Дискретная оптимизация. Весенний семестр 2013.

Лектор: А. Б. Дайняк

Подход к перебору дискретных объектов R. C. Read

Пусть задан некоторый параметр «сложности» объектов. Например, сложностью графа на фиксированном числе вершин можно считать число рёбер в графе. Множество всех объектов разбивается на классы, внутри каждого из которых находятся объекты одной сложности. Так, например, $\mathcal{G} = \mathcal{G}_0 \sqcup \mathcal{G}_1 \sqcup \mathcal{G}_2 \sqcup \ldots$, где $\mathcal{G}_k$ — класс всевозможных графов с k рёбрами.

Основная проблема при переборе обычно состоит в отсеве дубликатов. Например, в случае перебора графов изоморфные графы нам рассматривать часто не имеет смысла. То есть перебрать мы хотим не множество $\mathcal{G}_k$, а некоторое подмножество $\mathcal{G}_k^c$, такое, что каждый граф из $\mathcal{G}_k$ изоморфен одному и только одному графу из $\mathcal{G}_k^c$. Верхним индексом c мы обозначаем тот факт, что множество $\mathcal{G}_k^c$ содержит лишь «канонических» (англ. canonical) представителей графов из $\mathcal{G}_k$.

Определить «каноничность» для перебора простых неориентированных графов несложно. Каждый граф можно представить матрицей смежности, и выписать матрицу по строкам, пропуская все элементы под главной диагональю и на ней. Таким образом, каждому неориентированному графу без петель на фиксированном множестве из n вершин однозначно отвечает двоичный вектор длины $\frac{n^2-n}{2}$. Изоморфным графам отвечают такие матрицы смежности, одна из которых может быть получена из другой синхронной перестановкой строк и столбцов. При этом двоичные вектора этих графов получаются друг из друга (весьма специфической) перестановкой координат. Тем самым, можно определить группу перестановок Γ на множестве координат векторов-кодов n-вершинных графов, так, что графы изоморфны тогда и только тогда, когда вектор-код одного графа можно получить из вектор-кода другого графа с помощью какой-нибудь перестановки из Γ. Группу перестановок Γ мы будем считать «зашитой» в алгоритм перебора, и не будем сосредотачивать внимание на её структуре.

Пример. Посмотрим, что будет представлять из себя Γ при переборе неориентированных четырёхвершинных графов. Матрица смежности любого такого графа имеет вид

$$\begin{pmatrix} 0 & a & b & c \\ a & 0 & d & e \\ b & d & 0 & f \\ c & e & f & 0 \end{pmatrix}.$$

Если синхронно переставить, скажем, первые две строки и первых два столбца, получим матрицу изоморфного графа:

$$\begin{pmatrix} 0 & a & d & e \\ a & 0 & b & c \\ d & b & 0 & f \\ e & c & f & 0 \end{pmatrix}.$$

Если представить матрицы векторами, получим вектора $(abcdef)$ и $(adebcf)$. Видно, что второй вектор получен из первого перестановкой координат. Отметим, что количество перестановок четырёх вершин графа, а значит, и размер группы Γ, равен $4!$. Количество же всевозможных перестановок шести координат вектор-кодов равно $6!$. То есть Γ заведомо не содержит все возможные перестановки координат. □

Каноническим мы будем считать такой граф G, код которого является лексикографически наибольшим среди всех кодов графов, изоморфных G. Проверить любой заданный код на каноничность можно, перебрав всевозможные перестановки из группы Γ, и убедившись, что вектора, полученные из данного под действием перестановок, лексикографически меньше данного.

Пример. Посмотрим, каково каноническое представление цепи на трёх вершинах. Есть три варианта матриц смежности для такого графа:

$$\begin{pmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Выписав построчно матрицы смежности без поддиагональных элементов, получим три вектора: (101), (110) и (011). Из этих векторов нас интересует лексикографически наибольший: (110). Его мы и выбираем в качестве канонического кода для трёхвершинной цепи. □

Далее мы опишем подход к перебору дискретных объектов, предложенный R. C. Read. Пусть объекты разбиты на «слои» согласно некоторой мере сложности, и $\mathcal{O}_k^c$ — множество канонических объектов (к. о.) сложноси k, а $\mathcal{O}_k$ — множество всех объектов сложности k.

Рассмотрим вначале тривиальный алгоритм перебора канонических объектов:
пока не все объекты из $\mathcal{O}_k$ перебраны
берём очередной объект (код объекта) из $\mathcal{O}_k$
если он не канонический, переходим на следующую итерацию цикла
сравниваем его с каждым из уже построенных объектов из $\mathcal{O}_k^c$
если объект не совпадает ни с одним из построенных, добавляем его в список построенных

Проблемы тривиального алгоритма: необходимость сравнивать каждый канонический объект со всеми уже построенными к. о., а также необходимость рассматривать все объекты из $\mathcal{O}_k$. Это приводит к тому, что сложность алгоритма пропорциональна $(|\mathcal{O}_k^c|^2 + T_k \cdot |\mathcal{O}_k|)$, где T_k — время проверки одного кода объекта на каноничность.

Описанные проблемы призван решить рассматриваемый далее алгоритм *упорядоченного перечисления* (orderly algorithm). В его основе следующие идеи:

- Объекты строятся по возрастанию сложности. Не начинаем строить список к. о. сложности $(k+1)$, пока не построен список всех к. о. сложности k.
- Каждый «простой» объект порождает не слишком большое число «сложных».
- Простые объекты перебираются, а сложные порождаются, в строго определённом порядке (например, в порядке возрастания или убывания их кодов).
- Когда порождён очередной сложный объект, он проверяется на каноничность. Если он каноничен и его код следует (*непосредственное* следование необязательно) за кодом последнего из построенных ранее объектов, то этот новый объект можно смело добавлять в список построенных, не сравнивая со всеми предыдущими построенными объектами.

Естественно, что алгоритм упорядоченного перечисления будет работать только при выполнении определённых условий, а выбор эффективного кодирования и упорядочения объектов в каждом конкретном случае может быть нетривиальной задачей.

Также заметим, что алгоритм требует фактически наличия списка к. о. сложности k перед началом построения объектов сложности $(k+1)$. Однако это не является большой проблемой в задачах перебора, поскольку часто мощность $|\mathcal{O}_k^c|$ существенно меньше мощности $|\mathcal{O}_{k+1}^c|$ и на порядок времени работы алгоритма не влияет.

Опять рассмотрим пример с графами. Очевидно, каждый n-вершинный граф с $(k+1)$ рёбрами можно получить несложной операцией, — добавлением ребра, — из графа с k рёбрами. При этом из каждого k-рёберного графа мы получаем не более $(\binom{n}{2} - k)$ графов, которые придётся проверять на каноничность и новизну.

Перейдём от терминологии задачи перебора графов к перебору векторов. Будем рассматривать векторы из $\{0, 1, \ldots, q-1\}^n$. Векторы, у которых сумма всех координат равна фиксированному числу k, отнесём ко множеству $\mathcal{C}_k$. Будем также считать заданной некоторую группу Γ — группу перестановок на множестве $\{1, \ldots, n\}$. Векторы из $\mathcal{C}_k$ будем считать эквивалентными, если существует перестановка из Γ, переводящая один из векторов в другой. Каноническим будем считать вектор, являющийся лексикографически наибольшим в своём классе эквивалентности. Предположим, что уже построен список $\mathcal{C}_k^c$ всех канонических векторов из $\mathcal{C}_k$, упорядоченный в порядке лексикографического убывания, и требуется построить аналогичный список $\mathcal{C}_{k+1}^c$. Видно, что задача включает как частный случай задачу перебора неизоморфных графов на фиксированном множестве вершин с фиксированным числом рёбер, имеющих не более $(q-1)$ кратных рёбер между каждой парой вершин.

Введём операцию порождения, которая по вектору $\mathbf{a} = (a_1, a_2, \ldots, a_m, \underbrace{0, \ldots, 0}_{n-m})$, где $a_m > 0$, строит последовательность векторов

$$\begin{array}{l}(a_1,\, a_2,\, \ldots,\, a_{m-1},\, a_m + 1,\, 0, 0, \ldots, 0),\\ (a_1,\, a_2,\, \ldots,\, a_{m-1},\, a_m,\quad 1, 0, \ldots, 0),\\ (a_1,\, a_2,\, \ldots,\, a_{m-1},\, a_m,\quad 0, 1, \ldots, 0),\\ \qquad\qquad\vdots\\ (a_1,\, a_2,\, \ldots,\, a_{m-1},\, a_m,\quad 0, 0, \ldots, 1).\end{array}$$

То есть первый вектор получается из $\mathbf{a}$ увеличением на единицу крайней правой ненулевой координаты. Остальные $(n - m)$ векторов получаются из $\mathbf{a}$ последовательными заменами нулей на единицы слева направа. Если $a_m = q - 1$, то первый из перечисленных векторов отсутствует. Если $m = n$, то остальные вектора отсутствуют. Если $a_n = q - 1$, то порождаемая последовательность векторов пуста.

Будем писать $\mathbf{a} \preceq \mathbf{b}$, если $\mathbf{a}$ лексикографически не больше $\mathbf{b}$. Чтобы доказать корректность алгоритма упорядоченного перечисления, достаточно проверить выполнение трёх условий:

1. Каждый канонический вектор в $\mathcal{C}^c_{k+1}$ может быть получен с помощью операции порождения из некоторого канонического вектора в $\mathcal{C}^c_k$.

2. Пусть $\mathbf{a}, \mathbf{b} \in \mathcal{C}^c_{k+1}$ таковы, что $\mathbf{a} \succ \mathbf{b}$. Пусть $\mathbf{a}'$ и $\mathbf{b}'$ — первые по порядку вектора в списке $\mathcal{C}^c_k$, из которых могут быть порождены вектора $\mathbf{a}'$ и $\mathbf{b}'$. Тогда с необходимостью должно быть выполнено $\mathbf{a}' \succeq \mathbf{b}'$.

3. Для каждого вектора из $\mathcal{C}^c_k$ порождаемые им вектора в $\mathcal{C}_{k+1}$ порождаются в порядке убывания.

Первое из перечисленных условий говорит о том, что каждый объект в $\mathcal{C}^c_{k+1}$ будет хотя бы раз порождён в ходе алгоритма.

Второе условие гарантирует, что если на каком-то шаге алгоритма из некоторого вектора $\mathbf{a}' \in \mathcal{C}^c_k$ будет порождён и добавлен в результирующий список вектор $\mathbf{a} \in \mathcal{C}^c_{k+1}$, то он не заблокирует собой добавление в список никакого другого объекта $\mathbf{b} \in \mathcal{C}^c_{k+1}$ (напомним, что результирующий список мы строим строго в порядке убывания векторов).

Третье условие, в отличие от перых двух, не необходимо, но является удобным достаточным для корректности алгоритма. Оно требует, чтобы вектора, порождаемые одним и тем же вектором $\mathbf{a} \in \mathcal{C}^c_k$, не блокировали друг друга. Его можно несколько ослабить, потребовав, чтобы по порядку порождались лишь «новые» вектора, — те, которые ещё не порождались ранее векторами вида $\mathbf{b} \succ \mathbf{a}$, $\mathbf{b} \in \mathcal{C}^c_k$.

Тройка указанных условий гарантирует корректность алгоритма в общем виде: достаточно слово «вектор» заменить словом «объект», а вместо лексикографического порядка рассматривать абстрактный линейный порядок.

Далее проверим, что условия выполнены в нашем конкретном случае. Выполение третьего условия вытекает сразу из определения операции порождения; докажем первые два.

Теорема 1. *Пусть $\mathbf{a} \in \mathcal{C}^c_{k+1}$. Пусть $\mathbf{a}' \in \mathcal{C}_k$ — вектор, полученный из $\mathbf{a}$ уменьшением на единицу крайней правой ненулевой координаты. Тогда $\mathbf{a}' \in \mathcal{C}^c_k$.*

Доказательство. Пусть $\mathcal{C}_k(\mathbf{a})$ — всевозможные вектора из $\mathcal{C}_k$ получаемые из $\mathbf{a}$ уменьшением какой-либо из координат на единицу. Пусть $\mathcal{C}^c_k(\mathbf{a})$ — всевозможные канонические вектора, эквивалентные векторам из $\mathcal{C}_k(\mathbf{a})$. Выберем в $\mathcal{C}^c_k(\mathbf{a})$ лексикографически максимальный вектор $\mathbf{a}'$ и докажем, что он совпадает с вектором, получаемым из $\mathbf{a}$ уменьшением крайней правой ненулевой координаты.

В любом случае, $\mathbf{a}'$ можно получить уменьшением одной из координат некоторого вектора $\widetilde{\mathbf{a}} \in \mathcal{C}_{k+1}$, эквивалентного $\mathbf{a}$. Имеем $\mathbf{a} \succeq \widetilde{\mathbf{a}} \succ \mathbf{a}'$, откуда следует, что самая левая несовпадающая координата у вектора $\mathbf{a}$ больше, чем у $\mathbf{a}'$. Запишем вектора $\mathbf{a}'$ и $\mathbf{a}$ друг под другом, обозначив через i и j номера самой левой и самой правой координат, которые у $\mathbf{a}$ больше, чем у $\mathbf{a}'$. Наша задача —

доказать, что $i = j$ и что правее i-й позиции в $\mathbf{a}$ идут одни лишь нули.

$$\begin{array}{rccccccc} \mathbf{a}' = & (& a'_1 & \dots & a'_{i-1} & a'_i & \dots & a'_j & \dots &) \\ & & \| & \| & \| & \wedge & ? & \wedge & \vee\!\!\!\vee & \\ \mathbf{a} = & (& a_1 & \dots & a_{i-1} & a_i & \dots & a_j & \dots &) \end{array}$$

Вначале покажем, что $a_i = a'_i + 1$. Допустим, что это не так, т. е. $a_i \geqslant a'_i + 2$, и придём к противоречию. Рассмотрим вектор $\mathbf{x}$, получаемый из $\mathbf{a}$ уменьшением i-й координаты на единицу. Пусть $\mathbf{x}'$ — канонический вектор, эквивалентный $\mathbf{x}$. Поскольку $\mathbf{x} \in \mathcal{C}_k(\mathbf{a})$, то $\mathbf{x}' \in \mathcal{C}_k^c(\mathbf{a})$. Имеем $\mathbf{x}' \succeq \mathbf{x} \succ \mathbf{a}'$, а это противоречит выбору $\mathbf{a}'$ как вектора с максимальным кодом из $\mathcal{C}_k^c(\mathbf{a})$.

Итак, $a_i = a'_i + 1$. Допустим теперь, что $j > i$ и придём к противоречию. Рассмотрим вектор $\mathbf{y}$, получаемый из $\mathbf{a}$ уменьшением на единицу j-й координаты. Как и раньше, рассмотрев вектор $\mathbf{y}' \in \mathcal{C}_k^c(\mathbf{a})$, эквивалентный $\mathbf{y}$, получаем $\mathbf{y}' \succeq \mathbf{y} \succ \mathbf{a}'$, что противоречит выбору $\mathbf{a}'$.

Итак, $i = j$, и картина, на самом деле, такая:

$$\begin{array}{rccccc} \mathbf{a}' = & (& a_1 & \dots & a_{i-1} & a_i - 1 & \dots &) \\ & & & & & & \vee\!\!\!\vee & \\ \mathbf{a} = & (& a_1 & \dots & a_{i-1} & a_i & \dots &) \end{array}$$

Больше того, если бы какая либо компонента $\mathbf{a}'$ правее i-й была бы строго меньше, чем в $\mathbf{a}$, это противоречило бы тому, что $\mathbf{a}' \in \mathcal{C}_k$ и $\mathbf{a} \in \mathcal{C}_{k+1}$. Значит, у векторов $\mathbf{a}$ и $\mathbf{a}'$ все компоненты, кроме i-й, совпадают. Остаётся доказать, что правее i-й позиции в $\mathbf{a}'$ и $\mathbf{a}$ нет ненулевых значений. Допустим, что это не так, и рассмотрим самую правую ненулевую компоненту в $\mathbf{a}$. Предположив, что это не i-я компонента, имеем такую картину:

$$\begin{array}{rccccccccc} \mathbf{a}' = & (& a_1 & \dots & a_{i-1} & a_i - 1 & \dots & a_l & 0 & \dots & 0 &), \\ \mathbf{a} = & (& a_1 & \dots & a_{i-1} & a_i & \dots & a_l & 0 & \dots & 0 &), \end{array}$$

где $a_l > 0$. Рассматриваем вектор $\mathbf{z}$, получаемый из $\mathbf{a}$ уменьшением a_l на единицу, и, совершенно аналогично предыдущим случаям, получаем противоречие. □

Итак, мы доказали, что для любого $\mathbf{a} \in \mathcal{C}_{k+1}^c$ найдётся такой $\mathbf{a}' \in \mathcal{C}_k^c$ из которого $\mathbf{a}$ может получиться в результате операции порождения. Тем самым, мы доказали выполнение первого условия, необходимого для корректности алгоритма. Очевидно, что вектор $\mathbf{a}'$, о котором шла речь в доказательстве теоремы 1, является *единственным* вектором в $\mathcal{C}_k^c$, из которого в принципе мог бы получиться $\mathbf{a}$ с помощью операции порождения. Обоснуем теперь второе условие:

Теорема 2. *Пусть $\mathbf{a}, \mathbf{b} \in \mathcal{C}_{k+1}^c$ таковы, что $\mathbf{a} \succ \mathbf{b}$. Пусть $\mathbf{a}'$ и $\mathbf{b}'$ — вектора из $\mathcal{C}_k^c$, полученные из $\mathbf{a}$ и $\mathbf{b}$ уменьшением крайних правых ненулевых координат. Тогда $\mathbf{a}' \succeq \mathbf{b}'$.*

Доказательство. Пусть i — номер самой левой координаты, в которой в $\mathbf{a}$ значение больше, чем в $\mathbf{b}$. Рассмотрим два случая:

Если $a_i \geqslant b_i + 2$, то, поскольку при переходе от $\mathbf{a}, \mathbf{b}$ к $\mathbf{a}', \mathbf{b}'$ мы уменьшаем крайние правые ненулевые координаты, мы будем иметь $\mathbf{a}' \succ \mathbf{b}'$, даже если i-я координата была самой правой ненулевой в $\mathbf{a}$.

Теперь рассмотрим случай $a_i = b_i + 1$. Так как суммы координат у $\mathbf{a}$ и $\mathbf{b}$ одинаковы, то у вектора $\mathbf{b}$ есть положительная координата правее i-й. Если правее i-й координаты что-то ненулевое было и в векторе $\mathbf{a}$, то у векторов $\mathbf{a}'$ и $\mathbf{b}'$ первые i координат будут совпадать с координатами $\mathbf{a}$ и $\mathbf{b}$ соответственно, и мы получим $\mathbf{a} \succ \mathbf{b}$. Осталось рассмотреть случай, когда правее i-й координаты в $\mathbf{a}$ идут одни нули. Тогда в $\mathbf{b}$ правее i-й координаты есть только одна единичная координата, а остальные также равны нулю. В этом случае векторы $\mathbf{a}'$ и $\mathbf{b}'$ совпадают: первые их i координат равны соответствующим координатам $\mathbf{b}$, а остальные координаты равны нулю. □

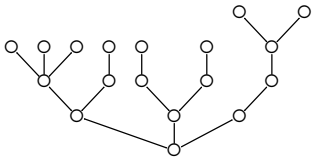
Итак, организовать перебор всех графов несложно, перейдя к перебору векторов. Помимо класса всех графов большой интерес представляют подклассы: регулярных графов, деревьев и другие. Сейчас мы рассмотрим применение описанных выше идей в переборе корневых деревьев.

Корневое дерево — это дерево с выделенной вершиной (корнем). *Упорядоченное* или *посаженное (planted)* дерево — это корневое дерево, в котором фиксирован порядок поддеревьев. Кодом

упорядоченного дерева будем называть двоичный вектор, который получается при прямом обходе дерева (обходе в глубину) и выписывании нуля при движении от корня и единицы при движении к корню. На рис. приведён пример упорядоченного дерева с кодом

$$00010101100111\,0001100111\,0000101111$$

(везде далее мы считаем, что поддеревья упорядочены по часовой стрелке).



Как и в случае с графами, полезно ввести понятие каноничности дерева (кода). Это мы сделаем по индукции. Главными поддеревьями будем называть поддеревья, в которые можно попасть из корня дерева за один шаг. Вершины, смежные с корнем дерева, считаются корнями главных поддеревьев. Итак, упорядоченное дерево считается каноническим, если

- главные поддеревья канонические,
- главные поддеревья упорядочены в дереве по убыванию числа рёбер,
- если пара главных поддеревьев имеют одинаковое число рёбер, то раньше по порядку идёт то из них, которое имеет лексикографически больший код.

Например, дерево на рисунке является каноническим.

Видно, что для любого корневого дерева существует единственное изоморфное ему каноническое упорядоченное дерево. Теперь для описания алгоритма перебора нам нужно ввести порядок на множестве посаженных деревьев и описать, как по каноническому дереву с k рёбрами порождать деревья $(k+1)$ рёбрами.

В отличие от перебора графов, мы будем рассматривать деревья сложности k и строить деревья сложности $(k+1)$ в порядке *возрастания* их кодов, в то время, как графы мы перебирали в порядке убывания кодов.

Пусть T — упорядоченное дерево. Операция порождения будет по дереву T строит последовательность деревьев, каждое из которых добавлением одного ребра к одной из вершин «правой ветви» дерева T, начиная с самой верхней вершины и заканчивая корнем дерева. В терминах кодов, если код дерева T имеет вид $w011\ldots1$, где w — некоторое двоичное слово, то порождаемая последовательность кодов имеет вид $w0\underline{01}1\ldots1$, $w01\underline{01}1\ldots1$, $w011\underline{01}\ldots1$, …, $w011\ldots1\underline{01}$. Каждый из этих кодов получен вставкой «01» в различные места крайней правой последовательности единиц кода T. Видно, что получаемая последовательность кодов упорядочена лексикографически по возрастанию. А значит, третье требование для корректности алгоритма перебора выполнено.

Теперь нужно проверить, что удовлетворены первые два требования, аналогичные тем, что мы проверяли в алгоритме для перебора графов.

Теорема 3. *Пусть T — каноническое дерево. Пусть T' — дерево, полученное из T удалением самого правого-верхнего листа (то есть листа, начиная с которого при обходе дерева выполняется последний спуск к корню). Тогда T' является каноническим.*

Доказательство.

Индукция по числу рёбер в T. Для дерева с двумя вершинами утверждение теоремы с очевидностью выполнено. Пусть теорема выполняется для деревьев с k вершинами, — докажем, что тогда она выполнена и для произвольного дерева T с $(k+1)$ вершинами.

Рассмотрим два случая. Если лист, который был удалён из T, был смежен с корнем T, то он представлял собой главное поддерево в T, причём оно было самым правым. При его удалении никакие другие главные поддеревья не изменились, а значит, не нарушилась и каноничность.

Второй случай — когда удаляемый лист лежал в крайнем правом главном поддереве в T (и был не единственной его вершиной). Тогда если дерево T имело главные поддеревья $T_1, \ldots, T_l$, то дерево T' будет иметь главные поддеревья $T_1, \ldots, T_{l-1}, T'_l$, где T'_l получено из T_l удалением правого-верхнего листа. Так как главные поддеревья были упорядочены в T по убыванию числа рёбер, то тот же порядок сохранится и для главных поддеревьев в T'. А по предположению индукции, так как T_l канонично, то и T'_l канонично. То есть все главные поддеревья в T' каноничны и упорядочены, как нужно, — отсюда T' канонично. Индуктивный переход завершён. □

Отметим, что дерево T', рассматриваемое в теореме 3, является единственным деревом, из которого можно получить T с помощью введённой операции порождения.

В качестве упражнения предлагается доказать по индукции, что выполнено второе требование, необходимое для корректности алгоритма перебора:

Теорема 4. *Пусть T_1 и T_2 — канонические деревья, такие, что код дерева T_1 лексикографически меньше кода T_2, и пусть T'_1 и T'_2 — деревья, из которых T_1 и T_2 получаются в результате операции порождения. Тогда код T'_1 лексикографически меньше кода T'_2.*