Дисциплина

«РАЗРАБОТКА ЭФФЕКТИВНЫХ АЛГОРИТМОВ»

Лекция 10

ОСНОВНЫЕ КЛАССЫ ТЕОРИИ СЛОЖНОСТИ ВЫЧИСЛЕНИЙ

КЛАССЫ P И NP

Лектор: Михаил Васильевич Ульянов,

muljanov@mail.ru, **8 916 589 94 04**

Обозначения в асимптотическом анализе функций

Поскольку значения функции трудоемкости есть положительные целые числа асимптотические обозначения будут введены в предположении, что функции $f(n)$ и $g(n)$ есть функции положительного целочисленного аргумента $n \geq 1$, имеющие положительные целые значения.

Обозначение 1.1. Оценка Θ (тета)

Функция $f(n) = \Theta\big(g(n)\big)$, если:

$$\exists\ c_1 > 0, c_2 > 0, n_0 > 0 :\ \forall n > n_0 \quad c_1 g(n) \leq f(n) \leq c_2 g(n).$$

При этом говорят, что функция $g(n)$ является асимптотически точной оценкой функции $f(n)$, т. к. функция $f(n)$ отличается от функции $g(n)$ на мультипликативную константу при всех значениях аргумента $n > n_0$. Запись $f(n) = \Theta(1)$ означает, что функция $f(n)$ или равна константе, не равной нулю, или ограничена двумя положительными константами при любых значениях

аргумента $n > n_0$. Более корректно обозначение $\Theta\big(g(n)\big)$ есть обозначение класса функций $f(n)$, которые зажаты между $c_1 g(n) \leq f(n) \leq c_2 g(n)$.

Обозначение 1.2. Оценка O (О большое)

В отличие от оценки Θ, оценка O требует только, чтобы функция $f(n)$ не превышала значения функции $g(n)$ при $n > n_0$, с точностью до мультипликативной константы, а именно: $f(n) = O\big(g(n)\big)$, если:

$$\exists\ c > 0,\ n_0 > 0 :\ \forall n > n_0 \quad 0 \leq f(n) \leq cg(n).$$

Как и в предыдущем определении, запись $O\big(g(n)\big)$ обозначает класс функций, таких, что $0 \leq f(n) \leq cg(n)$, поэтому иногда говорят, что функция $g(n)$ мажорирует функцию $f(n)$. Например, для всех функций:

$$f_1(n) = 12,\ f_2(n) = 5n + 23,\ f_3(n) = n\,ln\,n\,,\ f_4(n) = 7n^2 + 12n - 34$$

будет справедлива оценка $O(n^2)$. Однако, например, для функции $f(n) = 12n^3$ справедлива оценка $O(2^n)$, однако практически она будет мало пригодна.

Обозначение 1.3. Оценка Ω (омега)

В отличие от оценки O, оценка Ω является оценкой снизу — т. е. определяет класс функций, которые растут не медленнее, чем функция $g(n)$ с точностью до положительного постоянного множителя: $f(n) = \Omega\big(g(n)\big)$, если:

$$\exists\ c > 0,\ n_0 > 0 :\ \forall n > n_0 \quad 0 \le cg(n) \le f(n).$$

Например, запись $\Omega(n\,ln\,n)$ обозначает класс функций, которые растут не медленнее, чем $g(n) = n\,ln\,n$, в этот класс попадают, например, все полиномы со степенью большей единицы.

Асимптотическое обозначение O восходит к учебнику Бахмана по теории простых чисел (конец XIX века), обозначения Θ, Ω введены Д. Кнутом:

Knuth D. Big Omicron and Big Omega and Big Theta. SIGACT News 8(2), (1976), pp. 18–24.

Введение в теорию сложности вычислений

Теория сложности оперирует с задачами, а не алгоритмами их решения.

Объект исследования — задачи (сортировка, умножение матриц, сжатие изображений, умножение длинных целых чисел и т.д.).

Предмет исследования — свойства этих задач, определяющие возможность их решения с определенной асимптотической оценкой трудоемкости и введение **классов задач**, обладающих определенными свойствами.

Основной термин — **сложность** задачи. В идеале это попытка теоретически ограничить (в данном классе моделей вычислений, например в RAM классе моделей) снизу по оценке $\Omega\big(g(n)\big)$ любой алгоритм, решающий данную задачу. Такой результат называют «теоретическая нижняя граница задачи». Под *сложностью алгоритма* будем понимать асимптотическую оценку трудоемкости в худшем случае.

Формализмы описания сложностных классов задач

Существует несколько подходов к формальному введению классов задач в теории сложности вычислений — основные из них:

— формализм распознавания языков на машинах Тьюринга;

— формализм сертификатов.

В теории сложности вычислений принят формализм распознавания языков (см. Кормен, Лейзерсон, Риверст и Штайн «Алгоритмы: построение и анализ» и Гэри, Джонсон «Вычислительные машины и труднорешаемые задачи»).

В лекции будет использоваться более простой и понятный формализм сертификатов: для вхождения в некоторый класс задача должна предъявить сертификат, обладающий определенными свойствами. В основном в качестве таких сертификатов используются алгоритмы, имеющие требуемые асимптотические оценки трудоемкости в худшем случае (сложность).

Класс P — класс задач с полиномиальной сложностью или класс полиномиально-решаемых задач

В середине 1960-х годов, в связи с началом широкого использования вычислительной техники для решения практических задач, возник вопрос о границах практической применимости данного алгоритма решения некоторой задачи в смысле ограничений на ее размерность. Какие задачи и каких размерностей могут быть решены на ЭВМ за реальное время? Теоретический ответ на этот вопрос был предложен в работах А. Кобхема (Alan Cobham, 1964), Р. Карпа (Robert Karp) и Дж. Эдмнодса (Jack Edmonds, 1965), в которых был введен сложностной класс задач P.

Задача относится к классу P, если существует константа k и алгоритм, решающий эту задачу за время $O(n^k)$, где n есть длина входа алгоритма в битах. Такой алгоритм и является **сертификатом** включения задачи в класс P.

Отметим, что класс задач P определяется через существование полиномиального по времени алгоритма ее решения, при этом неявно предполагается худший случай по времени для всех различных входов длины n. Интуитивно задачи класса P — это задачи, решаемые за реальное время. Отметим следующие преимущества задач из этого класса:

— для большинства реальных задач из класса P константа k меньше или равна 4, что позволяет прогнозировать приемлемые времена решения задач для практически значимых размерностей входов;

— задачи класса P обладают инвариантностью (в смысле полиномиального времени) в рамках достаточно широкого класса моделей вычислений;

— класс P обладает свойством естественной замкнутости, т. к. сумма или произведение полиномов также есть полином.

Класс NP — класс задач с полиномиально проверяемым решением или класс полиномиально-проверяемых задач

Представим себе, что некоторая *программа* получает решение некоторой задачи. Мы вправе задать вопрос — соответствует ли полученный ответ поставленной задаче, и насколько быстро мы можем проверить его правильность? Рассмотрим в качестве примера задачу о сумме. Дано n целых чисел, содержащихся в массиве $A = \{a_1, \dots, a_n\}$ и целое число V. Постановка задачи: может ли быть представлено число V в виде суммы каких-либо чисел из массива A, т.е. необходимо найти массив $X = \{x_1, \dots, x_n\}, x_i \in \{0,1\}$, такой, что

$$\sum_{k=1}^{n} a_k x_k = V.$$

Если мы получаем массив X, то проверка правильности может быть выполнена с полиномиальной сложностью — очевидно, за $\Theta(n)$ операций.

Формальное описание процесса проверки решения определенной задачи, полученного некоторым алгоритмом.

Поставим в соответствие каждому входу $D, |D| = n$ результат S, такой, что $|S| = O(n^l)$, где l — некоторая константа, и алгоритм $A_S = A_S(D, S)$, такой, что он выдает «1», если проверяемый результат верен, и «0», если нет. Тогда задача принадлежит сложностному классу NP, если существует константа m, и алгоритм A_S имеет временную сложность не более чем $O(n^m)$. Такой алгоритм и является сертификатом для вхождения задачи в класс NP.

Содержательно задача относится к классу NP, если ее решение, полученное некоторой *программой*, может быть проверено с полиномиальной временной сложностью. Иначе говоря, задача относится к классу NP, если *алгоритм*, проверяющий решение этой задачи относится к классу P. Класс NP был впервые введен в работах Эдмондса.

Проблема $P = NP$?

После введения в теорию алгоритмов понятий сложностных классов, Эдмондсом (Edmonds, 1965) была сформулирована основная проблема теории сложности — $P = NP$?, и высказана гипотеза о несовпадении этих классов.

Словесно проблему можно сформулировать следующим образом: можно ли все задачи, решение которых проверяется с полиномиальной сложностью, решить также за полиномиальное время?

Очевидно, что любая задача, принадлежащая классу P, принадлежит и классу NP, т. к. она может быть полиномиально проверена, при этом задача проверки решения может состоять просто в повторном решении задачи полиномиальным алгоритмом.

На сегодня **отсутствуют** теоретические доказательства как совпадения классов P и NP, так и их несовпадения — проблема открыта с двух сторон!

В настоящее время предположение состоит в том, что класс P является собственным подмножеством класса NP, т. е. множество задач $NP \backslash P$ не пусто, как это показано на рисунке 1. Это предположение опирается на существование еще одного класса, а именно класса NPC или класса NP -полных задач, которому будет посвящена следующая лекция.

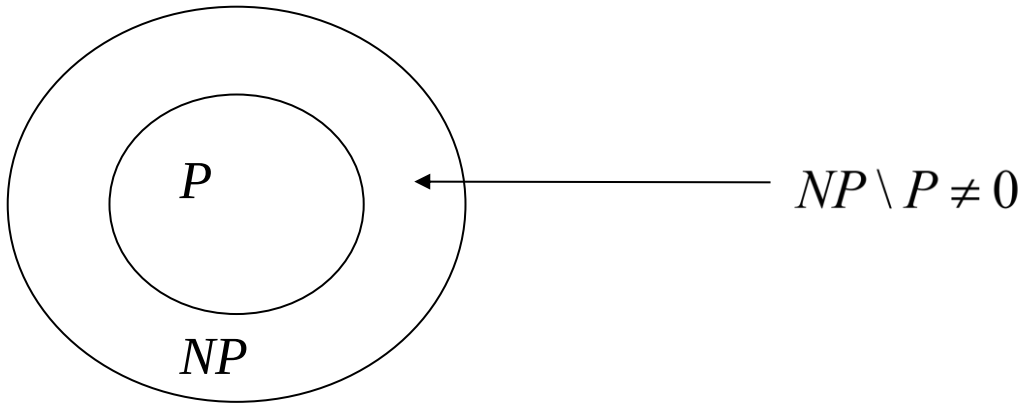

Рис. 1. Соотношение классов P и NP.

Список литературы к лекции 10

[B.1.] Cobham A. The intrinsic computational difficulty of functions // Proc. Congress for Logic, Mathematics, and the Philosophy of Science. – North Holland, Amsterdam, 1964. pp. 24–30.

[B.2.] Cook S. C. The complexity of theorem–proving procedures // Third ACM Symposium on Theory of Computing., — ACM, New York, 1971. pp. 151–158.

[B.3.] Karp R. M. Reducibility among combinatorial problems // Complexity of Computer Computations / R. E. Miller, ed., – Plenum Press, New York, 1972. pp. 85–104.

[B.4.] Knuth D. Big Omicron and Big Omega and Big Theta. SIGACT News 8(2), (1976), pp. 18–24.

[B.5.] Кормен Т., Лейзерсон Ч., Ривест Р., Штайн К. Алгоритмы: построение и анализ, 2-ое издание: Пер. с англ. — М.: Издательский дом «Вильямс», 2005. — 1296 с.

[B.6.] Гэри М., Джонсон Д. Вычислительные машины и труднорешаемые задачи. — М.: Мир, 1982. — 416 с.