Министерство образования Российской Федерации
Ярославский государственный университет им. П.Г. Демидова
Кафедра компьютерных сетей

О сложности дискретных задач

Методические указания

Ярославль 2004

ŋqdqɏeɎ bɆ dәdɆbɎɜә d ɕɔɑbɔɏ Ɉqddɏqɏè

ББК В 311
О 11
УДК 519.6

Составители **В.А. Бондаренко, М.В. Краснов**

О сложности дискретных задач: Метод. указания / Сост. В.А. Бондаренко, М.В. Краснов; Яросл. гос. ун-т. Ярославль, 2004. 23 с.

Приводятся основные факты возникшей тридцать лет назад и уже ставшей классической теории Кука - Карпа. Внимание читателя акцентируется на содержательных аспектах теории.

Предназначены для студентов, обучающихся по дисциплине «Методы построения эффективных алгоритмов» (блок СД), специальность 010200 Прикладная математика и информатика, форма обучения очная.

Рецензент: кафедра дискретного анализа Ярославского государственного университета им. П.Г. Демидова.

© Ярославский государственный университет, 2004
© Бондаренко В.А., Краснов М.В., 2004

ŋqdqɏeɎ bɆ dәdɆbɎɜә d ɕɔɑbɔɏ Ɉqddɏqɏè

В последнее время стало обычным использование терминов "непрерывные задачи" и "дискретные задачи". Установить строгое различие между этими двумя типами математических задач непросто, да и не представляет особого смысла, так как одна и та же задача может рассматриваться как непрерывная и как дискретная в зависимости от выбора подходов к ее решению. Говоря же грубо, можно считать непрерывными те задачи, в постановке которых фигурируют непрерывные множества, например, R – множество всех действительных чисел и т.п. Для описания дискретных задач используются множества дискретной природы, например целочисленные множества.

Систематический интерес к дискретным задачам возник относительно недавно - несколько десятилетий назад. Формирование соответствующих разделов математики в значительной мере стимулировалось появлением и развитием вычислительной техники. Это влияние, постоянно возрастая, проявляется в разных направлениях. С одной стороны, высокопроизводительные компьютеры позволяют применять методы, связанные с вычислениями большого объема, и тем самым расширять круг эффективно решаемых задач. С другой стороны, при конструировании и организации работы современных вычислительных систем появляются новые математические задачи дискретного характера.

1. Примеры дискретных задач

Сформулируем несколько задач. Они окажутся полезными в качестве иллюстрации основных понятий, обсуждаемых далее.

Задача КОММИВОЯЖЕРА (ЗК). Имеется n пунктов, расстояния между любыми двумя из которых известны и представляют собой целые числа. Требуется объехать все пункты, посетив каждый по одному разу и возвратившись в исходный, так, чтобы длина полученного кольцевого маршрута была наименьшей.

Задача О МИНИМАЛЬНОМ ОСТОВНОМ ДЕРЕВЕ. Здесь исходная ситуация аналогична: имеется n пунктов с заданными целочисленными попарными расстояниями. Требуется спроектировать минимальную по протяженности дорожную сеть без дополнительных

узлов, которая позволяла бы из каждого пункта добраться в любой другой.

Задача о составном числе. Для заданного натурального числа z требуется выяснить, является ли оно составным.

Перечень подобного рода задач можно значительно расширить. Но уже приведенные дают основания для обсуждения достаточно характерных общих вопросов. Прежде всего обратим внимание на термин "задача". В тех постановках, которые даны выше, речь идет о массовых задачах, так как предполагается, что в двух первых случаях число пунктов и попарные расстояния между ними могут быть любыми, а в третьей задаче z выбирается также произвольно. Итак, под дискретной задачей мы будем понимать массовую задачу как совокупность индивидуальных задач, соответствующих всевозможным конкретным исходным данным. Примером индивидуальной задачи о составном числе служит такая: выяснить, является ли составным число 12345678910111213.

2. Алгоритмы

Разговор о задаче предполагает рассмотрение средств ее решения, или алгоритмов. Оставляя пока в стороне возможные конкретные алгоритмы, обсудим понятие алгоритма с общей точки зрения. Так как речь идет о массовой задаче, алгоритм должен уметь воспринимать информацию о каждой индивидуальной задаче, уметь перерабатывать эту информацию и, наконец, выдавать результат решения каждой индивидуальной задачи. Эти три части, связанные с функционированием алгоритма, назовем соответственно входом, собственно алгоритмом, или программой, и выходом.

Вход алгоритма представляет собой конечную последовательность цифр, обусловленным образом характеризующую индивидуальную задачу. Под длиной входа индивидуальной задачи z будем понимать количество цифр $l(z)$ в этой последовательности.

Содержательная сторона алгоритма сосредоточена в программе, выполнение которой начинается с подачи входа и завершается формированием выхода. Работа программы осуществляется в виде некоторой последовательности операций, на каждую из которых затрачивается условная единица времени. Характер используемых операций и их последовательность определяются выбором вычислительного

устройства, программы и входа. Обозначим через A алгоритм некоторой задачи, через $t_A(z)$ – число операций, затрачиваемое алгоритмом A на индивидуальную задачу z, и назовем трудоемкостью алгоритма A функцию

$$T_A(L) = \max\{t_A(z):\ \ l(z) \le L\}. \qquad (1)$$

Эта функция характеризует зависимость между длиной входа и временем работы алгоритма. Ясно, что более предпочтительным является такой алгоритм, для которого $T_A(L)$ растет как можно медленней.

Обращаясь теперь к третьему элементу алгоритма - выходу, отметим различие между приведенными выше задачами. Именно в задаче о составном числе выход предусматривает лишь два варианта: "да" или "нет". Задачи подобного вида называются задачами распознавания. Другие две задачи относятся к классу оптимизационных, или экстремальных, задач.

3. Полиномиальная разрешимость

Задачи рассматриваемого вида объединяет то, что для решения каждой из них можно использовать простейший алгоритм – алгоритм прямого перебора, который обозначим через S. Попытаемся грубо оценить трудоемкость переборного алгоритма для приведенных выше задач. Для простоты примем, что в первых двух задачах расстояния между пунктами могут принимать целые значения от 1 до 9. Тогда длина $l(z)$ входа индивидуальной задачи z для n пунктов составляет $\frac{n(n-1)}{2}$. Алгоритм прямого перебора для задачи КОММИВОЯЖЕРА заключается в вычислении длины каждого кольцевого маршрута и выборе среди них минимального. Общее число обходов n пунктов равно, как легко заметить, $\frac{(n-1)!}{2}$, и эта величина служит нижней оценкой числа операций при прямом переборе. Учитывая, что $l(z) < n^2$ и $t_S(z) \ge 2^n$ при $n > 5$, можем оценить снизу трудоемкость (1) алгоритма S:

$$T_S(L) > 2^{\sqrt{L}} \qquad \text{при} \quad L > 10. \qquad (2)$$

Оценка (2) справедлива и для второй задачи, так как при одном и том же n общее число остовных деревьев не меньше количества всех обходов, ведь, "разорвав" кольцевой маршрут, мы получим остовное дерево.

Проведем аналогичные рассуждения для оценки трудоемкости прямого перебора в задаче о составном числе. В этом случае длина входа $l(z)$ равна количеству цифр в десятичной записи числа z, определяющего индивидуальную задачу, т.е. $l(z) < \lg z + 1$. Будем считать, что алгоритм последовательно делит z на натуральные числа, бóльшие единицы и не превосходящие $\sqrt{z}$, до тех пор, пока либо остаток не окажется равным нулю, либо не будут исчерпаны указанные числа. В первом случае выходом служит "да", во втором - "нет". Для получения грубой оценки трудоемкости под числом операцией будем понимать количество выполненных делений. Из этих рассуждений следует искомая оценка:

$$T_S(L) > 10^{\frac{L}{2} - 1}. \qquad (3)$$

Чтобы сделать полученные оценки более осязаемыми, прикинем объем прямого перебора для входа определенной длины. Предположим, что число n пунктов в первых двух задачах равно 100, т.е. $l(z) = 50 \times 99 > 70^2$. Из оценки (2) следует, что для решения индивидуальной задачи придется выполнить не менее 10^{21} операций. Скорость самых современных ЭВМ не превосходит 10^{12} операций в секунду. Следовательно, для получения результата придется ждать 10^9 секунд, или более 30 лет. Оценка (3) приводит к еще более внушительным цифрам. К сказанному можно добавить, что оценки (2) и (3) весьма грубые, действительный рост функции $T_S(L)$ более высок; в то же время при проектировании интегральных схем возникают задачи, подобные первым двум, для значений n, значительно превосходящих 100.

Таким образом, алгоритм прямого перебора, хотя и является универсальным для задач рассматриваемого вида, практически оказывается малоэффективным.

Естественно возникает вопрос о конструировании алгоритмов, принципиально более эффективных, чем примитивный прямой перебор. То, что подобного рода алгоритмы вообще существуют, проиллюстрируем следующим примером.