Тема 1. Алгоритмы и их сложности

1.2. Асимптотические обозначения

В большинстве случаев поставленная перед разработчиком задача может иметь различные методы и соответственно различные алгоритмы решения. Поэтому возникают вопросы оценки качества алгоритмов, сравнения характеристик различных алгоритмов и выбора на основе такого анализа наиболее эффективного алгоритма решения поставленной задачи.

Для оценки алгоритмов существует много критериев, часто противоречащих друг другу. В частности, такие требования, как простота алгоритма для понимания, эффективность использования ресурсов и быстрое выполнение не всегда совместимы. Существуют алгоритмы, к которым могут предъявляться и свои требования, связанные со спецификой решаемых задач. Например, в численных алгоритмах точность и устойчивость являются не менее существенными критериями оценки.

Важным критерием эффективности алгоритма является порядок роста необходимых для решения задачи времени и емкости памяти при увеличении входных данных (*вычислительная сложность*). Мера количества входных данных для каждой конкретной задачи связывается с некоторым числом, называемым *размером задачи*. Время, затрачиваемое алгоритмом на достижение результата, как функция размера задачи называется *временной сложностью* этого алгоритма. Аналогичным образом можно определить *емкостную сложность*. В учебном пособии основное внимание будет уделяться временной сложности, поскольку для многих рассматриваемых алгоритмов емкостная сложность очевидна.

Для определения временной сложности во многих случаях достаточно оценить асимптотику роста времени работы алгоритма при стремлении размера входа к бесконечности, т. е. определить скорость роста функции (*асимптотическая временная сложность*). Тогда если асимптотика одного алгоритма меньше другого, то в большинстве случаев можно сделать заключение о том, что он более эффективен с точки зрения времени работы (возможно, за исключением малых размеров входа).

Скорость роста функций описывается с помощью асимптотических обозначений. Поскольку размер входа и время работы алгоритма не могут быть отрицательными, предполагается, что при рассмотрении асимптотических соотношений все функции неотрицательны.

Говорят, что функция $f(n)$ *асимптотически равна* функции $g(n)$ (функция $f(n)$ растет с такой же скоростью, как и $g(n)$), и записывают $f(n) \sim g(n)$, если

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1.$$

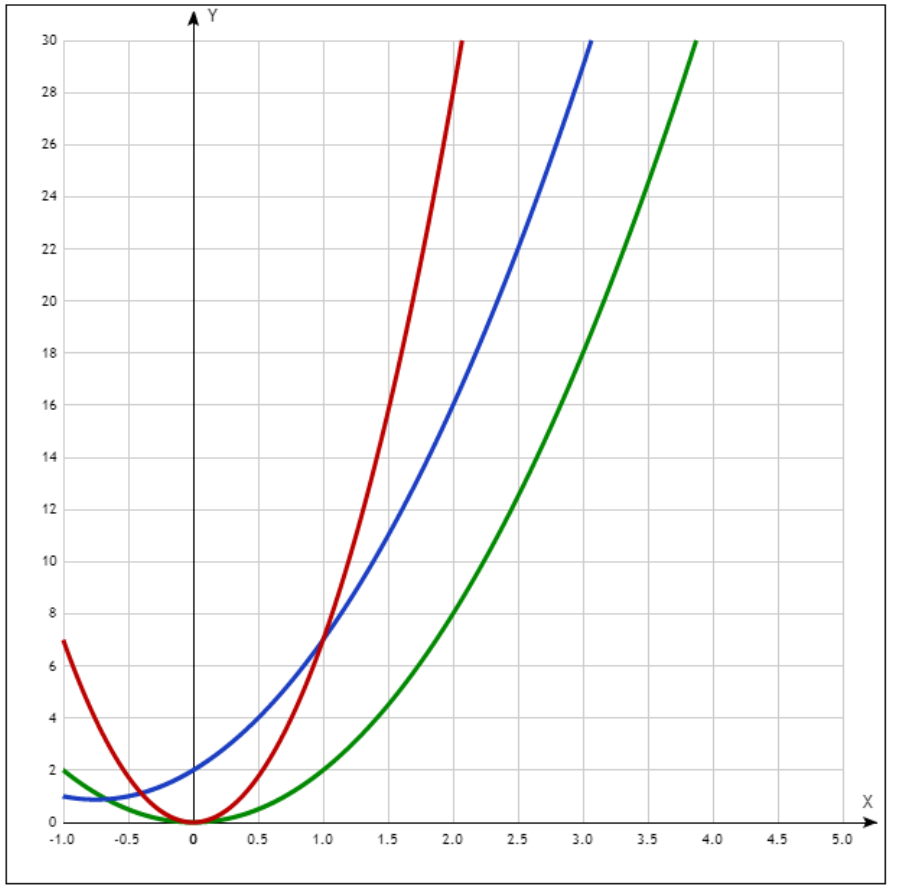
Говорят, что функция $g(n)$ является *асимптотически точной оценкой* функции $f(n)$, и записывают $f(n) = \Theta(g(n))$, если существуют константы $c_1, c_2 > 0$ и $n_0 \geq 0$, такие, что

$$c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ для всех } n \geq n_0.$$

Запись $f(n) = \Theta(g(n))$ читается «$f(n)$ есть тэта от $g(n)$». Данное асимптотическое соотношение симметрично: из $f(n) = \Theta(g(n))$ следует $g(n) = \Theta(f(n))$. Пусть $f(n) = 2n^2 + 3n + 2$, $g(n) = n^2$. Согласно определению должны выполняться неравенства

$$c_1 n^2 \leq 2n^2 + 3n + 2 \leq c_2 n^2$$

для всех $n \geq n_0$. Очевидно, что первое неравенство выполняется при $c_1 = 2$, а второе – при $n_0 = 1$ (т. е. для всех $n \geq 1$) и $c_2 = 7$. Таким образом, $2n^2 + 3n + 2 = \Theta(n^2)$. Следует отметить, что правое неравенство никогда не выполняется для $n < 1$: при $n = 0$ имеет место $2 \leq c_2 0^2$, что невозможно при любой константе $c_2 > 0$.



Асимптотическое соотношение $f(n) = \Theta(g(n))$ включает в себя верхнюю и нижнюю оценки. Эти оценки можно разделить.

Говорят, что функция $g(n)$ является *верхней границей скорости роста* функции $f(n)$ (или функция $f(n)$ растет не быстрее, чем функция $g(n)$), и записывают $f(n) = O(g(n))$, если существуют константы $c > 0$ и $n_0 \geq 0$, такие что

$$f(n) \leq c\,g(n) \text{ для всех } n \geq n_0.$$

Запись $f(n) = O(g(n))$ читается «$f(n)$ есть *о* большое от $g(n)$» или «$f(n)$ имеет порядок (степень) роста $O(g(n))$» и означает, что с ростом n отношение $f(n)/g(n)$ остается ограниченным.

Говорят, что функция $g(n)$ является *нижней границей скорости роста* функции $f(n)$ (или функция $f(n)$ растет не медленнее, чем функция $g(n)$), и записывают $f(n) = \Omega(g(n))$, если существуют константы $c > 0$ и $n_0 \geq 0$, такие что

$$c\,g(n) \leq f(n) \text{ для всех } n \geq n_0.$$

Запись $f(n) = \Omega(g(n))$ читается «$f(n)$ есть омега большое от $g(n)$».

Для любых двух функций $f(n)$ и $g(n)$ асимптотическое отношение $f(n) = \Theta(g(n))$ выполняется тогда и только тогда, когда $f(n) = O(g(n))$ и $f(n) = \Omega(g(n))$. Например,

$$f(n) = 2n^2 + 3n + 2 = \Theta(n^2) = O(n^2) = \Omega(n^2).$$

В соответствии с определением можно показать, что функция $f(n) = 2n^2 + 3n + 2$ имеет порядок роста $O(n^3)$, однако это более слабое утверждение, чем то, что $f(n)$ имеет порядок роста $O(n^2)$. В этом случае $2n^2 + 3n + 2 \neq \Omega(n^3)$ и $2n^2 + 3n + 2 \neq \Theta(n^3)$. Поэтому при определении отношения $f(n) = O(g(n))$ стремятся трактовать его как верхнюю границу отношения $f(n) = \Theta(g(n))$.

Говорят, что функция $f(n)$ растет медленнее, чем функция $g(n)$, и записывают $f(n) = o(g(n))$, если

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0.$$

Запись $f(n) = o(g(n))$ читается «$f(n)$ есть o малое от $g(n)$». Например, $2n^2 + 3n + 2 = o(n^3)$, но $2n^2 + 3n + 2 \neq O(n^3)$.

Аналогичным образом вводится ω-обозначение: говорят, что функция $f(n)$ растет быстрее, чем функция $g(n)$, и записывают $f(n) = \omega(g(n))$, если

$$\lim_{n\to\infty} \frac{g(n)}{f(n)} = 0.$$

Запись $f(n) = \omega(g(n))$ читается «$f(n)$ есть омега малое от $g(n)$». Например, $2n^2 + 3n + 2 = \omega(n)$, но $2n^2 + 3n + 2 \neq \Omega(n)$.

Важным частным случаем асимптотических обозначений является случай, когда $g(n) = 1$, например, $O(1)$. Это говорит о том, что соответствующая функция ограничена некоторой положительной константой и не зависит от n. С точки зрения временной сложности обозначение $O(1)$ говорит о том, что время работы постоянно и не зависит от размера входа.

Следует всегда иметь в виду, что в асимптотических отношениях левые и правые части не симметричны, так как правая часть всегда содержит меньше информации, чем левая. Поэтому нельзя заменять левую часть выражения правой. Например, из двух отношений $f(n) = O(g(n))$ и $h(n) = O(g(n))$ не вытекает ложное утверждение, что $f(n) = h(n)$.

Для оценки алгоритмов в качестве основной меры эффективности выполнения алгоритма используется асимптотическая временная сложность. При этом чаще всего учитываются верхние границы роста скорости временной сложности как функции от размера входа. Например, утверждение о том, что временная сложность некоторого алгоритма есть $O(n^2)$ (можно читать как «порядка n^2»), т. е. время работы алгоритма асимптотически ограничено сверху квадратичной функцией, более полезно с точки зрения оценки алгоритма, чем утверждение о том, что временная сложность алгоритма асимптотически ограничена снизу некоторой функцией. Поэтому в литературе для определения временной сложности алгоритмов в большинстве случаев используется O-символика, т. е. запись $f(n) = O(g(n))$ трактуется как верхняя граница отношения $f(n) = \Theta(g(n))$. При этом можно сказать, что функция $f(n)$ асимптотически равна функции $g(n)$, хотя с формальной точки зрения это не совсем корректно, поскольку нижняя граница роста не является частью определения $O(g(n))$.

При анализе алгоритмов с использованием O-символики могут выполняться различные операции, в частности, сложение и умножение. Пусть $T_1(n)$ – время выполнения некоторого фрагмента F_1 алгоритма, $T_2(n)$ – фрагмента F_2, $T_1(n)$ имеет временную сложность $O(f(n))$ и $T_2(n) - O(g(n))$. Тогда

$$T_1(n) + T_2(n) = O(\max(f(n), g(n))),$$
$$T_1(n)T_2(n) = O(f(n)g(n)).$$

Правило сложения используется, когда необходимо определить порядок роста времени последовательного выполнения фрагментов F_1 и F_2 алгоритма. Например, $O(n^2 + n) = O(n^2)$. Из правила умножения следует, что $O(c\,f(n)) = O(f(n))$, если c – положительная константа, т. е. $c = O(1)$.