Модуль 7.5. Асимптотические обозначения

Введем в первую очередь обозначение, связанное с асимптотической оценкой функций. Хотя во многих случаях эти обозначения используются неформально, полезно начать с точных определений (см. [1] или [2]).

Пусть даны две функции $f(n)$ и $g(n)$ натурального аргумента n, значениями которого являются положительные действительные числа. Говорят, что функция g ***мажорирует*** функцию f (или «f растет не быстрее g»), если существует действительное положительное число c и натуральное число n_0, такое, что $f(n) \leqslant cg(n)$ для всех $n \geqslant n_0$. Если g мажорирует f, это обозначается как $f(n) = O\big(g(n)\big)$. Символ $O\big(g(n)\big)$ читается как «*O* ***большое от*** $g(n)$»; при этом говорят, что $f(n)$ имеет порядок O большое от $g(n)$. Также говорят, что функция g является асимптотически верхней оценкой для функции f.

Другими словами, $f = O(g)$ означает, что отношение $f(n)/g(n)$ ограничено сверху некоторой константой.

Пример 1

Проверим, что $(1/2)n^2 - 3n = O(n^2)$. Согласно определению надо указать положительную константу c и число n_0 так, чтобы неравенство

$$\frac{n^2}{2} - 3n \leqslant cn^2$$

выполнялось для всех $n \geqslant n_0$. Разделим на n^2:

$$\frac{1}{2} - \frac{3}{n} \leqslant c.$$

Видно, что для выполнения неравенства достаточно положить $c = 1/2$ и $n_0 = 1$.

Пример 2

При $a > 0$ можно записать $an + b = O(n^2)$ (положим, $c = a + |b|$ и $n_0 = 1$).

Пример 3

Покажем, что $6n^3 \neq O(n^2)$. В самом деле, пусть найдутся такие c и n_0, что $6n^3 \leqslant cn^2$ для всех $n \geqslant n_0$. Но тогда $n \leqslant c/6$ для всех $n \geqslant n_0$, что невозможно.

Определение $O(g(n))$ предполагает, что функции $f(n)$ и $g(n)$ асимптотически неотрицательны, т. е. неотрицательны для достаточно больших значений n. Заметим, что если f и g строго положительны, то можно исключить n_0 из определения (изменив константу c так, чтобы для малых n неравенство также выполнялось).

Отыскивая асимптотически верхнюю оценку для суммы, мы можем отбрасывать члены меньшего порядка, которые при больших n становятся малыми по сравнению с основным слагаемым. Заметим также, что коэффициент при старшем члене роли не играет (он может повлиять только на выбор константы c). Например, рассмотрим квадратичную функцию $f(n) = an^2 + bn + d$, где a, b и d — некоторые константы и $a > 0$. Отбрасывая члены младших порядков и коэффициент при старшем члене, находим, что $f(n) = O(n^2)$. Чтобы убедиться в этом формально, можно положить $c = 3 \times \max(a, b, d)$ и $n_0 = 1$.

Упомянем важный частный случай использования O-обозначений: $O(1)$ обозначает ограниченную сверху функцию.

Обозначение $O(\cdot)$ можно считать аналогом $\leqslant$. Аналоги для $\geqslant$ и $=$ также существуют: $f(n) = \Omega(g(n))$ (f растет не медленее g, с точностью до константы) означает $g(n) = O(f(n))$; $f(n) = \Theta(g(n))$ (f и g имеют одинаковый порядок роста) означает что $f(n) = O(g(n))$ и $g(n) = O(f(n))$.

Работая с символами O, Ω и Θ, мы имеем дело с *односторонними* равенствами — эти символы могут стоять только справа от знака $=$.

Отношение «O большое» для функций обладает рефлексивностью и транзитивностью: $f(n) = O(f(n))$, $f(n) = O(g(n))$ и $g(n) = O(h(n))$ влечет $f(n) = O(h(n))$. Очевидно, такие же свойства у отношений $f(n) = \Omega(g(n))$ и $f(n) = \Theta(g(n))$, но последнее еще симметрично.

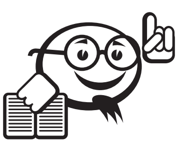
Рассмотрим отношение «O большое» для сравнения асимптотического роста конкретных функций.

Для положительных целых чисел r и s следующую теорему можно доказать методом индукции. Справедливость утверждения теоремы для положительных рациональных чисел r и s можно показать, не используя логарифмы.

Теорема 1. Если r и s — действительные числа, $r \leqslant s$ и $n > 1$, тогда $n^r \leqslant n^s$. Следовательно, $n^r = O(n^s)$.

Доказательство. Функция $\ln(x)$ — возрастающая, поэтому $a \leqslant b$ тогда и только тогда, когда $\ln(a) \leqslant \ln(b)$. Отсюда $n^r \leqslant n^s$ тогда и только тогда, когда $\ln(n^r) \leqslant \ln(n^s)$, что, в свою очередь, выполняется тогда и только тогда, когда $r\ln(n) \leqslant s\ln(n)$, т. е. тогда и только тогда, когда $r \leqslant s$, поскольку $\ln(n)$ для $n > 1$ — величина положительная.

Следующие теоремы показывают, что свойство функции иметь порядок $O(g(n))$ замкнуто относительно операций сложения и умножения на число.



Теорема 2. Если $f(n) = O(g(n))$, то $kf(n) = O(g(n))$.

Доказательство. По определению, $f(n) \leqslant cg(n)$ для некоторого положительного действительного числа c и всех $n \geqslant n_0$. Поэтому

$$kf(n) \leqslant ckg(n)$$

и $cf(n) = O\big(g(n)\big)$.

Теорема 3. Если $f(n) = O\big(g(n)\big)$ и $h(n) = O\big(g(n)\big)$, то $(f+h)(n) = O\big(g(n)\big)$.

Доказательство. По определению, для некоторого постоянного k и некоторого целого числа m_1 имеем $f(n) \leqslant kg(n)$ для всех $n > m_1$. Опять же по определению, для некоторого постоянного l и некоторого целого числа m_2 имеем $h(n) \leqslant lg(n)$ для всех $n > m_2$. Пусть $m = \max(m_1, m_2)$. Следовательно, для всех $n > m$

$$f(n) + h(n) \leqslant kg(n) + lg(n) = (k+l)g(n)$$

и $(f+g)(n) = O\big(g(n)\big)$.

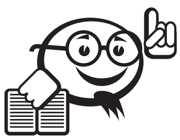
Теорема 4. Если $f(n) = O\big(g(n)\big)$ и $h(n) = O\big(e(n)\big)$, то $(f \times h)(n) = O(g \times e)(n)$.

Доказательство. По определению, для некоторого постоянного k и некоторого целого числа m_1 имеем $f(n) \leqslant kg(n)$ для всех $n > m_1$. Опять же по определению, для некоторого постоянного l и некоторого целого числа m_2 имеем $h(n) \leqslant le(n)$ для всех $n > m_2$. Пусть $m = \max(m_1, m_2)$. Следовательно, для всех $n > m$

$$f(n) \times h(n) \leqslant kg(n)le(n) = (kl)g(n) \times e(n)$$

и $(f \times g)(n) = O(g \times e)(n)$.

Следующая теорема устанавливает мажоранту для полинома.



Теорема 5.

Если $p(n) = a_k n^k + a_{k-1}n^{k-1} + \ldots + a_1 n + a_0$, то $p(n) = O(n^k)$.

Доказательство.

$$p(n) \leqslant |a_k n^k + a_{k-1}n^{k-1} + \ldots + a_1 n + a_0| \leqslant$$

(в силу неравенства треугольника: $|A+B| \leqslant |A| + |B|$)

$$\leqslant |a_k n^k| + |a_{k-1}n^{k-1}| + \ldots + |a_1 n| + |a_0| =$$

$$= |a_k|n^k + |a_{k-1}|n^{k-1} + \ldots + |a_1|n + |a_0| \leqslant \text{ (по теореме 1)}$$

$$\leqslant |a_k|n^k + |a_{k-1}|n^k + \ldots + |a_1|n^k + |a_0|n^k = \big(|a_k| + |a_{k-1}| + \ldots + |a_1| + |a_0|\big)n^k$$

и $p(n) = O(n^k)$.

Теорема 6. Для целых чисел a и b, больших единицы, $\log_a(n) = O(\log_b(n))$.

Доказательство. Следует непосредственно из равенства

$$\log_a(n) = \frac{\log_b(n)}{\log_a(b)}.$$

Теорема 7. Пусть n — неотрицательное целое число. Тогда $n < 2^n$ и, следовательно, $n = O(2^n)$.

Доказательство. Воспользуемся индукцией, имея для $n = 0$, $0 < 2^0 = 1$. Допустим, что $k < 2^k$, тогда

$$k + 1 \leqslant k + k \leqslant 2^k + 2^k = 2^{k+1}$$

и, по индукции, $n < 2^n$.

Следующие теоремы дают ответ на вопрос о том, какие функции могут выступать в роли мажорант для других функций.

Теорема 8. Для целых чисел a, больших единицы, $\log_a(n) = O(n)$.

Доказательство. Согласно теореме 7 имеет место неравенство $n < 2^n$. Поэтому $\log_2(n) < \log_2(2^n) = n$ и $\log_2(n) = O(n)$. Поскольку по теореме 6 имеем $\log_a(n) = O(\log_2(n))$, то по транзитивности получаем $\log_a(n) = O(n)$.

Теорема 9. Пусть n — неотрицательное целое число, тогда $n! < n^n$ и, следовательно, $n! = O(n^n)$.

Теорема 10. Пусть $a > 1$ и n — неотрицательное целое число, тогда $\log_a(n!) \leqslant n\log_a(n)$ и, следовательно, $\log_a(n!) = O(n\log_a(n))$.

Благодаря введенной символике мы можем заменить $7n^2 + 3n + 1$ на $\Theta(n)$, пренебрегая остальными слагаемыми. Вот несколько общих правил такого рода замен:

1. Постоянные множители можно опускать. Например, $\pi n^2 + \varepsilon n^3$ можно заменить на n^3.
2. Любая экспонента растет быстрее любого полинома. Так, например, 2^n растет быстрее n^{1000}.
3. Любой полином растет быстрее любого логарифма. Например, n (и даже $\sqrt{n}$) растет быстрее $(\log n)^3$.

Покажем теперь, как мы можем использовать O-обозначения для оценки количества определенных действий в алгоритмах.

Пример 4

Определим число арифметических операций, необходимых для сложения двух матриц.

Пусть матрицы имеют размеры $m \times k$. Тогда алгоритм сложения матриц $A + B = C$ можно описать на Паскале следующим образом:

```
for i := 1 to m do
    for j := 1 to k do
        C[i, j] := A[i, j] + B[i, j];
```

Как видим, сложение выполняется для каждого i и каждого j. Поскольку i принимает m значений, а j принимает k значений, то выполняется mk операций сложения. Пусть $n = \max(m, k)$. Тогда число выполняемых арифметических операций имеет порядок $O(n^2)$.

Пример 5

Определим число арифметических операций, необходимых для умножения двух матриц.

Пусть матрицы A и B имеют размеры $m \times p$ и $p \times k$ соответственно. Тогда алгоритм умножения матриц $A \times B = C$ можно описать на Паскале следующим образом:

```
for i := 1 to m do
    for j := 1 to k do
        begin
        C[i,j] := 0;
        for s := 1 to p do
            C[i, j] := C[i, j] + A[i, s] * B[s, j];
        end;
```

Поскольку s принимает значения от 1 до p, то выполняется p операций сложения и p операций умножения. Величина s изменяется от 1 до p для каждого i и каждого j, поэтому s пробегает значения от 1 до pmk раз. Таким образом, выполняется mkp операций сложения и столько же операций умножения. Следовательно, всего выполняется $2mkp$ операций. Пусть $n = \max(m, k, p)$. Тогда число выполняемых арифметических операций имеет порядок $O(n^3)$.

Пример 6

Сравним количество операций, которое требуется для непосредственного вычисления значения многочлена традиционным способом и по схеме Горнера.

Пусть требуется вычислить $p(c)$, где $p(x) = a_n x^n + a_{n-1}x^{n-1} + \ldots + a_1 x + a_0$. Если $p(c)$ вычисляется непосредственно, то для подсчета c^k требуется выполнить $k - 1$ операций умножения. Еще одна операция нужна для умножения на a_k, так что вычисление $a_k x^k$ требует k операций умножения. Таким образом, нужно выполнить $1 + 2 + \ldots + n = n(n-1)/2$ умножений. Для того чтобы найти сумму $n+1$ слагаемых, требуется выполнить n сложений, так что общее число арифметических операций равно $n(n-1)/2 + n$ и имеет порядок $O(n^2)$.

При вычислении полинома $a_4x^4 + a_3x^3 + a_2x^2 + a_1x + a_0$ по схеме Горнера мы переписываем полином в виде $x \cdot \Big(x \cdot \big(x \cdot (a_4x + a_3) + a_2\big) + a_1\Big) + a_0$ и замечаем, что выражение включает четыре операции умножения и четыре операции сложения. Очевидно, в общем случае

$$p(x) = x \cdot \Big(x \cdot \big(\ldots\big(x \cdot (a_n x + a_{n-1}) + a_{n-2}\big) + \ldots a_2\big) + a_1\Big) + a_0$$

включает n операций сложения и n операций умножения. Таким образом, общее число арифметических операций равно $2n$ и имеет порядок $O(n)$.

Список литературы по модулю

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

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