Математические основы информатики

Сложность вычислений.

Сергей Леонидович Бабичев

Definition (символ Θ)

Для данной функции $g(n)$ запись $\Theta(g(n))$ обозначает множество функций, удовлетворяющих условию

$$\Theta(g(n)) = \{f(n) : \exists c_1, c_2 > 0, N_0 : 0 \leqslant c_1 g(n) \leqslant f(n) \leqslant c_2 g(n) \ \ \forall n \geqslant N_0\}$$

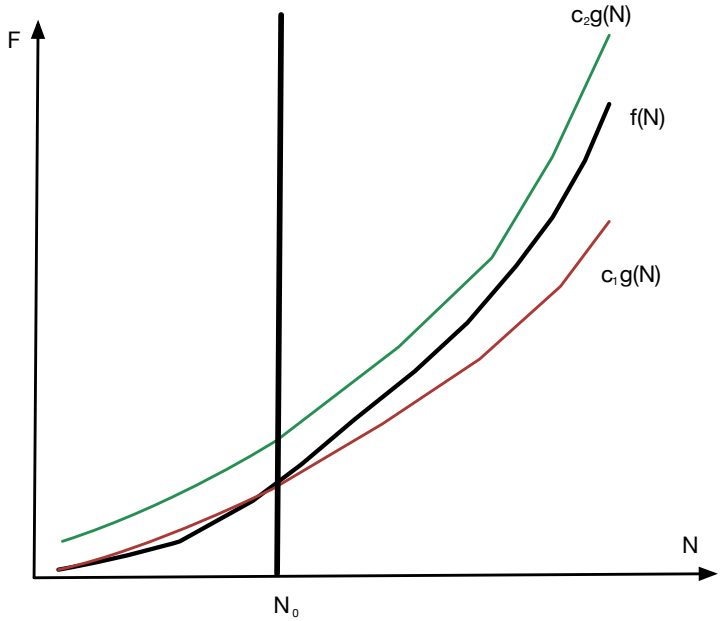
- $\Theta(g(n))$ есть множество и корректна запись $f(n) \in \Theta(g(n))$.
- Запись $f(n) = \Theta(g(n))$ будет далее использоваться как эквивалентная.
- Говорят, что функция $g(n)$ является асимптотически точной оценкой $f(n)$.

Definition (асимптотическая неотрицательность)

Функция $f(n)$ является *асимптотически неотрицательной*, если
$\exists N_0 : \forall n > N_0 \quad f(n) \geqslant 0$.

Definition (асимптотическая положительность)

Функция $f(n)$ является *асимптотически положительной*, если
$\exists N_0 : \forall n > N_0 \quad f(n) > 0$.

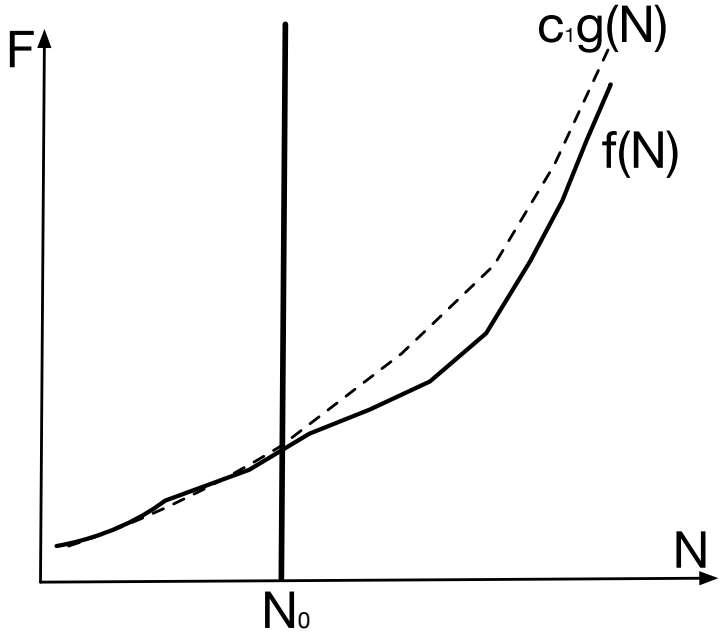

F
$c_2g(N)$
f(N)
$c_1g(N)$
N
N_0

Definition (символ O)

Для данной функции $g(n)$ запись $O(g(n))$ обозначает множество функций, удовлетворяющих условию

$$O(g(n)) = \{f(n) : \exists c > 0, N_0 : 0 \leqslant f(n) \leqslant cg(n) \ \ \forall n \geqslant N_0\}$$

- $O(g(n))$ есть множество и корректна запись $f(n) \in O(g(n))$.
- Запись $f(n) = O(g(n))$ будет далее использоваться как эквивалентная.
- Говорят, что функция $g(n)$ является асимптотически верхней оценкой $f(n)$.
- Из оценки $f(n) = \Theta(g(n))$ следует $f(n) = O(g(n))$.


F
$c_1 g(N)$
f(N)
N
N_0

Definition (символ Ω)

Для данной функции $g(n)$ запись $\Omega(g(n))$ обозначает множество функций, удовлетворяющих условию

$$\Omega(g(n)) = \{f(n) : \exists c > 0, N_0 : 0 \leqslant cg(n) \leqslant f(n) \ \ \forall n \geqslant N_0\}$$

- $\Omega(g(n))$ есть множество и корректна запись $f(n) \in \Omega(g(n))$.
- Запись $f(n) = \Omega(g(n))$ будет далее использоваться как эквивалентная.
- Говорят, что функция $g(n)$ является асимптотически нижней оценкой $f(n)$.

Theorem (о связи символов Θ, O и Ω.)

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

Definition (символ o)

Для данной функции $g(n)$ запись $o(g(n))$ обозначает множество функций, удовлетворяющих условию

$$o(g(n)) = \{f(n) : \forall c > 0 \;\; \exists N_0 : 0 \leqslant f(n) \leqslant cg(n) \;\; \forall n \geqslant N_0\}$$

- Говорят, что верхняя граница $g(n)$ не является асимптотически верхней оценкой $f(n)$.
- Отличия $O(g(n))$ и $o(g(n))$: в первом случае ограничивается для *некоторой* константы $c > 0$, во-втором — для *всех* констант $c > 0$.

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

Definition (символ ω)

Для данной функции $g(n)$ запись $\omega(g(n))$ обозначает множество функций, удовлетворяющих условию

$$\omega(g(n)) = \{f(n) : \forall c > 0 \exists N_0 : 0 \leqslant cg(n) \leqslant f(n) \ \ \forall n \geqslant N_0\}$$

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

Сравнение функций

Для асимптотически положительных функций $f(n)$ и $g(n)$ верны свойства транзитивности:

$$
\begin{aligned}
f(n) = \Theta(g(n)) \wedge g(n) = \Theta(h(n)) &\implies f(n) = \Theta(h(n)) \\
f(n) = O(g(n)) \wedge g(n) = O(h(n)) &\implies f(n) = O(h(n)) \\
f(n) = \Omega(g(n)) \wedge g(n) = \Omega(h(n)) &\implies f(n) = \Omega(h(n)) \\
f(n) = o(g(n)) \wedge g(n) = o(h(n)) &\implies f(n) = o(h(n)) \\
f(n) = \omega(g(n)) \wedge g(n) = \omega(h(n)) &\implies f(n) = \omega(h(n))
\end{aligned}
$$

Рефлексивности:

$$
\begin{aligned}
f(n) &= \Theta(f(n)) \\
f(n) &= O(f(n)) \\
f(n) &= \Omega(f(n))
\end{aligned}
$$

Сравнение функций

Симметрия:

$$f(n) = \Theta(g(n)) \Leftrightarrow g(n) = \Theta(f(n))$$

Антисимметрия:

$$f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$$
$$f(n) = o(g(n)) \Leftrightarrow g(n) = \omega(f(n))$$

Сравнение функций: аналогия с числами

$$
\begin{aligned}
f(n) &= O(g(n)) \sim a \leqslant b \\
f(n) &= \Omega(g(n)) \sim a \geqslant b \\
f(n) &= \Theta(g(n)) \sim a = b \\
f(n) &= o(g(n)) \sim a < b \\
f(n) &= \omega(g(n)) \sim a > b
\end{aligned}
$$

Свойство *трихотомии*: для любых действительных чисел a, b выполняется только одно соотношение $a < b$, $a = b$, $a > b$.
Для асимптотических символов это неверно. Для функций $f(n)$ и $g(n)$ может не выполняться ни $f(n) = O(g(n))$, ни $f(n) = \Omega(g(n))$.

Разделяй и властвуй.

- В первой лекции мы хотели получить произведение двух длинных чисел с помощью алгоритма Карацубы.
- При классическом умножении в столбик

$$N_1 \times N_2 = T^2 x_1 x_2 + T((x_1 y_2 + x_2 y_1) + y_1 y_2$$

имеется четыре операции умножения и три операции сложения.

$$T(N) = 4T(N/2) + \Theta(N). \tag{1}$$

- При умножении по алгоритму Карацубы

$$N_1 \times N_2 = T^2 x_1 x_2 + T((x_1 + y_1)(x_2 + y_2) - x_1 x_2 - y_1 y_2) + y_1 y_2$$

имеется три операции умножения и шесть операций сложения.

$$T(N) = 3T(N/2) + \Theta(N). \tag{2}$$

Умножение матриц

- Классический алгоритм умножения квадратных матриц размером $N \times N$ требует N^2 раз исполнить скалярное произведение $i-$й строки на $j-$й столбец.
- Скалярное произведение векторов размером N имеет сложность $\Theta(N)$.
- Эта операция исполняется $N \times N$ раз, следовательно, сложность классического произведения матриц $\Theta(N^3)$.

Умножение матриц: разделяй и властвуй

- Для простоты предположим, что матрица имет размер 2^n.
- Тогда каждую матрицу можно разбить на четыре подматрицы.

$$A = \begin{pmatrix} A_{11} A_{12} \\ A_{21} A_{22} \end{pmatrix} \quad B = \begin{pmatrix} B_{11} B_{12} \\ B_{21} B_{22} \end{pmatrix} \quad C = \begin{pmatrix} C_{11} C_{12} \\ C_{21} C_{22} \end{pmatrix}$$

- Уравнение $C = A \cdot B$ теперь выглядит так:

$$\begin{pmatrix} C_{11} C_{12} \\ C_{21} C_{22} \end{pmatrix} = \begin{pmatrix} A_{11} A_{12} \\ A_{21} A_{22} \end{pmatrix} \cdot \begin{pmatrix} B_{11} B_{12} \\ B_{21} B_{22} \end{pmatrix} \tag{3}$$

Умножение матриц: разделяй и властвуй

Уравнение (3) соответствует

$$
\begin{aligned}
C_{11} &= A_{11} \cdot B_{11} + A_{12} \cdot B_{21}, \\
C_{12} &= A_{11} \cdot B_{12} + A_{12} \cdot B_{22}, \\
C_{21} &= A_{21} \cdot B_{11} + A_{22} \cdot B_{21}, \\
C_{22} &= A_{21} \cdot B_{12} + A_{22} \cdot B_{22}.
\end{aligned}
$$

Умножения половинных матриц можно использовать для алгоритма «разделяй и властвуй».

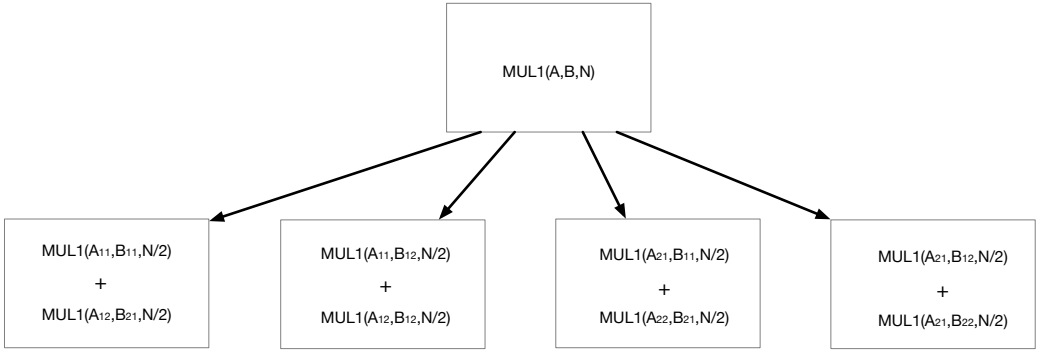
1: **procedure** $MUL1(A, B, N)$
2: **if** $N = 1$ **then**
3: **return** $C = \{c_{11} = a_{11} * b_{11}\}$
4: **end if**
5: $C_{11} = MUL1(A_{11}, B_{11}, N/2) + MUL1(A_{12}, B_{21}, N/2)$
6: $C_{11} = MUL1(A_{11}, B_{12}, N/2) + MUL1(A_{12}, B_{22}, N/2)$
7: $C_{21} = MUL1(A_{21}, B_{11}, N/2) + MUL1(A_{22}, B_{21}, N/2)$
8: $C_{22} = MUL1(A_{21}, B_{12}, N/2) + MUL1(A_{22}, B_{22}, N/2)$
9: **return** C
10: **end procedure**

Оценка времени работы алгоритма

- Количество вызовов $MUL1$ равно 8.
- Размер каждой задачи при вызове равен $N/2$.
- Каждая операция сложения матриц требует $\Theta(N^2/4)$.
- Операций сложения матриц — 4.
- Общая сложность рекуррентного алгоритма

$$T(N) = 8T(N/2) + \Theta(N^2). \tag{4}$$

- Обратим внимание на то, что при учёте операций сложения мы могли убрать множитель, так как $\Theta(N^2/4) = \Theta(N^2)$.
- Однако, мы не можем убрать множитель 8 перед $8T(N/2)$.
- Этот множитель определяет количество рекурсивных вызовов функции, что влияет и на суммарную сложность всей рекурсивной операции.


MUL1(A,B,N)
MUL1(A11,B11,N/2)
+
MUL1(A12,B21,N/2)
MUL1(A11,B12,N/2)
+
MUL1(A12,B12,N/2)
MUL1(A21,B11,N/2)
+
MUL1(A22,B21,N/2)
MUL1(A21,B12,N/2)
+
MUL1(A21,B22,N/2)

Сложность алгоритма

- Как определить сложность алгоритма (4)?
- Давайте рассмотрим ещё один алгоритм.
- Мы уменьшим количество операций рекурсии на 1, увеличив количество операций сложения подматриц.

Алгоритм Штрассена

1. Разобьём матрицы A и B на подматрицы.
2. Создадим 10 новых матриц размером $(N/2 \times N/2)$.
3. Рекурсивно вычислим семь произведений вновь созданных подматриц.
4. Сложим результат в итоговую матрицу C.

2-й шаг алгоритма Штрассена

Нужные 10 матриц 2-го шага алгоритма вычисляются так:

$$
\begin{aligned}
S_1 &= B_{12} - B_{22}, \\
S_2 &= A_{11} + A_{12}, \\
S_3 &= A_{21} + A_{22}, \\
S_4 &= B_{21} - B_{11}, \\
S_5 &= A_{11} + A_{22}, \\
S_6 &= B_{11} + B_{22}, \\
S_7 &= A_{12} - A_{22}, \\
S_8 &= B_{21} + B_{22}, \\
S_9 &= A_{11} - A_{21}, \\
S_{10} &= B_{11} + B_{12}.
\end{aligned}
$$

3-й шаг алгоритма Штрассена

3-й шаг состоит в рекурсивном вычислении следующих семи подматриц:

$$
\begin{aligned}
P_1 &= A_{11} \cdot S_1, \\
P_2 &= S_2 \cdot B_{22}, \\
P_3 &= S_3 \cdot B_{11}, \\
P_4 &= A_{22} \cdot S_4, \\
P_5 &= S_5 \cdot S_6, \\
P_6 &= S_7 \cdot S_8, \\
P_7 &= S_9 \cdot S_{10}.
\end{aligned}
$$

4-й шаг алгоритма Штрассена

4-й шаг даёт подматрицы матрицы C, собираемые из матриц P_i.

$$\begin{aligned} C_{11} &= P_5 + P_4 - P_2 + P_6 \\ C_{12} &= P_1 + P_2 \\ C_{21} &= P_3 + P_4 \\ C_{22} &= P_5 + P_1 - P_3 - P_7. \end{aligned}$$

Алгоритм Штрассена: оценка сложности

1. Разобъём матрицы A и B на подматрицы — $\Theta(1)$.
2. Создадим 10 новых матриц размером $(N/2 \times N/2)$ — $\Theta(N^2)$.
3. Рекурсивно вычислим семь произведений вновь созданных подматриц — $7T(N/2)$.
4. Сложим результат в итоговую матрицу C — $\Theta(N^2)$.

Итого:

$$T(N) = 7T(N/2) + \Theta(N^2). \tag{5}$$

Как определять сложность таких рекуррентных выражений?

Наши рекуррентные соотношения:

$$
\begin{aligned}
T(N) = &\quad 4T(N/2) + \Theta(N) && (1)\\
T(N) = &\quad 3T(N/2) + \Theta(N) && (2)\\
T(N) = &\quad 8T(N/2) + \Theta(N^2) && (4)\\
T(N) = &\quad 7T(N/2) + \Theta(N^2) && (5).
\end{aligned}
$$

Как определять рекуррентные соотношения

- Угадать и подставить.

$$T(N) = 2T(N/2) + N.$$

Предположим решение в виде $O(N \log N)$. Это означает, что существует константа $c > 0$ такая, что $T(N) \leqslant cN \log N$.
Тогда подставим:

$$\begin{aligned} T(N) &\leqslant 2(cN/2 \log(N/2)) + N \leqslant \\ &\leqslant cN \log(N/2) + N = \\ &= cN \log N - cN \log 2 + N = \\ &= cN \log N - cN + N \leqslant cN \log N. \end{aligned}$$

Это — индуктивный переход.
Что взять за базу индукции?

Мастер-теорема о рекурсии

- Мастер-теорема отвечает на вопрос: как оценить сложность рекуррентных выражений «разделяй и властвуй».
- Рекуррентная задача должна отвечать следующим ограничениям:
  - Задача размером N разбивается на $a \geqslant 1$ подзадач.
  - Каждая из подзадач в $b > 1$ раз проще основной.
  - Все подзадачи решаются тем же методом, что и основная задача за исключением, может быть, граничных данных.
  - После решение подзадач производится *консолидация* результатов, требующая `f(N)`.

Мастер-теорема о рекурсии

Theorem (Мастер-теорема)

Для чисел $a \geqslant 1$, $b > 1$, *функции* $f(N)$ *и рекуррентного соотношения* $T(N)$, *определённого на множестве чисел* $N \in \mathbb{Z}_0$ *как*

$$T(N) = aT(N/b) + f(N),$$

$T(N)$ *имеет следующие асимптотические границы:*

1. *Если* $\exists \varepsilon > 0 : f(N) = O(N^{\log_b a - \varepsilon})$, *то* $T(N) = \Theta(N^{\log_b a})$;
2. *Если* $f(N) = \Theta(N^{\log_b a})$, *то* $T(N) = \Theta(N^{\log_b a} \cdot \log N)$;
3. *Если* $\exists \varepsilon > 0 : f(N) = \Omega(N^{\log_b a + \varepsilon})$, *причём* $\exists c < 1 : af(N/b) \leqslant cf(N)$, *то* $T(N) = \Theta(f(N))$.

Мастер-теорема: суть и условия

1. Если $\exists \varepsilon > 0 : f(N) = O(N^{\log_b a - \varepsilon})$, то $T(N) = \Theta(N^{\log_b a})$;
Функция $N^{\log_b a}$ должна быть *полиномиально больше*, чем $f(N)$.

3. Если $\exists \varepsilon > 0 : f(N) = \Omega(N^{\log_b a + \varepsilon})$, причём $\exists c < 1 : af(N/b) \leqslant cf(N)$, то $T(N) = \Theta(f(n))$.
Функция $f(N)$ должна быть полиномиально больше $N^{\log_b a}$ и обладать условием *регулярности* $af(N/b) \leqslant cf(N)$.
Если $f(N)$ меньше функции $N^{\log_b a}$, но неполиномиально или $f(N)$ больше функции $N^{\log_b a}$, но неполиномиально, то мастер-теорема неприменима.

Мастер-теорема: применение

Рассмотрим

$$T(N) = 4T(N/2) + \Theta(N).$$

$$a = 4, b = 2, f(N) = N^1.$$

$$N^{\log_b a} = N^{\log_2 4} = \Theta(N^2).$$

Так как $f(N) = O(N^{\log_2 4 - \varepsilon})$, где $\varepsilon = 1$, то случай теоремы — 1.

$$T(N) = \Theta(N^2).$$

Мастер-теорема: применение

Рассмотрим алгоритм Карацубы.

$$T(N) = 3T(N/2) + \Theta(N).$$

$$a = 3, b = 2, f(N) = N^1.$$

$$N^{\log_b a} = N^{\log_2 3} = \Theta(N^{\log_2 3}).$$

Так как $f(N) = O(N^{\log_2 3 - \varepsilon})$, где $\varepsilon = \log_2 3 - 1$, то случай теоремы — 1.

$$T(N) = \Theta(N^{\log_2 3}) \approx \Theta(N^{1.58}).$$

Мастер-теорема: применение

Рассмотрим алгоритм Штрассена:

$$T(N) = 7T(N/2) + \Theta(N^2).$$

$$a = 7, b = 2, f(N) = N^2.$$

$$N^{\log_b a} = N^{\log_2 7} = \Theta(N^{\log_2 7}).$$

Так как $f(N) = O(N^{\log_2 7 - \varepsilon})$, где $\varepsilon = \log_2 7 - 2$, то случай теоремы — 1.

$$T(N) = \Theta(N^{\log_2 7}) \approx \Theta(N^{2.81}).$$

Мастер-теорема: применение

Рассмотрим

$$T(N) = T(3n/4) + 1.$$

Здесь $a = 1, b = 4/3, f(N) = 1, N^{\log_b a} = 1$.
Применяется случай 2, так как $f(N) = \Theta(1)$, откуда

$$T(N) = \Theta(\log N).$$

Мастер-теорема: применение

Рассмотрим

$$T(N) = 3T(N/4) + N \log N.$$

Здесь $a = 3, b = 4, f(N) = N \log N, N^{\log_b a} = N^{\log_4 3} = O(N^{0.793})$.
$f(N) = \Omega(n^{\log_4 3 + \varepsilon})$, где $\varepsilon \approx 0.2$.
Покажем, что для функции $f(N)$ применяется условие регулярности.
При $N > N_0$: $af(N/b) = 3(N/4)\log(N/4) \leqslant (3/4)N \log N = cf(N)$ для $c = 3/4$.
Применяется случай 3:

$$T(N) = \Theta(N \log N).$$

Мастер-теорема: применение

Рассмотрим

$$T(N) = 2T(N/2) + N \log N.$$

Здесь $a = 2, b = 2, f(N) = N \log N, N^{\log_b a} = N$.
Хотя функция $f(N) = N \log N$ асимптотически больше, чем $N^{\log_b a} = N$, но она асимптотически больше *не полиномиально*.
Отношение

$$\frac{f(N)}{n^{\log_b a}} = \frac{N \log N}{N} = \log N$$

асимптотически меньше N^{ε} для любого $\varepsilon > 0$.