Как из 100 000 000 000 вариантов выбрать нужный,

если перебрать успеваешь только 100 из них?

Владимир Сарванов, ведущий научный сотрудник отдела теории чисел и дискретной математики Института математики, кандидат физико-математических наук

Евгений Макаров, зав. отделом дифференциальных уравнений Института математики, доктор физико-математических наук, профессор;
jcm@im.bas-net.by

Окончание. Начало в №№3, 4

В рамках совместного проекта журнала «Наука и инновации» и Института математики НАН Беларуси представляем заключительную часть публикации об истории математических открытий и их роли в современных технологиях.

Мифический коммивояжер

Многие знают, как найти экстремум (максимум или минимум) функции числового аргумента с помощью производной. Еще недавно этот прием повсеместно изучался в средней школе, и потому многие помнят, что нужно приравнять производную к нулю и решить полученное уравнение. Далеко не каждый из тех, кому это подскажет память, вспомнит и о том, что получившиеся точки – это локальные экстремумы, «местечковые знаменитости», превосходящие лишь ближайших соседей, из-за чего для поиска настоящего максимума или минимума нужно вычислить значения функции в них и сравнить между собой. Если этих точек получилось мало – задача решается быстро. Но их может быть много или очень много. То есть оказалось, что внутри исходной непрерывной задачи оптимизации «пряталась» дискретная, да еще и такая, что решить ее ничуть не проще первоначальной.

Дискретная оптимизация всегда имеет дело с задачами, в которых требуется выбрать из конечного множества допустимых вариантов наилучший – оптимальное решение, то есть разработать эффективный алгоритм, который «быстро» осуществляет требуемый выбор. Если количество допустимых решений невелико, то их можно просто перебрать все по очереди и сравнить между собой за вполне приемлемое время. В общем случае организация полного перебора, то есть перехода от одного варианта к другому без пропусков и повторений, как правило, не сложна. Но такой алгоритм не будет эффективным, поскольку число вариантов, среди которых ищется оптимальный, обычно очень велико. Именно в этом заключается основная трудность задач дискретной оптимизации.

В зависимости от того, каково происхождение обилия сравниваемых вариантов, дискретная оптимизация распадается на:

- *дискретное программирование, где допустимые решения задаются неявно, как решения системы равенств и неравенств (так же, как в задаче сравнения значений оптимизируемой функции в точках локальных экстремумов), в которых область изменения каждой переменной является заданным дискретным множеством чисел, например всех целых неотрицательных;*
- *комбинаторную оптимизацию, в задачах которой многообразие рассматриваемых вариантов определяется большим числом комбинаций заданного типа из относительно небольшого набора исходных данных задачи.*

Типичная задача комбинаторной оптимизации – о построении минимального остовного дерева заданного графа. Допустимыми решениями здесь будут все его остовные деревья, количество которых в графах, богатых ребрами, очень быстро растет при увеличении числа вершин. Еще один такой случай – знаменитая «задача о коммивояжере», которой посвящены несколько монографий и сотни статей в серьезных научных журналах. Классическая версия этой задачи формулируется следующим образом.

Имеется n городов, и заданы стоимости переезда из одного города в другой для каждой их пары. Коммивояжер должен, начиная с произвольно выбранного, объехать все эти города и вернуться в исходный так, чтобы посетить каждый город в точности один раз, и при этом суммарная стоимость всех переездов была бы минимальна.

По-видимому, никто точно не знает, откуда взялась эта задача и кто придумал ее название. Во всяком случае, ни авторы фундаментального сборника [23], специально ей посвященного, ни всезнающая Википедия ничего определенного и несомненного по этому вопросу сказать не могут. Вполне ясно лишь одно: она, как и многие другие популярные проблемы математики XX в., зародилась в густом тумане секретности, окутывавшем военные разработки 40-х гг., и появилась на свет в уже хорошо оформленном виде в отчетах такого монстра холодной войны, как RAND Corporation, незадолго до 1954 г., когда была опубликована статья [24], которая, скорее всего, и являлась самой первой из ряда работ по этой теме.

Задача о коммивояжере обладает удивительной «универсальностью», возникая в очень многих и разноплановых приложениях, внешне ничем не напоминающих ситуацию, в которой оказался мифический коммивояжер. Некоторые из них перечислены в [23]: маршрутизация транспорта, резка рулонного материала, кластеризация массива данных, составление графика работы оборудования. Ей, как и большинству комбинаторных задач, присуще раздражающе-волнующее сочетание простой формулировки и трудного решения.

Из условия задачи понятно, что для ее полного задания необходима таблица стоимостей переезда между городами и схема дорог. Впрочем, без карты можно и обойтись: если некоторые города не соединены дорогами, всегда можно считать, что дорога есть, но билеты стоят дорого – введен запретительный тариф. Если цена достаточно велика, дороже, чем любой путь в объезд, в оптимальный маршрут коммивояжера такой отрезок не войдет. Поэтому всегда можно считать, что условие задачи имеет вид взвешенного полного графа на n вершинах, веса ребер которого определяют стоимость проезда по ним. В этом случае она эквивалентна задаче о поиске в полном графе гамильтонова цикла наименьшего веса.

Поскольку выбор первого города не влияет на искомую стоимость, то вариантами здесь будут все возможные перестановки из n –1 города, задающие все возможные последовательности их посещения. Их число равно $(n-1)!$ или вдвое меньше, если отождествить маршруты, отличающиеся только направлением обхода. И уже, скажем, при n=40 решение задачи путем простого перебора всех вариантов невозможно даже с использованием самого современного компьютера – требуемое для этого время составит миллиарды лет. Между тем на практике приходится решать задачи с числом «городов», измеряемым сотнями и даже тысячами. Например, поиск оптимального маршрута перемещения лазера при

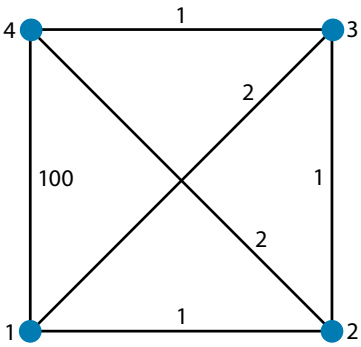

Рис. 1. Пример задачи коммивояжера. На дороге из города 1 в город 4 установлен запретительный тариф

изготовлении современных микросхем приводит к задаче коммивояжера, в которой число городов *n* превосходит 50 000.

Эффективное решение задачи о минимальном остовном дереве было получено с помощью алгоритмов, относящихся к классу жадных. Но в задаче о коммивояжере они не работают. В простейшем примере *(рис. 1)*, выйдя из города 1, коммивояжер, руководствующийся принципом жадного выбора, отправится последовательно в города 2, 3 и 4. Но из города 4 он сможет вернуться в пункт отправления, лишь заплатив запретительный тариф, что делает общую стоимость маршрута заведомо неоптимальной, поскольку, двигаясь в порядке 1–3–4–2–1, можно получить намного меньшую общую стоимость переездов. В этом примере жадный коммивояжер может построить оптимальный маршрут только в том случае, если он отправится в путь из города 3. Но есть примеры задач с несколько большим числом городов, в которых жадный выбор не приводит к цели независимо от места старта.

Для задач, требующих при своем решении обработки особенно больших объемов информации или гигантского количества операций, придуман специальный термин: трансвычислительные задачи. При их решении требуется обработка более чем 10^{93} бит информации. Число 10^{93} называется пределом Бреммермана и равно максимальному количеству двоичных единиц информации, которые мог бы обработать гипотетический компьютер размером с Землю за период времени, равный времени существования нашей планеты.

Трансвычислительной является задача полной обработки данных, поступающих от сетчатки человеческого глаза. Задача о коммивояжере становится трансвычислительной начиная с n=67, если она решается методом простого перебора.

Неэкзотичность трансвычислительных задач означает, что простым наращиванием вычислительных мощностей сколь-нибудь объемные реальные проблемы решить не удастся никогда.

В настоящее время общепринятой формализацией интуитивного представления о быстром (эффективном) алгоритме является понятие полиномиального алгоритма. К ним относят алгоритмы, трудоемкость которых ограничена степенной функцией от размера решаемой задачи. Под трудоемкостью здесь понимается число элементарных операций, выполняемых алгоритмом в процессе решения задачи. Отсюда следует, что время выполнения полиномиального алгоритма на компьютере при решении задачи размера *n* не превосходит величины Cn^s для некоторых констант *C* и *s* и всех значений n. В этом случае говорят, что задача полиномиально разрешима и может быть решена за время $O(n^s)$.

Такая оценка (она называется асимптотической), по сути дела, – лишь оценка по порядку величины. Она имеет смысл только на бесконечных семействах задач, в которые входят однотипные индивидуальные задачи сколь угодно большой сложности, определяемой их размером. Эти семейства называются массовыми задачами. Внутри них отдельные **индивидуальные** задачи различаются значениями параметров, входящих в общее для всех задач семейства условие. Например, в условии массовой задачи поиска гамильтонова цикла в произвольном графе в качестве параметра фигурирует граф *G*. Последнее означает, что в ее постановке скрыто бесконечно много индивидуальных задач, отличающихся друг от друга выбором конкретного графа *G*.

Для каждой индивидуальной задачи трудоемкость, выраженная числом элементарных операций, принимает конкретное числовое значение, а ее асимптотическая оценка возможна лишь при сравнении трудоемкости с размером задачи на всем семействе индивидуальных задач, входящих в массовую. Для задачи о гамильтоновом цикле наиболее естественной характеристикой размера является число ребер в графе *G*. Для задачи о коммивояжере принято считать размером количество городов, которые он должен посетить. В общем случае выбор числового параметра, характеризующего размер, несколько сложнее. Он осуществляется путем, по возможности, экономного кодирования условий задачи, одинакового для всех индивидуальных задач с последующим отождествлением длины кода и размера каждой из них.

Указанная формализация, как и всякая другая, не лишена недостатков. Например, полиномиальный алгоритм, имеющий трудоемкость Cn^s, где, скажем, s =100 и (или) C – огромное число, вряд ли можно назвать эффективным. Имеются и другие, не столь очевидные, патологии этой формализации. Тем не менее сам факт полиномиальной разрешимости задачи имеет принципиальное значение. После того как он установлен, дальнейшие усилия направляются на понижение степени полинома (то есть на уменьшение величины s). В большинстве случаев удается прийти к алгоритмам с оценками трудоемкости не выше чем $O(n^3)$. Какую их трудоемкость можно считать приемлемой для практических задач? На этот вопрос невозможно дать точный ответ вне контекста конкретной области приложений. С большой долей уверенности можно утверждать, что область применимости алгоритмов трудоемкости $O(n^4)$ и выше весьма ограничена, а $O(n^2)$ и ниже подходит для большинства ситуаций. Вместе с тем имеется достаточно представительный круг прикладных задач, для которых алгоритмы с оценкой трудоемкости $O(n^2)$ являются недостаточно быстрыми. Это, например, задачи вычислительной геометрии, возникающие при автоматизации проектирования микросхем.

Гонка за производительностью, сокращением трудоемкости алгоритмов и их потребности в ресурсах как раз и составляет сердцевину всего того, что происходит в алгоритмической теории графов и других алгоритмических разделах дискретной математики. Например, трудоемкость классического алгоритма Прима для построения минимального остовного дерева при его простейшей реализации составляет $O(n^2)$. Совершенствование способов представления данных, необходимых для работы алгоритма, позволяет существенно уменьшить эту величину. Показатели алгоритмов Борувки и Краскала исходно несколько лучше, но и они далеки от рекордных показателей лучшего из современных алгоритмов, оцениваемых величиной $O(ma(m))$, где $a(m)$ – некоторый показатель, который с увеличением m возрастает гораздо медленнее не только любой степени m, но и вообще с трудом отличим от константы, хотя таковой не является. Вопрос о существовании алгоритма построения минимального остовного дерева с трудоемкостью $O(m)$ продолжает оставаться важной открытой проблемой, до сих пор привлекающей внимание исследователей.

В задаче о коммивояжере полный перебор $(n-1)!$ перестановок очевидно не полиномиален. Более того, не известно никакого полиномиального алгоритма решения, и имеются веские, хоть и косвенные доводы, что построить такой алгоритм невозможно.

Впечатляющее сочетание простоты и сложности, свойственное задаче о коммивояжере, привело к тому, что начиная с 50-х гг. XX в. она стала полигоном для оттачивания всех новых методов в трех направлениях:

- *поиск точных алгоритмов, решающих задачу в общем случае, но с меньшими затратами, чем простой перебор;*
- *построение приближенных алгоритмов, дающих не оптимальное, но достаточно хорошее решение;*
- *поиск частных случаев задачи, допускающих эффективное решение.*

Принципиальные обстоятельства существенно ограничивают успехи первого направления. Тем не менее рекордные достижения здесь постепенно растут: решены задачи для 49 городов в 1950-х, для 2392 – в 1980-х гг. и для 85 900 – в 2006 г.

Второе направление за годы своего существования породило большое число эвристических алгоритмов, значительная часть которых оформлена в виде коммерческого софта и потому не полностью раскрывается своими авторами. Создание эффективных коммерческих солверов, решающих популярные задачи дискретной математики с приемлемой затратой вычислительных ресурсов, – процветающий бизнес.

Каждый эвристический алгоритм явно или неявно вводит в задачу определенные дополнительные условия. Они, как правило, основаны на неких правдоподобных (эвристических!) предположениях о том, как должно выглядеть искомое решение. Совместимость этих условий с условиями исходной задачи не всегда доказывается, а иногда может и отсутствовать. Зато они уменьшают множество допустимых решений, что, собственно, и приводит к ускорению работы алгоритма. Поэтому эвристический алгоритм дает не оптимальное решение, а некоторое другое, обычно достаточно хорошее, то есть обеспечивающее не слишком большую суммарную стоимость маршрута. Иногда же работа такого алгоритма может закончиться и неудачей: преждевременным завершением, зацикливанием или маршрутом с заведомо чрезмерной стоимостью в каких-то случаях, представляющихся пользователю несущественными или редко встречающимися. Все эти недостатки с лихвой возмещаются малой трудоемкостью и, соответственно, быстрой работой, а иногда и таким бонусом, как фиксированный размер ошибки в суммарной стоимости маршрута. Простейшим примером

эвристического алгоритма является рассмотренный выше жадный алгоритм, который принято называть алгоритмом ближайшего соседа, а также его усовершенствование, заключающееся в запуске этого алгоритма из каждой вершины графа задачи с последующим выбором маршрута минимальной стоимости. Нетрудно видеть, что на нашем примере это усовершенствование дает правильный результат.

Важный частный случай, в котором поиск приближенных решений проходит особенно успешно, – это когда стоимости переезда связаны с расстоянием между городами, то есть когда любая дорога в объезд дороже прямого пути. Одним из достижений здесь стал открытый в 1976 г. полиномиальный алгоритм Кристофидеса – Сердюкова, гарантированно обеспечивающий решение, стоимость которого не более чем в полтора раза выше, чем стоимость оптимального маршрута.

Алгоритм Кристофидеса – Сердюкова удивляет своей простотой и кажущейся немотивированностью:

1. Строим минимальное остовное дерево T в графе G.

2. Находим совершенное паросочетание M минимального веса в подграфе, порожденном множеством вершин с нечетными степенями в дереве T.

3. Объединяем множества ребер M и T, получая в результате связный мультиграф H, в котором каждая вершина имеет четную степень.

4. Строим эйлеров цикл в H.

5. Преобразуем этот цикл в гамильтонов путем устранения повторений вершин.

Не так давно, в 2011 г., результат Кристофидеса – Сердюкова был улучшен: константа оценки ошибки 3/2 была уменьшена на четыре сотых триллионной триллионной триллионной доли процента. Сейчас попытки улучшить это достижение стали достаточно популярной темой публикаций.

Третье направление также было очень популярным и плодотворным, в том числе и в Институте математики: в [23] целый раздел посвящен результатам, полученным у нас.

Из приведенной выше формулировки задачи о коммивояжере ясно, что ее условие можно задавать не только в виде взвешенного графа, но и с помощью матрицы, элементы которой a_{ij} определяют стоимость переезда из города i в город j.

Для этого, разумеется, города нужно предварительно пронумеровать. Существуют классы матриц, для которых решение можно найти за полиномиальное время. Первый из таких классов был найден еще в 1970 г. академиком Д.А. Супруненко. Он состоит из симметрических матриц, элементы которых возрастают с удалением от главной диагонали. В этом случае задача сводится к перебору маршрутов специального вида, так называемых пирамидальных циклов, при движении по которым номера городов сначала монотонно возрастают, а затем убывают. Существенно, что такой перебор можно выполнить полиномиальным алгоритмом, где степень полинома равна двум, то есть достаточно быстро. Этот результат имел принципиальный характер. Развитая при его получении техника позволила получить результаты подобного рода в других специальных случаях задачи о коммивояжере. В частности, был описан класс матриц, включающий пространство верхнетреугольных матриц, для которых данная задача полиномиально разрешима.

Корень зла

Каждый, вероятно, может вспомнить, как испытывал огорчение из-за того, что решение школьной задачи, над которой безуспешно просидел битый час, на самом деле оказывалось убийственно простым. А ведь чем проще решение, тем труднее его найти!

Вот именно из этой коллизии и выросла самая популярная классификация алгоритмических задач. Она включает в себя знаменитые классы сложности *P* и *NP*, задача о совпадении или несовпадении которых включена в выставленный в 2000 г. Математическим институтом Клэя список из 7 математических «проблем тысячелетия». За решение каждой из них объявлена награда в 1 млн долл. Из этого списка пока решена (Григорием Перельманом) только проблема Пуанкаре.

Теория, лежащая в основе этой классификации, использует асимптотические оценки трудоемкости алгоритмов и поэтому оперирует лишь массовыми задачами. Кроме того, она ограничивается лишь распознавательными задачами, имеющими предельно простую форму ответа – «да» или «нет». Формулировка такой задачи включает 2 позиции: условие (описание объектов и параметров задачи) и вопрос о наличии у них некоторого свойства. Приведем примеры трех распознавательных задач.

ГАМИЛЬТОНОВ ЦИКЛ: дан граф *G*. Содержит ли граф *G* гамильтонов цикл?

ФАКТОРИЗАЦИЯ: даны числа *n* и *k*. Имеется ли у числа *n* делитель *p*, такой, что $2 \leqslant p \leqslant k$?

МИНИМАЛЬНАЯ ЦЕПЬ: дан граф G с двумя выделенными вершинами x, y и число k. Содержит ли граф цепь длины $\leq k$, соединяющую эти вершины?

Между тем подавляющее большинство практических задач являются оптимизационными, в которых нужно найти минимум или максимум некоторой функции $f(x)$ на заданном множестве X. Например, в задаче коммивояжера X – множество всевозможных перестановок, задающих порядок посещения городов, а $f(x)$ – суммарная стоимость всех переездов. Но каждой оптимизационной задаче можно сопоставить ее распознавательную постановку: по заданным множеству X, функции f и числу k требуется определить, существует ли такой элемент x из X, что $f(x)\leq k$. Это дает возможность не исключать такие задачи из рассмотрения, поскольку можно показать, что при естественных предположениях относительно функции f переход к распознавательной постановке не ведет к принципиальному упрощению задачи, то есть труднорешаемой оптимизационной задаче соответствует труднорешаемая распознавательная.

Алгоритм решает массовую задачу, если он в состоянии решить каждую из соответствующих ей индивидуальных задач, а его трудоемкость выражается как функция $s(x)$, отражающая зависимость времени решения индивидуальной задачи от ее размера x. Теперь определим два важнейших класса массовых распознавательных задач – P и NP.

Класс P состоит из задач, для решения которых существуют полиномиальные алгоритмы, то есть тех, которые могут быть решены «быстро».

С определением класса NP дело обстоит несколько сложнее. Говоря неформально, массовая задача принадлежит этому классу, если существует «быстрый» алгоритм проверки того, что ее предполагаемое решение действительно является решением.

В определении принадлежности массовой задачи R классу NP фигурируют следующие три объекта.

I. Сертификат s, связанный с каждой индивидуальной задачей, имеющей ответ «да». Это строчка цифр, представляющая собой запись некоторой дополнительной информации об этой задаче.

II. Алгоритм A проверки сертификатов, применяемый к входу индивидуальной задачи и ее сертификату и являющийся единым для всех индивидуальных задач, составляющих данную массовую.

III. Полином $p(x)$, который характеризует трудоемкость алгоритма A. Он также является единым для всех индивидуальных задач, составляющих данную массовую.

Задача R входит в класс NP, только если существуют алгоритм A и полином $p(x)$, удовлетворяющие следующим условиям: для каждой ее индивидуальной задачи I с ответом «да» найдется такой сертификат $s=s(I)$, что алгоритм A, получив на входе s и I, придет к ответу «да» за время не более чем $p(x)$, где x – суммарная длина записи сертификата s и входа задачи I. Если же вместо задачи I взять любую индивидуальную задачу с ответом «нет», то алгоритм A либо остановится с ответом «нет», либо не остановится вообще.

Например, в массовой задаче ГАМИЛЬТОНОВ ЦИКЛ в качестве сертификата для каждого конкретного n-вершинного гамильтонова графа G можно взять тот самый гамильтонов цикл C, который этот граф содержит. Тогда работа алгоритма будет состоять попросту в выполнении n проверок принадлежности каждого ребра цикла C графу G. Все эти проверки можно выполнить за время, не превосходящее $O(n^2)$, то есть время проверки сертификата ограничено полиномом p второй степени. Следовательно, данная задача относится к классу NP. Доказательство принадлежности этому классу двух других, сформулированных выше, распознавательных задач осуществляется столь же просто.

А если уж совсем «на пальцах», то тогда так. Пусть стоит задача набрать в лесу как можно больше хороших грибов. Это массовая оптимизационная задача. Набрать побольше настоящих грибов в том лесу, который у вас под боком, – индивидуальная оптимизационная задача. Ей соответствует задача распознавания: а есть ли в этом лесу хотя бы один не червивый боровик? Или два? Если теперь предположить, что задача поиска белого гриба под указанной елкой решается просто (при условии, что он там есть), то указание: «А иди-ка посмотри вон под той елкой» и будет сертификатом в этой задаче.

Словосочетание типа «задача B сводится к задаче A» часто встречается во всех разделах математики. Оно означает, что решение задачи B можно получить, решив задачу A, и поэтому последняя не

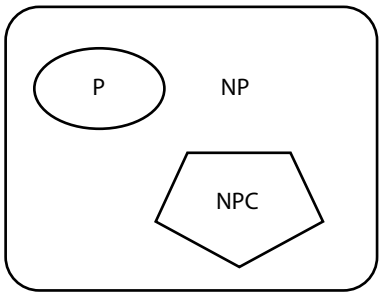

Рис. 2. Соотношение классов сложности P, NP и NPC

проще, чем первая. Нам понадобится сводимость специального вида, которая учитывает трудоемкость получения решения одной массовой задачи из решения другой. Будем говорить, что массовая задача R_2 полиномиально сводится к массовой задаче R_1, если из существования полиномиального алгоритма A для ее решения следует существование полиномиального алгоритма для решения задачи R_2, который имеет возможность использовать алгоритм A как подпрограмму.

Задача называется *NP*-полной, если она принадлежит классу *NP* и любая задача из этого класса полиномиально сводится к ней. Таким образом, *NP*-полные задачи – самые трудные в *NP* и составляют класс, обозначаемый как *NPC (рис. 2)*.

Из трех сформулированных выше распознавательных задач класса *NP* первая является *NP*-полной, третья принадлежит классу *P*, а статус второй неизвестен. Многочисленные примеры других задач из всех этих классов приведены в [25].

Как уже отмечалось, подавляющее большинство практических задач являются оптимизационными и, следовательно, не принадлежат классу *NP*. В то же время ко многим из них удается полиномиально свести некоторые *NP*-полные задачи. В этой ситуации полезным оказывается следующее определение. Задача называется *NP*-трудной, если к ней полиномиально сводится некоторая *NP*-полная задача.

На практике это означает, что разработчику алгоритмов следует отказаться от попыток найти наилучшее решение и довольствоваться «достаточно хорошим» (гарантированно или с большой вероятностью).

Мнение о том, что для *NP*-полных задач не существует полиномиальных алгоритмов, общепринято, однако сам этот факт до сих пор не доказан: именно в этом и заключается суть проблемы равенства классов *P* и *NP*. Для того чтобы доказать равенство *P=NP*, достаточно предъявить полиномиальный алгоритм для решения любой из *NP*-полных задач. Последствия этого будут неисчислимы. Подробная пошаговая инструкция, как подчинить себе весь мир с помощью этого доказательства, приведена в [26].

Но есть вариант и покруче. Кроме классификации задач по сложности, основанной на трудоемкости алгоритмов, известно много других способов учета лимитированности вычислительных ресурсов. Один из них основан на оценке размера памяти, необходимого для решения задачи. Класс задач, для которых эта оценка полиномиальна относительно их размера, обозначается *P-SPACE*. В него, кроме множества других, входят и все задачи из класса *NP*.

Игра «Ним», описанная в начале статьи, может быть переформулирована в виде игры с фишкой на графе, в которой два игрока по очереди двигают фишку по ориентированному графу в соответствии с ориентацией ребер – в направлении, заданном стрелочками на них. Для игры «Ним» этот граф получается в виде дерева, характер ветвления которого определяется ее условиями. В дереве нет циклов. Если их разрешить, а в качестве игрового поля взять произвольный ориентированный граф, получится игра, которую обычно называют «Обобщенная география». Задача поиска выигрышной стратегии в ней является полной в классе *P-SPACE*. Найдите способ всегда побеждать в этой игре – и весь мир падет к вашим ногам!

И все же!..

Как все-таки разобраться с целой горой допустимых, но не лучших вариантов, если на это абсолютно нет ни времени, ни ресурсов?

Ясно, что по причинам, которые обойти не удастся, это возможно не всегда. Ясно, что старые теоремы годятся для прежних задач, а для новых нужны новые. А еще – азарт, эрудиция и изобретательность, которых было в избытке у отцов-основателей алгоритмической комбинаторики. Само собой, сверх того потребуется хоть чуточку везения и обязательно – очень много упорства.

– Но все же, каков ответ?

Да, в общем, ответ прост:

– Уметь надо… Ну, и работать. Много.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

23. D.B. Shmoys, J.K. Lenstra, A.H.G. Rinnooy Kan, E.L. Lawler. The Traveling Salesman Problem (A guided tour of combinatorial optimisation) – Chichester, 1985.
24. Dantzig G.B., Fulkerson R., Johnson S.M. Solution of a large-scale traveling salesman problem // Operations Research. 1954. Vol. 2 (4). P. 393–410.
25. Фортноу Л. Золотой билет. P, NP и границы возможного. – М., 2016.
26. Как захватить мир, доказав, что P=NP // https://habr.com/ru/post/227687/.