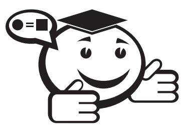
Модуль 7.7. Сложность задач



Сложность задачи *— это асимптотическая временная сложность наилучшего алгоритма, известного для ее решения.*

Основной вопрос теории сложности: насколько успешно или с какой стоимостью может быть решена заданная проблема Q? Мы не имеем в виду никакого конкретного алгоритма решения Q. Наша цель — рассмотреть все возможные алгоритмы решения Q и попытаться сформулировать утверждение о вычислительной сложности, внутренне присущей Q. В то время как всякий алгоритм A для Q дает верхнюю оценку величины сложности Q, нас интересует нижняя оценка. Знание нижней оценки представляет интерес математически и, кроме того, руководит нами в поиске хороших алгоритмов, указывая, какие попытки заведомо будут безуспешны.

Быстрыми являются линейные алгоритмы, которые обладают сложностью порядка $O(n)$, где n — размерность входных данных. К линейным алгоритмам относится школьный алгоритм нахождения суммы десятичных чисел, состоящих из n_1 и n_2 цифр. Сложность этого алгоритма — $O(n_1+n_2)$. Есть алгоритмы, которые быстрее линейных, например алгоритм двоичного поиска в линейном упорядоченном массиве имеет сложность $O(\log n)$, где n — длина массива.

Другие хорошо известные алгоритмы — деление, извлечение квадратного корня, решение систем линейных уравнений и др. — попадают в более общий класс полиномиальных алгоритмов.

Полиномиальным алгоритмом (или алгоритмом полиномиальной временной сложности, или алгоритмом принадлежащим классу $\boldsymbol{P}$) называется алгоритм, у которого временная сложность равна $O(n^k)$, где k — положительное целое число. Алгоритмы, для временной сложности которых не существует такой оценки, называются ***экспоненциальными***, и такие задачи считаются ***труднорешаемыми***. Понятие полиномиально разрешимой задачи принято считать уточнением идеи «практически разрешимой» задачи. Чем объясняется такое соглашение?

Во-первых, используемые на практике полиномиальные алгоритмы обычно действительно работают довольно быстро. Конечно, трудно назвать практически разрешимой задачу, которая требует времени $\Theta(n^{100})$. Однако полиномы такой степени в реальных задачах почти не встречаются.

Второй аргумент в пользу рассмотрения класса полиномиальных алгоритмов — тот факт, что объем этого класса не зависит от выбора конкретной модели вычислений (для достаточно широкого класса моделей) [1]. Например, класс задач, которые могут быть решены за полиномиальное время на последовательной машине с произвольным доступом (RAM), совпадает с классом задач, полиномиально разрешимых на машинах Тьюринга. Класс будет тем же и для моделей параллельных вычислений, если, конечно, число процессоров ограничено полиномом от длины входа.

В-третьих, класс полиномиально разрешимых задач обладает естественными свойствами замкнутости. Например, композиция двух полиномиальных алгоритмов (выход первого алгоритма подается на вход второго) также работает полиномиальное время. Объясняется это тем, что сумма, произведение и композиция многочленов снова есть многочлен.

Приведем примеры классификации задач по их сложности.

Класс *P*

- Рассортировать множество из n чисел. Сложность поведения в среднем порядка $O(n\log n)$ для быстрого алгоритма Хоара [2, с. 198–219].
- Найти эйлеровый цикл на графе из m ребер. В силу теоремы Эйлера мы имеем необходимое и достаточное условие для существования эйлерова цикла и проверка этого условия есть алгоритм порядка $O(m)$.
- Задача Прима—Краскала. *Дана плоская страна и в ней n городов. Нужно соединить все города телефонной связью так, чтобы общая длина телефонных линий была минимальной.* В терминах теории графов задача Прима—Краскала выглядит следующим образом: *Дан граф с n вершинами; длины ребер заданы матрицей* $(d[i,j])$, $i,j=1,\ldots,n$. *Найти остовное дерево минимальной длины.* Эта задача решается с помощью жадного алгоритма сложности $O(n\log n)$ [2, с. 644–661].
- Кратчайший путь на графе, состоящем из n вершин и m ребер. Сложность алгоритма $O(mn)$ [3, с. 105–120].
- Связные компоненты графа. Определяются подмножества вершин в графе (связные компоненты), такие, что две вершины, принадлежащие одной и той же компоненте, всегда связаны цепочкой дуг. Если n — количество вершин, а m — количество ребер, то сложность алгоритма $O(n+m)$ [4, с. 364–365].
- Быстрое преобразование Фурье [3, с. 61–74], требующее $O(n\log n)$ арифметических операций, — один из наиболее часто используемых алгоритмов в научных вычислениях.
- Умножение целых чисел. Алгоритм Шёнхаге—Штрассена [5, с. 304–308]. Сложность алгоритма порядка $O(n\log n\log\log n)$. Отметим, что школьный метод для умножения двух n-разрядных чисел имеет сложность порядка $O(n^2)$.
- Умножение матриц. Алгоритм Штрассена [3, с. 60] имеет сложность порядка $O(n^{\log 7})$ для умножения двух матриц размера $n\times n$. Очевидный алгоритм имеет порядок сложности $O(n^3)$.
- Тест на простоту натурального числа. Алгоритм *AKS* (Агравал, Кайл и Саксен — авторы алгоритма, [6, с. 228–242]) имеет сложность порядка $O(\log^{7,5} n)$, где n — количество цифр в числе.

Класс *E*: задачи, экспоненциальные по природе

К экспоненциальным задачам относятся задачи, в которых требуется построить множество всех подмножеств данного множества, все полные подграфы некоторого графа или же все поддеревья некоторого графа.

Существует масса примеров задач с экспоненциальной сложностью. Например, чтобы вычислить $2^{(2^k)}$ для заданного натурального k, нам только для записи конечного ответа потребуется около 2^n шагов (где n — число цифр в двоичной записи k), не говоря даже о самом вычислении.

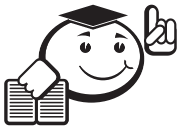
Задачи, не попадающие ни в класс *P*, ни в класс *E*

На практике существуют задачи, которые заранее не могут быть отнесены ни к одному из рассмотренных выше классов. Хотя в их условиях не содержатся экспоненциальные вычисления, однако для многих из них до сих пор не разработан эффективный (т. е. полиномиальный) алгоритм.

К этому классу относятся следующие задачи:

- задача о выполнимости: существует ли для данной булевской формулы такое распределение истинностных значений, что она имеет значение «истина»;
- задача коммивояжера (Коммивояжер хочет объехать все города, побывав в каждом ровно по одному разу, и вернуться в город, из которого начато путешествие. Известно, что переезд из города i в город j стоит $c(i, j)$ руб. Требуется найти путь минимальной стоимости.);
- решение диофантовых систем уравнений;
- составление расписаний, учитывающих определенные условия;
- размещение обслуживающих центров (телефон, телевидение, срочные службы) для максимального числа клиентов при минимальном числе центров;
- оптимальная загрузка емкости (рюкзак, поезд, корабль, самолёт) при наименьшей стоимости;
- оптимальный раскрой (бумага, картон, стальной прокат, отливки), оптимизация маршрутов в воздушном пространстве, инвестиций, станочного парка;
- факторизация — разложение натурального числа на множители [6, с. 254–288].

Заметим, что все эти задачи весьма важны с точки зрения приложений. Поэтому знание того, что полиномиальные алгоритмы для них не найдены, помогут не тратить время на безуспешные поиски точных эффективных алгоритмов, а сосредоточить усилия на создании приближенных или эвристических алгоритмов.



Список литературы по модулю

[1] Катленд Н. Вычислимость. Введение в теорию рекурсивных функций : пер. с англ. / Н. Катленд. — М. : Мир, 1983. — 256 с.

[2] Кормен Т. Алгоритмы: построение и анализ / Т. Кормен, Ч. Лейзерсон, Р. Ривест. — М. : МЦНМО, 2001. — 960 с.

[3] Дасгупта С. Алгоритмы / С. Дасгупта, Х. Пападимитриу, У. Вазирани. — М. : МЦНМО, 2014. — 320 с.

[4] Рейнгольд Э. Комбинаторные алгоритмы. Теория и практика : пер. с англ. / Э. Рейнгольд, Ю. Нивергельт, Н. Део. — М. : Мир, 1980. — 478 с.

[5] Ахо А. Построение и анализ вычислительных алгоритмов / А. Ахо, Дж. Хопкрофт, Дж. Ульман. — М. : Мир, 1979. — 536 с.

[6] Крэндалл Р. Простые числа: Криптографические и вычислительные аспекты / Р. Крэндалл, К. Померанс. — М. : Книжный дом «ЛИБРОКОМ», 2011. — 664 с.