Тема 1. Алгоритмы и их сложности

1.4. Рекуррентные соотношения

Анализ рекурсивных алгоритмов обычно приводит к соотношениям, которые называются *рекуррентными*. В них время выполнения алгоритма выражается через время выполнения того же алгоритма на входах меньшего размера. В качестве примера рассмотрим рекурсивную функцию *FACT* вычисления $n!$ для целых чисел от 0 до n включительно, представленную алгоритмом 1.4. Очевидно, что размером входа является значение n.

function $FACT(n)$
  if $n \le 1$
    then $FACT \leftarrow 1$
    else $FACT \leftarrow n \times FACT(n-1)$
return

Алгоритм 1.4. Рекурсивное вычисление факториала

Проверка условия в операторе **if** и выполнение присваивания в его **then**-части требуют времени порядка $O(1)$ (обозначенного константой a_1). Время выполнения оператора в **else**-части складывается из времени работы функции $FACT(n-1)$ и постоянной составляющей (обозначенной константой a_2) порядка $O(1)$, которая включает в себя проверку условия, вызов функции, умножение и присваивание. Следовательно, время работы алгоритма описывается рекуррентным соотношением

$$T(n) = \begin{cases} a_1, & \text{если } n \le 1, \\ T(n-1) + a_2, & \text{если } n > 1. \end{cases} \qquad (1.1)$$

Рассмотрим три способа решения рекуррентных соотношений: метод подстановки, метод итераций и общий метод [9].

Метод подстановки. Идея метода заключается в нахождении такой функции $f(n)$, чтобы для всех n выполнялось неравенство $T(n) \le f(n)$. Тогда функция $f(n)$ будет верхней границей скорости роста $T(n)$, т. е. $T(n) = O(f(n))$. Другими словами, следует отгадать ответ и затем доказать его по индукции.

В качестве примера определим верхнюю оценку рекуррентного соотношения (1.1). Вероятнее всего, $T(n) = O(n)$, т. е. $T(n) \le cn$ для некоторой константы $c > 0$. При $n = 0$ это условие не выполняется, поскольку cn равно нулю, независимо от значения c. Применим функцию $f(n) = cn + d$, т. е. $T(n) \le cn + d$, где $d > 0$. Теперь при $n \le 1$ условие выполняется, если $d \ge a_1$ и $c > 0$. Пусть $n > 1$. Должно выполняться условие

$$T(n) = T(n-1) + a_2 \le cn + d.$$

Пусть эта оценка верна для $n - 1$, т. е. $T(n-1) \le c(n-1) + d$. Подставив ее в рекуррентное соотношение, получим

$$T(n) \le c(n-1) + d + a_2 = cn - c + d + a_2 \le cn + d.$$

Данное неравенство выполняется, если $c \ge a_2$. Таким образом, оценка $T(n) \le cn + d$ будет справедлива, если $d \ge a_1$ и $c \ge a_2$. Если положить $d = a_1$ и $c = a_2$, т. е. определить константы пропорциональности функции $f(n) = cn + d$, то для всех $n \ge 0$ выполняется неравенство $T(n) \le a_2 n + a_1$. Следовательно, $T(n) = O(n)$.

Рассмотрим более сложное рекуррентное соотношение

$$T(n)=\begin{cases}a_1, & \text{если } n=1,\\ 2T(n/2)+a_2 n, & \text{если } n>1,\end{cases} \qquad (1.2)$$

где a_1 и a_2 – некоторые положительные константы. Предположим, что $T(n) \le cn \log n$. При $n = 1$ отношение не выполняется, так как в этом случае выражение $cn \log n$ всегда равно нулю. Применим функцию $T(n) \le cn \log n + d$. Теперь при $n = 1$ эта оценка справедлива, если $d \ge a_1$. Пусть $n > 1$. Должно выполняться условие $T(n) = 2\,T(n/2) + a_2\,n \le cn \log n + d$. Пусть эта оценка верна для $n/2$, т. е. $T(n/2) \le c\,(n/2) \log (n/2) + d$. Подставив ее в рекуррентное соотношение, получим

$$\begin{aligned} T(n) &\le 2(c(n/2)\log(n/2)+d)+a_2 n = \\ &= cn\log n - cn\log 2 + 2d + a_2 n = \\ &= cn\log n - cn + 2d + a_2 n \le cn\log n + d. \end{aligned}$$

Данное неравенство выполняется, если $c \ge a_2 + d$. Если положить $d = a_1$ и $c = a_1 + a_2$, то для всех $n > 0$ выполняется неравенство $T(n) \le (a_1 + a_2)\,n \log n + a_1$. Следовательно, $T(n) = O\,(n \log n)$.

В ряде случаев сложные рекуррентные соотношения можно упростить, если применить известный в математике прием – замену переменных. В качестве примера рассмотрим соотношение

$$T(n)=\begin{cases}a_1, & \text{если } n=1,\\ 2T(\sqrt{n})+a_2 \log n, & \text{если } n>1.\end{cases}$$

Выполним замену переменной m на $\log n$, т. е. $m = \log n$. В результате получим

$$T(2^m) = 2T(2^{m/2}) + a_2 m\,.$$

Замена $T(2^m)$ на $S\,(m)$ приводит к соотношению

$$S(m) = 2\,S(m/2) + a_2 m\,,$$

которое соответствует рекуррентной части соотношения (1.2) и, следовательно, имеет решение $S(m) = O(m \log m)$. Возвращаясь к обозначению $T(n)$ вместо $S(m)$, получим

$$T(n) = T(2^m) = S(m) = O(m \log m) = O(\log n \log \log n).$$

Метод итераций. Заключается в итерации рекуррентного соотношения, т. е. подстановки его в самого себя до тех пор, пока из правой части не исключатся рекурсивные обращения.

Рассмотрим соотношение (1.1). Подставляя его самого в себя, получим

$$T(n) = T(n-1) + a_2 = T(n-2) + 2a_2 = T(n-3) + 3a_2.$$

Продолжая этот процесс, в общем случае для некоторого $i < n$ получаем $T(n) = T(n-i) + i\,a_2$. Положив в последнем выражении $i = n-1$, окончательно получаем

$$T(n) = T(1) + a_2(n-1) = a_2(n-1) + a_1 = O(n).$$

Рассмотрим этот метод для соотношения (1.2). Подставляя его самого в себя, получим

$$T(n) = 2T(n/2) + a_2 n = 2(2T(n/4) + a_2 n/2) + a_2 n$$
$$= 4T(n/4) + 2a_2 n = 8T(n/8) + 3a_2 n.$$

Продолжая этот процесс, в общем случае для некоторого $i < n$, получаем

$$T(n) = 2^i T(n/2^i) + i\,a_2 n.$$

Пусть $n = 2^k$, т. е. является степенью числа 2. Тогда при $i = k$ процесс подстановок завершается подстановкой $T(1)$

$$T(n) = 2^k T(1) + k\,a_2 n.$$

Поскольку $k = \log n$ и $T(1) = a_1$, окончательно получаем

$$T(n) = a_2 n \log n + a_1 n = O(n \log n).$$

Общий метод. Этот метод позволяет получить асимптотические оценки для рекуррентных соотношений вида

$$T(1) = d,$$
$$T(n) = aT(n/b) + f(n), \qquad (1.3)$$

где $a \geq 1$, $b > 1$ и $d > 0$ – некоторые константы, а $f(n)$ – положительная функция. Данное соотношение обычно получается, когда алгоритм разбивает исходную задачу размера n на a подзадач размера n/b. Подзадачи решаются рекурсивно каждая за время $T(n/b)$ и результаты объединяются за время $f(n)$. Такой подход к проектированию эффективных алгоритмов называется *методом декомпозиции* (или методом разбиения).

Пусть n является натуральной степенью числа b, т. е. $n = b^k$. Применим метод итераций, выполняя последовательную подстановку рекуррентного соотношения самого в себя

$$T(n) = f(n) + aT(n/b) = f(n) + a\,f(n/b) + a^2T(n/b^2) =$$
$$= f(n) + a\,f(n/b) + a^2 f(n/b^2) + \ldots + a^{k-1} f(n/b^{k-1}) + a^k T(1),$$

где $k = \log_b n$. Поскольку $a^k = a^{\log_b n} = n^{\log_b a}$, получаем

$$T(n) = d\,n^{\log_b a} + \sum_{j=0}^{k-1} a^j f(n/b^j). \qquad (1.4)$$

Таким образом, рекуррентное соотношение можно представить в виде суммы, вычисление которой позволит определить временную сложность алгоритма. Например, в соотношении (1.2) имеем $d = a_1$, $a = b = 2$, $f(n) = a_2 n$, $k = \log n$, поэтому

$$T(n) = a_1 n + \sum_{j=0}^{k-1} 2^j a_2 n / 2^j = a_1 n + a_2 n \sum_{j=0}^{k-1} 1 =$$
$$= a_1 n + a_2 n k = a_2 n \log n + a_1 n = O(n \log n).$$

Часто преобразование рекуррентного соотношения в сумму приводит к довольно сложным выражениям. Если требуется определить только асимптотические оценки, общий метод решения рекуррентных соотношений позволяет получить их более простым способом для достаточно широкого класса функций $f(n)$.

Рассмотрим влияние каждого выражения в формуле (1.4) на оценку функции $T(n)$. Асимптотическая оценка первого выражения очевидна и составляет $O(n^{\log_b a})$. Рассмотрим функцию

$$g(n) = \sum_{j=0}^{k-1} a^j f(n / b^j).$$

С точки зрения анализа алгоритмов наибольший интерес представляют ряды, образуемые функцией $f(n)$ вида $c\,n^{\alpha}$, где $c > 0$ и $\alpha \geq 0$ – некоторые константы. Тогда ряд представляет собой геометрическую прогрессию

$$g(n) = cn^{\alpha} + ac(n/b)^{\alpha} + a^2 c(n/b^2)^{\alpha} + \ldots + a^{k-1} c(n/b^{k-1})^{\alpha}$$

с числом членов $k = \log_b n$ и знаменателем $q = a / b^{\alpha}$. Возможны следующие ситуации:

1) если $a > b^{\alpha}$, т. е. $\alpha < \log_b a$ или $\alpha = \log_b a - \varepsilon$ для некоторой константы $\varepsilon > 0$, то знаменатель $q > 1$. Для такой возрастающей прогрессии сумма асимптотически равна последнему члену, т. е. $g(n) = O(a^{k-1}) = O(a^{\log_b n - 1}) = O(n^{\log_b a} / a)$. Таким образом,

$$T(n) = O(n^{\log_b a}) + O(n^{\log_b a} / a) = O(n^{\log_b a});$$

2) если $a = b^{\alpha}$, т. е. $\alpha = \log_b a$, то знаменатель $q = 1$ (все члены прогрессии равны). Число членов есть $\log_b n$, поэтому сумма равна $cn^{\log_b a} \log_b n$, т. е. $g(n) = O(n^{\log_b a} \log n)$. Следовательно,

$$T(n) = O(n^{\log_b a}) + O(n^{\log_b a} \log n) = O(n^{\log_b a} \log n);$$

3) если $a < b^{\alpha}$, т. е. $\alpha > \log_b a$ или $\alpha = \log_b a + \varepsilon$ для некоторой константы $\varepsilon > 0$, то знаменатель меньше единицы. Для такой убывающей прогрессии сумма асимптотически равна первому члену, т. е. $g(n) = O(n^{\alpha})$. В этом случае, поскольку $\alpha > \log_b a$,

$$T(n) = O(n^{\log_b a}) + O(n^{\alpha}) = O(n^{\alpha}) = O(f(n)).$$

Таким образом, метод асимптотической оценки рекуррентных соотношений вида (1.3) для функций $f(n)$ вида cn^{α} формулируется следующим образом:

$$T(n) = \begin{cases} O(n^{\log_b a}), & \text{если } a > b^{\alpha}, \\ O(n^{\log_b a} \log n), & \text{если } a = b^{\alpha}, \\ O(f(n)), & \text{если } a < b^{\alpha}. \end{cases} \qquad (1.5)$$

Например, в соотношении (1.2) имеем $a = b = 2$, $f(n) = a_2 n$, т. е. $\alpha = 1$. Поскольку $2 = 2^1$, т. е. подходит второй случай, получаем $T(n) = O(n \log n)$.

Более общий метод решения рекуррентных соотношений вида (1.3) формулируется следующим образом [9]:

1) если $f(n) = O(n^{\log_b a - \varepsilon})$ для некоторой величины ε > 0, то $T(n) = \Theta(n^{\log_b a})$;

2) если $f(n) = \Theta(n^{\log_b a})$, то $T(n) = \Theta(n^{\log_b a} \log n)$;

3) если $f(n) = \Omega(n^{\log_b a + \varepsilon})$ для некоторой величины ε > 0 и если $af(n/b) \le cf(n)$ для некоторой константы $c < 1$ и достаточно больших n, то $T(n) = \Theta(f(n))$.

Следует отметить, что в первом случае недостаточно, чтобы функция $f(n)$ была асимптотически меньше, чем $n^{\log_b a}$, а необходим некоторый зазор размера n^{ε} для некоторого ε > 0. Аналогично в третьем случае функция $f(n)$ должна быть асимптотически больше $n^{\log_b a}$ с зазором n^{ε}, кроме того, должно удовлетворяться условие $af(n/b) \le cf(n)$.

В качестве примера применения общего метода рассмотрим рекуррентное соотношение $T(n) = 3T(n/4) + n \log n$ с начальным значением $T(1) = 1$. Имеем $a = 3$, $b = 4$, $f(n) = n \log n$; при этом $n^{\log_b a} = n^{\log_4 3} < n^{0,8}$. Функция $f(n)$ асимптотически больше, чем $n^{\log_b a}$, так как отношение $f(n)/n^{\log_b a} = (n \log n)/n^{0,8} = n^{0,2} \log n$ оценивается снизу величиной $n^{0,2}$, т. е. имеется зазор $n^{0,2}$. Проверим условие $af(n/b) \le cf(n)$: $3(n/4) \log (n/4) \le cn \log n$. Условие выполняется для $c = 3/4$. Таким образом, согласно третьему условию, $T(n) = \Theta(n \log n)$.

Существуют и другие типы рекуррентных соотношений, для которых не подходит рассмотренный общий метод решения, например, $T(n) = 2T(n/2) + n \log n$ с начальным значением $T(1) = 1$. Попробуем применить общий метод. Имеем $a = b = 2$, $f(n) = n \log n$, $n^{\log_b a} = n$. Очевидно, что $f(n) = n \log n$ асимптотически больше, чем $n^{\log_b a} = n$. Однако зазор недостаточен, поскольку отношение

$f(n)/n^{\log_b a} = (n\log n)/n = \log n$ не оценивается снизу величиной n^{ε} ни для какого $\varepsilon > 0$. Таким образом, применить общий метод не удается. Тем не менее это соотношение легко можно решить по формуле (1.4)

$$T(n) = n + \sum_{j=0}^{k-1} 2^j (n/2^j)\log(n/2^j) = n + n\sum_{j=0}^{k-1}(\log n - j) =$$
$$= n + n\log n(\log n + 1)/2 = O(n\log^2 n).$$

В рассмотренных выше методах решения рекуррентных соотношений предполагалось, что n является целой степенью числа b, т. е. $n = b^k$. Для произвольных значений n в рекуррентных соотношениях должны рассматриваться только целые части, поскольку функция $T(n)$ определена только для целых n. Например, рекуррентная часть соотношения (1.2) должна записываться в виде $T(n) = T(\lceil n/2 \rceil) + T(\lfloor n/2 \rfloor) + a_2 n$, а в соотношении (1.3) под выражением n/b должно пониматься либо $\lceil n/b \rceil$, либо $\lfloor n/b \rfloor$. Во многих случаях оценки, полученные в предположении, что $n = b^k$, распространяются и на произвольные значения n. Это объясняется тем, что задачу размера n можно вложить в задачу размера n', где n' – наименьшая степень числа b, большая или равная n, а затем решить рекуррентное соотношение для задачи размера n'. Однако следует быть внимательным к деталям, связанным с округлением до целого сверху или снизу, поскольку возможны случаи (пусть и достаточно редкие), когда оценки для $n = b^k$ могут отличаться от оценок для других значений n.