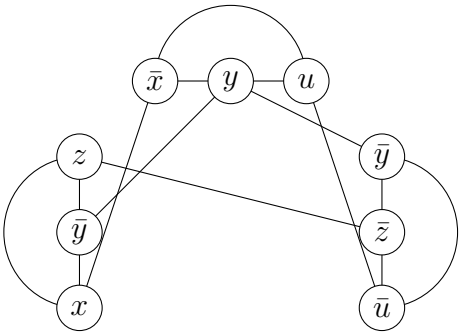
Покажем сводимость задачи 3SAT к задаче INDSET в одну сторону, а обратную сводимость вы можете придумать самостоятельно.

Значит, нужно придумать граф, и если мы в нем найдем независимое множество точек, то задача будет решена. Будем работать с уже известной нам функцией $f(x, y, z, u) = (x \vee \bar{y} \vee z) \wedge (\bar{x} \vee y \vee u) \wedge (\bar{y} \vee \bar{z} \vee \bar{u})$. У нас три множителя, в каждом по три слагаемых, и мы отдельно их изобразим:



Все вершины соединим между собой так, чтобы из них выбиралась только одна вершина. Мы строим независимость множества и не хотим, чтобы из подмножеств их было выбрано две. Поэтому мы их все соединяем, поэтому две выбрать нельзя.

Теперь нужно соединить их так, чтобы одновременно не брали отрицание x и x, отрицание y и y и так далее.



В данном случае нужно построить три независимых множителя, и это не из-за того, что три КНФ: три КНФ, если в каждом множителе три слагаемых, а у нас получилось, что множителей три.

Нужно построить три независимые множества в этом графе, причем мы не можем выбрать две вершины в одной тройке — каждой вот этой части будет выбрана только одна. Если мы сможем найти такие три, значит, мы аналогично можем сказать, что граф построен. Допустим, выберем точки $x, y, \bar{z}$. Видно, что это независимое множество и по этому независимому множеству сразу же строится и набор, на котором наша функция выполняется.

Этим рисунком мы с вами показали, что задача 3SAT сводится полиномиально или сводится к $3SAT \geq_p INDSET$.

§4. Метод ветвей и границ.

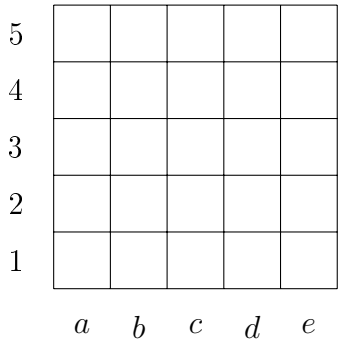
46:06

Мы будем с вами обсуждать теперь, что делать с задачами, у которых нет эффективных алгоритмов, но все равно нам их нужно решать. Есть такая известная задача, задача коммивояжера, которую хотелось бы с вами обсудить, но там картинка гораздо сложнее, поэтому мы рассмотрим задачу попроще. Поэтому если кто-то хочет получить полбалла на экзамене, разберитесь с тем, как задача коммивояжера приводятся к задаче 3SAT и наоборот.

Давайте теперь обсудим такой метод, перед тем как перейти к обсуждению метода ветвей и границ. Мы обсудим сначала более простой прием — перебор с возвратом. Начнем с одной популярной задачи.

48:36

Пример 5 (Задача о расстановке ферзей.)**.** Мы возьмем доску 5 на 5: это минимальная доска, на которой их можно расставить. Ну и, как положено делать в шахматах, мы пронумеруем клетки.



Введем некоторую терминологию: мы будем рассматривать частное решение и расширение этого решения. Частным решением у меня будет то, когда мы поставили часть ферзей. Частичное решение.

Первое, понятно, что мы не будем перебирать, ставить двух ферзей на одну вертикаль и так далее. Поэтому первый ферзь будет перемещаться вверх по первому столбцу, второй — по второму и так далее. Поставим ферзя в клетку $(a;1)$, и это будет частичное решение задачи, значит, что-то поставлено.

Затем расширение. Здесь мы будем пытаться поставить второго ферзя: второго ферзя в клетки $(b;1),(b;2)$ поставить нельзя, поэтому расширением будут остальные три клетки в столбце. Мы выбираем из нее одну и получаем следующее частичное решение, и у этого частичного решения расширение будет в третьем столбце, в котором можно поставить ферзя только в клетку $(c;5)$. Вот мы получили расширение этого решения. Теперь пробуем его дальше расширить: в $(d;1)$ — нельзя, а вот в $(d;2)$ — можно, имеем частичное решение с четырьмя ферзями.

Заметим, что на доске 4 на 4 также можно расставить ферзей, поэтому протокол будем делать на четыре.

Сформулируем алгоритм перебора с возвратом. У него будет частичное решение, которое будет расширяться. Значит инициализация, в начале x будет пустое. 52:08

Algorithm 1 Алгоритм АПВ(x).

Инициализация:
$x = \emptyset$
Функции:
$sol(x)$ — является ли x решением
$ext(x)$ — расширение частного решения x
Алгоритм:
if $sol(x)$ **then**
    $save(x)$ ▷ Сохранить x
else
    for all $y \in ext(x)$ **do**
        АПВ($x \circ y$)
    end for
end if

Функция сохранения написана, чтобы рекурсия какая-нибудь закончилась. Мы идем внутрь, если не попадаем, то переберем следующую, и если этот алгоритм реализовывать итеративно, а не рекурсивно, там появится стек. Если вы просто будете писать протокол работы рекурсивного алгоритма, то это называется *стек вызовов процедур*: вот вызвали, потом еще, еще, потом вернулись и так далее — это все равно так или иначе использует стек.

Давайте теперь напишем протокол на этом примере уже покороче:

Протокол.

x	y	$sol(x)$
$\emptyset$	a_1, a_2, a_3, a_4	F
a_1	b_3, b_4	F
a_1b_3	$\emptyset$	F
a_1b_4	c_2	F

Делаем первый шаг, выбираем a_1, оно становится частичным решением. Оно еще неполное, и расширения у него будут b_3, b_4. Снова выбираем первое — теперь частичным решением будет a_1, b_3, и оно все еще не будет полным. Расширения у него не будет, так что возьмем a_1, b_4 и посмотрим, что будет дальше: расширение c_2 — решения нет, и так далее.

Теперь рассмотрим более сложную ситуацию: что делать, если у нас есть еще одна мера, которая соответствует нашему решению? Если говорить о перспективе, о задаче коммивояжера, то мы, с одной стороны, строим гамильтонов путь, то есть путь, который проходит через все вершины. Если он прошел, то это решение, но это еще не значит, что это самый короткий путь. Поэтому для каждого такого пути мы введем понятие *целевой функции*.

$f(x)$ — целевая (весовая) функция, определена на x, для которых $sol(x) = true$, то есть для которых она является решением.

Мы хотим выбрать лучший путь, поэтому значение, которое является минимальным, мы будем называть r — *рекорд*. И когда мы в очередной раз будем вычислять значение целевой функции, мы будем сравнивать рекорды: если мы получили меньшее значение, то мы будем менять и x, и рекорд.

Ну и, наконец, самое главное — *метод ветвей и границ*. *Граница* означает, что мы умеем оценивать частичное решение, что, еще не дойдя до конца, мы уже можем сказать, перспективно решение или нет.

Как это можно считать? Допустим, мы не можем посчитать $f(x)$ частичного решения, не можем посчитать длину пути до того, как мы этот путь построили, но мы можем его оценить: например, оставшиеся ребра, которые настолько длинные, что если добавить еще одно ребро, уже получится больше, чем рекорд.

Зачем тогда еще дальше идти? А если придумать что-то поумнее, чем просто грубая оценка, то мы остановимся раньше и нам не нужно будет двигаться очень далеко, то есть будет способ *уменьшить перебор*.

Итак, пусть $b(x)$ — оценочная функция, которая определена на *частичных* решениях.

Причем она не просто там определена, а можно так написать, что $b(z) \leq f(x)$, где x — решение, полученное из z последовательными расширениями.

Когда мы пытаемся оценить, насколько хорошо это решение, то как бы рассматриваем все и как они могут там продолжиться, но если есть какой-то теоретический способ оценить то, как их ни продолжай, все равно, если функция будет больше, то уже дальше идти смысла нет, а если меньше, то все нормально. И мы оцениваем это частичное решение: если оно будет больше, чем уже имеющийся рекорд, то дальше идти смысла нет. 1:04:26

А теперь из алгоритма АПВ сделаем алгоритм метода ветвей и границ.

Algorithm 2 Алгоритм МВГ(x).

Инициализация:
$x = \emptyset$
$r = +\infty$
Функции:
$sol(x)$ — является ли x решением
$ext(x)$ — расширение частного решения x
Алгоритм:

```
if sol(x) then
    if f(x) ≤ r then
        r := f(x)
        save(x)
    else
        for all y ∈ ext(x) do
            if b(x ∘ y) < r then
                МВГ(x ∘ y)
            end if
        end for
    end if
end if
```

Так как теоретически нет ощущения, что мы придумали что-то существенное, поэтому рассмотрим пример задачи коммивояжера, на котором и будет показано, как этот алгоритм работает.

1:07:51

Пример 6 (Задача коммивояжера.)**.** Итак, рассмотрим пример одной из версий задачи коммивояжера.

Классическая задача коммивояжера звучит так: надо выйти из конкретной вершины, обойти все вершины графа и вернуться обратно так, чтобы вот этот замкнутый путь имел минимальный вес.

Но можно другую версию придумать: например, не цикл, а путь, значит, мы начинаем с какой-то точки и не возвращаемся обратно. И можно еще одну версию придумать: когда все равно, из какой точки начинать, но надо обойти все вершины так, чтобы сумма была минимальна.

Мы рассмотрим третий вариант, то есть мы будем искать гамильтонов путь, начинающийся из любой точки, проходящий через все вершины (раз он гамильтонов) и имеющий наименьший вес для заданного неориентированного или ориентированного (все равно) взвешенного графа.

Теперь давайте посмотрим, как мы должны определить наши функции: $sol(x)$ означает, что путь проходит через все вершины, то есть он

гамильтонов. Проверить это легко: если вам дали путь, вы просто смотрите, проходит ли он через все вершины. Гамильтонов должен по одному разу проходить, но в данном случае, поскольку мы ищем минимальный путь, мы будем рассматривать граф по крайней мере без циклов отрицательной длины, но обычно его и с неотрицательными ребрами рассматривают, так что проходить через одну и ту же вершину два раза бессмысленно.

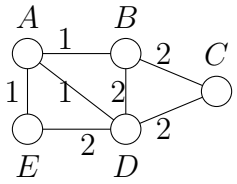
$f(x)$ у нас будет суммой ребер, или длиной гамильтонова пути. Рекорд — самый короткий на данный момент найденный путь.

Начинается самое интересное — это оценочная функция $b(x)$. Значит, если частичным решением считать недостроенный гамильтонов путь, то, например, за оценочную функцию можно взять наименьшее из оставшихся ребер и добавить к тому, что уже найдено, и это будет оценка.

Так вот оказывается, что это не очень эффективный способ, и в качестве частичных решений надо рассматривать не недостроенные гамильтоновы пути, а те ребра, которые не нужно включать в гамильтонов путь. Вот это и есть изюминка такого подхода к решению этой задачи.

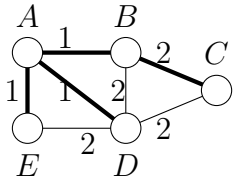
Значит, в качестве частичного решения будем рассматривать не частично построенный гамильтонов путь, а множество ребер, не входящих в искомый минимальный путь.

Покажем идею на примере. Давайте рассмотрим какой-нибудь граф:



Требуется найти путь, который проходит через все вершины.

Вместо того чтобы искать путь, мы можем построить минимальное остовное дерево.



Как вы считаете, верно, что вес этого дерева будет меньше либо равен длине кратчайшего гамильтонова пути? Почему? Потому что путь

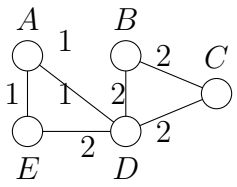
является деревом. Значит, если мы возьмем более широкое множество — все деревья — то пути у него будут только части.

Если же мы ищем минимум в меньшем множестве и минимум в большем, конечно, на более широком множестве минимум окажется меньше, поэтому вот это и есть оценка частичного решения. Вот что здесь самое главное: в качестве оценочной функции мы будем рассматривать остовное дерево тех ребер, которые еще не исключены.

У нас есть частичное решение, и как мы будем строить расширение? Вы согласны, что там, где сходятся три ребра, что-то не хорошо: это не может быть путем, если сошлись три ребра.

Значит, одно из этих ребер надо выбросить. Получается, что если в начале у нас было пустое решение $x = \emptyset$, то есть мы еще ни одного ребра не выбросили, то $ext(x) = AB; AD; AE$

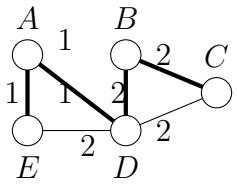
Выбросим ребро AB: у нас получится граф без него.



Теперь решение выглядит следующим образом: x расширили y, и получилось уже не пустое множество, а множество, состоящее из одного ребра: $x \circ y = AB$.

Это ребра, которые не входят в граф, в искомый путь, заведомо.

Строим снова минимальное остовное дерево:



Выходит, что на втором шаге $sol(x) = true$, и мы сохраним то, что раньше $sol(x)$ было равно $false$, а теперь равно $true$. И, значит, мы сохраним это решение и пересчитаем рекорд, потому что рекорд у нас равнялся бесконечности, а теперь будет равен шести. В данном случае это понятно, что ничего больше не будет.