1. Математические основы анализа алгоритмов

1.1. Лучший, средний, худший случаи

В этом и следующих разделах основной целью будет получение точной оценки времени выполнения для некоторых конкретных алгоритмов.

Рассмотрим следующую задачу: требуется найти максимальный элемент в массиве из n чисел. Для решения можно предложить следующий алгоритм (процедура 1.1.1):

```
Процедура 1.1.1. Get Max Element
01:  Get Max Element Impl(A[1, n])
02:        result = A[1]
03:        for i = 1 to n do
04:              if A[i] > result then
05:                    result = A[i]
06:        return result
```

Подсчитаем число шагов выполнения этого алгоритма как можно точнее. Первый оператор выполняется один раз. Оператор цикла – n раз. Проверка условия выполняется при каждой итерации цикла, т. е. тоже n раз. Особого внимания заслуживает оператор $\{5\}$ изменения результата. Количество выполнений этого оператора зависит от данных исходной задачи и не выражается в явной форме. В большинстве алгоритмов встречаются операторы, количество выполнений которых зависит от исходных данных. В связи с этим при точной оценке сложности алгоритма принят такой подход, когда вместо всех возможных наборов входных данных проводится оценка сложности в трех случаях – *наилучшем*, *наихудшем* и *среднем*.

Чтобы оценить количество выполнений оператора изменения результата, используем вычислительный эксперимент. Создадим метод, который генерирует все возможные перестановки заданного числового множества $\{1,2,\dots,n\}$. Затем для каждой из них будем запускать поиск максимума и считать число выполнений оператора $\{5\}$. Для множества из десяти элементов статистика по выполнению оператора присваивания имеет следующий вид:

Количество выполнений оператора	**Количество перестановок**	**Вероятность**
0	362880	0,1
1	1026576	0,282896825
2	1172700	0,323164683
3	723680	0,199426808
4	269325	0,07421875
5	63273	0,017436343
6	9450	0,002604167
7	870	0,000239749
8	45	1,24008e–05
9	1	2,75573e–07

Вероятность в третьей колонке таблицы посчитана как отношение значения второй колонки к сумме всех значений во второй колонке. Зная вероятно-

сти, можно построить график распределения случайной величины, соответствующей указанному количеству выполнений оператора {5} (рис. 1.1).

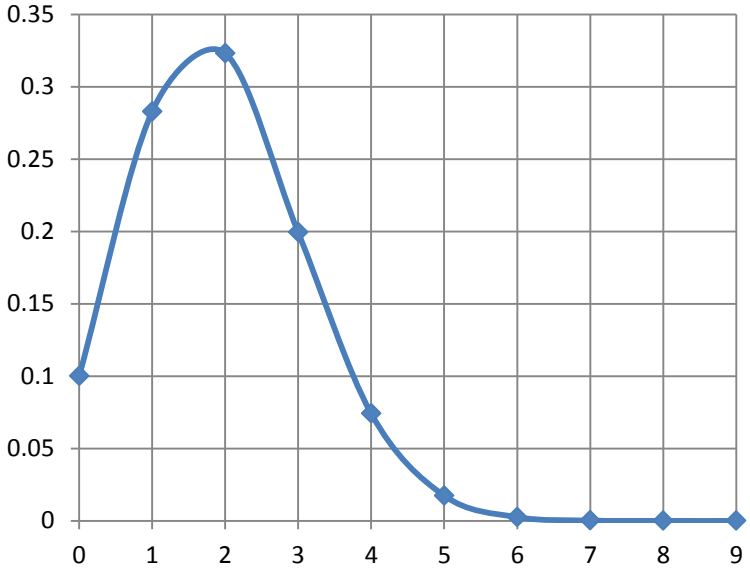

Рис. 1.1. График распределения случайной величины

Из приведенного графика можно видеть наихудший и наилучший случаи для оператора {5}. В наихудшем случае, который работает для единственной перестановки, количество выполнений оператора {5} равно 9. Лучший случай – количество выполнений оператора равно 0. Оценка для среднего случая находится по следующему правилу: для каждой строки таблицы значение из первого столбца умножается на значение из третьего столбца, после чего результат складывается. Получим приблизительно 1,93, т. е. в среднем оператор {5} выполняется два раза.

Вычислительный эксперимент обычно сложно провести для больших n. Поэтому анализ сложности алгоритма проводят аналитически. Рассмотрим все случаи для алгоритма поиска максимума. Наилучший ($B(n)$) – это случай, когда максимальный элемент является первым элементом в массиве. Тогда количество выполнений оператора {5} будет равно нулю. Наихудший случай ($W(n)$) соответствует ситуации, когда исходный массив отсортирован по возрастанию. Тогда оператор {5} будет выполняться при каждой итерации цикла, кроме последней, т. е. $n - 1$ раз.

При среднем случае ($A(n)$) для вывода оценки количества выполнений оператора {5} сделаем следующее предположение о природе элементов исходного массива: будем считать, что все они различны. Без этого предположения вывод оценки становится затруднительным. При первой итерации цикла оператор {5} не выполнится (это особенность реализации!). При второй итерации вероятность выполнения оператора равна $^1/_2$, так как есть возможность того, что добавляемый к множеству чисел (из одного элемента) новый элемент окажется среди них наибольшим. То есть можно сказать, что при второй итерации цикла оператор {5} «выполнится» $^1/_2$ раз. При третьей итерации цикла вероятность выполнения оператора {5} равна $^1/_3$: числовое множество состоит из двух эле-

ментов; добавляем третий, и вероятность того, что он наибольший равняется $^1/_3$. Таким образом, число выполнений оператора {5} выражается формулой

$$S = 0 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n}.$$

Замкнутое выражение для S получаем, воспользовавшись формулой для частичной суммы гармонического ряда:

$$H_n = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} = \ln n + \gamma + \frac{1}{2n} + o\left(\frac{1}{n}\right),$$

где $\gamma = 0{,}57721566\ldots$ – *постоянная Эйлера*.

Тогда $S = H_n - 1 \approx \ln n + \frac{1}{2n} - 0{,}42 \approx \ln n$.

Данная оценка является весьма точной (с точностью до аддитивной константы). Таким образом, для количества выполнений оператора {5} получены следующие результаты: $B(n) = 0$, $W(n) = n - 1$, $A(n) = \ln n$.

Теперь легко можно подсчитать общее число выполнений операторов в анализируемом алгоритме. Требуется рассмотрение трех случаев. Кроме этого, иногда для операторов вводят так называемые *стоимости выполнения*, характеризующие время выполнения отдельного оператора. Пусть, например, стоимости выполнения операторов процедуры (1.1.1) равны c_2, c_3, c_4, c_5. Тогда время выполнения алгоритма следующее:

- $B(n) = c_2 + nc_3 + nc_4 + 0 \cdot c_5 = c_2 + n(c_3 + c_4)$;
- $W(n) = c_2 + n(c_3 + c_4) + (n-1)c_5$;
- $A(n) = c_2 + nc_3 + nc_4 + \ln n \cdot c_5$.

Можно видеть, что при выполнении общей задачи точного анализа алгоритмов активно используется математический аппарат, в частности аппарат теории вероятностей.

1.2. Рекуррентные соотношения

Пусть задана последовательность $\{a_n\} = a_1, a_2, \ldots, a_n \ldots$ Тогда формулу для вычисления ее элементов будем называть *рекуррентной*, если при вычислении каждого члена последовательности необходимо знать предыдущие[*].

Последовательность $\{a_n\}$ называется *рекуррентной порядка* k ($k \in \mathbb{N}$), если существует формула $a_{n+k} = f(a_{n+k-1}, a_{n+k-2}, \ldots, a_n)$, с помощью которой элемент a_{n+k} вычисляется по k предыдущим элементам $a_{n+k-1}, a_{n+k-2}, \ldots, a_n$. Данная формула называется *рекуррентным соотношением* (рекуррентным уравнением) порядка k. Обратим внимание на то, что первые k элементов последовательности $\{a_n\}$ должны быть известны. Примером рекуррентного соотношения может служить последовательность Фибоначчи:

[*] Рекуррентная (от фр. *récurrente*) – возвращающаяся к началу.

$$\begin{cases} f_0 = 0, f_1 = 1, \\ f_{n+2} = f_{n+1} + f_n, n \geq 0 \end{cases} \text{ или } \begin{cases} f_0 = 0, f_1 = 1, \\ f_n = f_{n-1} + f_{n-2}, n \geq 2. \end{cases}$$

Более формально рекуррентное соотношение определяется как *рекурсивная функция* с целочисленными значениями.

Определение 1.1. Рекуррентное соотношение вида

$$a_{n+k} = \alpha_1 \cdot a_{n+k-1} + \alpha_2 \cdot a_{n+k-2} + \cdots + \alpha_k \cdot a_n, \tag{1.1}$$

где $n \geq k > 0$, называется *однородным линейным*.

Определение 1.2. Рекуррентное соотношение вида

$$a_{n+k} = \alpha_1 \cdot a_{n+k-1} + \alpha_2 \cdot a_{n+k-2} + \cdots + \alpha_k \cdot a_n + f(n), \tag{1.2}$$

где $n \geq k > 0$, называется *неоднородным линейным*.

Помимо вышеупомянутых уравнений (1.1) и (1.2), выделим отдельный класс уравнений, которые можно описать соотношением вида

$$\begin{cases} T(1) = c, \\ T(n) = a \cdot T(n/b) + f(n), n > 1, \end{cases} \tag{1.3}$$

где $a \geq 1$ и $b \geq 1$; c – константа; функция $f(n)$ – асимптотически положительная; n/b может рассматриваться как $\lfloor n/b \rfloor$, так и $\lceil n/b \rceil$.

В более общем виде уравнение (1.3) может быть приведено к виду

$$T(n) = \sum_{i=1}^{k} a_i \cdot T\left(\left\lfloor \frac{n}{b_i} \right\rfloor\right) + f(n), n > 1, \tag{1.4}$$

где $k \geq 1$; все коэффициенты $a_i \geq 0$ и $\sum a_i \geq 1$; все $b_i > 1$; функция $f(n)$ – ограниченная, положительная и неубывающая; для любой константы $c > 1$ существуют константы n_0 и $d > 0$, такие, что для всех $n > n_0$ справедливо соотношение $f(n/c) \geq d \cdot f(n)$.

Методы решения рекуррентных уравнений

Очень часто оказывается полезным получение замкнутого вида рекуррентного соотношения, т. е. такого вида, в котором значение a_n выражается через номер n. Получение замкнутого вида рекуррентного соотношения имеет смысл лишь в том случае, если требуется оценить скорость роста функции $\{a_n\}$ либо быстро вычислить значение a_n для достаточно больших n. Существует множество методик, которые позволяют построить такой вид. Основные из них будут рассмотрены ниже.

Метод подстановки

Метод подстановки работает в том случае, если решаемое уравнение можно путем замены переменных свести к уравнению, для которого решение уже известно. Например, пусть требуется найти решение в замкнутом виде уравнения $T(n) = 2T\left(\sqrt{n}\right) + 1$. Будем считать, что $n = 2^m$. С учетом этого рекуррентное соотношение примет вид $T(2^m) = 2T\left(2^{m/2}\right) + 1$ или, после замены $T(2^m)$ на $S(m)$,

$$S(m) = 2S(m/2) + 1. \quad (1.5)$$

Структура уравнения (1.5) совпадает со структурой уравнения (1.3), для которого существует решение в замкнутом виде. Предположив, что решение уравнения (1.5) имеет вид $F(m)$, можно записать решение исходного уравнения как $T(n) = F(\log_2 n)$.

Метод подстановки очень хорошо подходит для решения рекуррентных уравнений, описывающих декомпозиционные методы. Так как функция $T(n)$ – не убывающая, то решение можно искать подбором. Решением будет такая функция $g(n)$, для которой верно неравенство

$$g(n) \geq a \cdot g(n/b) + f(n).$$

Из всех таких функций требуется выбрать функцию наименьшего порядка. Рассмотрим идею метода на следующем примере. Требуется решить уравнение $T(n) = 2T(\sqrt{n}) + 1$. Пусть $g(n) = cn^2$, где c – некоторая положительная константа, отличная от нуля. Тогда $2c(\sqrt{n})^2 + 1 = 2c \cdot n + 1 < cn^2$, следовательно, $T(n) = O(n^2)$. Попробуем улучшить решение. Рассмотрим функцию $g(n) = c \cdot \log_2 \log_2 n$. Тогда

$$2c \log_2 \log_2 \sqrt{n} \vee c \log_2 \log_2 n \Leftrightarrow \frac{1}{2^{2c-1}} (\log_2 n)^{2c} \vee (\log_2 n)^c$$
$$\Leftrightarrow (\log_2 n)^c \vee 2^{2c-1} \Rightarrow (\log_2 n)^c > 2^{2c-1}.$$

Причем полученное неравенство выполняется, начиная с $n \geq 16$. Следовательно, $O(\log_2 \log_2 n)$ не является решением. Пусть $g(n) = c \cdot \log_2 n$. Тогда $2c \log_2 \sqrt{n} + 1 = c \log_2 n + 1 = O(\log_2 n)$. Из последнего равенства можно заключить, что трудоемкость алгоритма, описываемого рассмотренным рекуррентным соотношением, по крайней мере $T(n) = O(\log_2 n)$.

Метод итераций

Идея метода итераций заключается в том, чтобы расписать исследуемое рекуррентное соотношение через множество других и затем просуммировать полученные слагаемые. Например, требуется решить рекуррентное соотношение вида

$$\begin{cases} T(0) = 1, \\ T(n) = T(n-1) + c, n \geq 1. \end{cases}$$

Последовательно выражая $T(n-k)$, где $k \leq n$, через предыдущие члены, можем записать выражение для $T(n)$:

$$T(n) = T(n-1) + c = (T(n-2) + c) + c = \cdots = T(n-k) + k \cdot c$$

Несложно видеть, что данный процесс завершится при $k = n$. Следовательно, решением рекуррентного соотношения будет функция $T(n) = 1 + n \cdot c$.

Характеристические многочлены

Рассмотрим общую схему решения однородного линейного рекуррентного соотношения, имеющего вид

$$a_{n+k} = \alpha_1 \cdot a_{n+k-1} + \alpha_2 \cdot a_{n+k-2} + \cdots + \alpha_k \cdot a_n. \quad (1.6)$$

Так как всякая рекуррентная последовательность k-го порядка однозначно определяется заданием k ее первых членов, то можно сформировать следующий многочлен:

$$P(x) = x^k - \alpha_1 \cdot x^{k-1} - \alpha_2 \cdot x^{k-2} - \cdots - \alpha_k. \quad (1.7)$$

Определение 1.3. Многочлен $P(x)$ называется *характеристическим многочленом* однородного линейного рекуррентного уравнения (1.6).

Пусть λ – корень характеристического многочлена $P(x)$. Тогда последовательность $a_0, a_1, a_2, \dots, a_n$, где $a_n = c \cdot \lambda^n$, c – произвольная константа, удовлетворяет рекуррентному соотношению (1.6). Действительно, после подстановки значений a_i в (1.5) получим

$$c \cdot \lambda^{n+k} - \alpha_1 \cdot c \cdot \lambda^{n+k-1} - \cdots - \alpha_k \cdot c \cdot \lambda^n = \\ = c \cdot \lambda^n(\lambda^k - \alpha_1 \cdot \lambda^{k-1} - \cdots - \alpha_k) = 0.$$

Если же λ – корень кратности r, то решением рекуррентного соотношения может быть любая из последовательностей $c_1 \cdot \lambda^n, \dots, c_r \cdot n^{r-1}\lambda^n$, где $c_1, \dots, c_r$ – произвольные константы. Заметим, что если существуют последовательности $\{a_n\}$ и $\{b_n\}$, удовлетворяющие уравнению (1.6), то и линейная комбинация $\{\alpha \cdot a_n + \beta \cdot b_n\}$ также является решением (1.6), где α, β – константы.

Теорема 1. Пусть $\lambda_1, \dots, \lambda_k$ – простые корни характеристического многочлена $P(x)$. Тогда общее решение рекуррентного соотношения (1.5) будет иметь вид

$$a_n = c_1 \cdot \lambda_1^n + \cdots + c_k \cdot \lambda_k^n, \quad (1.8)$$

для любого n; $c_1, \dots, c_k$ – константы.

Если же характеристический многочлен $P(x)$ имеет кратные корни $\lambda_1, \dots, \lambda_s (s \le k)$ с кратностями $r_1, \dots, r_s, \sum r_i = k$, то решение рекуррентного соотношения (1.9) можно записать в виде

$$a_n = \sum_{i=1}^{s} \lambda_i^n \cdot \sum_{j=0}^{r_i-1} c_{ij} \cdot n^j, \; n \ge 0, \quad (1.9)$$

для любого n; c_{ij} – константы.

Константы $c_1, \dots, c_k$ в случае, если уравнение (1.6) имеет решение вида (1.8), можно определить из следующей системы уравнений:

$$\begin{cases} c_1 + c_2 + \cdots + c_k = a_0, \\ c_1 \cdot \lambda_1 + c_2 \cdot \lambda_2 + \cdots + c_k \cdot \lambda_k = a_1 \\ \dots \\ c_1 \cdot \lambda_1^{k-1} + c_2 \cdot \lambda_2^{k-1} + \cdots + c_k \cdot \lambda_k^{k-1} = a_{k-1}. \end{cases} \quad (1.10)$$

Система (1.10) разрешима относительно $c_1, \dots, c_k$, так как ее определитель есть определитель Вандермонда, и он не равен нулю:

$$\begin{bmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ \lambda_1^{k-1} & \cdots & \lambda_k^{k-1} \end{bmatrix} = \prod_{i<j} (\lambda_i - \lambda_j) \neq 0.$$

Если же уравнение (1.6) имеет решения вида (1.9), то рассуждения для нахождения свободных коэффициентов будут аналогичны.

Рассмотрим пример. Пусть требуется найти замкнутый вид рекуррентного соотношения Фибоначчи $f_{n+2} = f_{n+1} + f_n$. Характеристический многочлен для данного уравнения будет иметь вид $\lambda^2 - \lambda - 1 = 0$. Решением данного уравнения являются $\lambda_1 = (1 + \sqrt{5})/2$ и $\lambda_2 = (1 - \sqrt{5})/2$. Так как корни λ_1 и λ_2 являются простыми, то общий вид решения можно записать в виде $c_1(1 + \sqrt{5})/2 + c_2(1 - \sqrt{5})/2$, где c_1, c_2 – произвольные константы. Для определения констант c_1 и c_2 составим систему уравнений согласно (1.10)

$$\begin{cases} c_1 + c_2 = 0, \\ c_1(1 + \sqrt{5})/2 + c_2(1 - \sqrt{5})/2 = 1, \end{cases}$$

из которой $c_1 = +1/\sqrt{5}$ и $c_2 = -1/\sqrt{5}$.

С учетом полученных значений замкнутый вид для последовательности Фибоначчи примет вид

$$f_n = \frac{1}{\sqrt{5}}(\varphi^n - (1 - \varphi)^n), \varphi = \frac{1 + \sqrt{5}}{2}$$

Далее рассмотрим общую схему решения неоднородного линейного рекуррентного соотношения, имеющего вид

$$a_{n+k} = \alpha_1 \cdot a_{n+k-1} + \alpha_2 \cdot a_{n+k-2} + \cdots + \alpha_k \cdot a_n + f(n), \qquad (1.11)$$

Общее решение уравнения (1.11) представляет собой сумму частного решения (обозначим его через a^p) и общего решения соответствующего ему однородного уравнения (обозначим его через a^g), т. е. $a_n = a^p + a^g$.

Общего метода нахождения частного решения a^p не существует. Частное решение находят исходя из структуры функции $f(n)$. Так, если $f(n) = \alpha \cdot u^n$, то $a^p = \beta \cdot u^n$. Если $f(n)$ – многочлен степени k, то и частное решение – многочлен степени k. В качестве примера исследуем структуру частного решения уравнения (1.11) для случая, когда $f(n) = c$, где c – некоторая константа. Если $\sum \alpha_i \neq 1$, то существует решение $a_n^p = d$, где $d = c/(1 - \sum \alpha_i)$. Если же $\sum \alpha_i = 1$, то существует решение $a_n^p = dn$, где $d = c/\sum i\alpha_i$ и $\sum i\alpha_i \neq 0$.

Производящие функции

Для решения рекуррентных соотношений и выполнения комбинаторных подсчетов в ряде случаев удобно использовать аппарат производящих функций. Последовательности $\{a_n\}$ поставим в соответствие формальный ряд

$$G(z) = \sum_{n=0}^{\infty} a_n z^n. \qquad (1.12)$$

Определение 1.4. Ряд $G(z)$, имеющий вид (1.12), называется *производящей функцией* для данной последовательности $\{a_n\}$.

Соответствие между некоторой последовательность $\{a_n\}$ и рядом $G(z)$ обычно выражается словами «последовательность генерируется производящей функцией», «производящая функция генерирует (порождает, производит) последовательность», или математическими формулами вида:

$$(a_0, a_1, a_2, \dots) \Leftrightarrow (a_0 + a_1 z + a_2 z^2 + \cdots).$$

Сумма (1.12) включает все $n \geq 0$, но иногда оказывается полезным расширение диапазона суммирования на все целые n. Этого можно добиться, если положить $a_{-1} = a_{-2} = \cdots = 0$ для всех $n < 0$. В таких случаях можно по-прежнему говорить о последовательности $\{a_n\} = a_0, a_1, a_2, \dots$, считая, что a_n не существует для отрицательных n.

Перед тем, как рассмотреть операции над производящими функциями, отметим, что производящая функция, во-первых, есть символьная конструкция. Это значит, что вместо переменной z может быть использован произвольный объект, для которого определены операции сложения и умножения. Во-вторых, вопрос о сходимости или расходимости ряда (1.12) рассматриваться не будет, что является важным аспектом изучения числовых рядов. В-третьих, вопрос о вычислении данного ряда для конкретного значения z также опускается. Единственное, что необходимо при работе с производящими функциями (и в целом с бесконечными суммами), это понимание того, что объект z не обязательно суть число, хотя существуют ситуации, когда придавать ему свойства числа будет необходимо.

Рассмотрим основные операции над производящими функциями:

1. Сложение рядов: $G(z) = \sum_n a_n z^n$ и $H(z) = \sum_n b_n z^n$

$$G(z) + H(z) = \sum_n (a_n + b_n) z^n.$$

2. Умножение на число: $\alpha G(z) = \sum_n \alpha \cdot a_n z^n$.

3. Произведение рядов:

$$G(z) \cdot H(z) = \sum_n c_n z^n, c_n = \sum_{i=0}^{n} a_i b_{n-i}.$$

4. Производная:

$$G'(z) = \sum_n (n+1) a_{n+1} z^n \Leftrightarrow z G'(z) = \sum_n n a_n z^n.$$

В качестве примера использования операций над производящими функциями, получим производящую функцию для последовательности $1,1,1, \dots$. В этом случае $G(z) = 1 + z + z^2 + \cdots = 1 + z(1 + z + \cdots) = 1 + zG(z)$, откуда $G(z) = \frac{1}{1-z}$. В качестве следующего примера рассмотрим более общую последовательность $1, \alpha, \alpha^2, \dots$. Для нее производящая функция есть $G(z) = \frac{1}{1-\alpha z}$, так как $G(z) = 1 + \alpha z + \alpha^2 z^2 + \cdots = 1 + \alpha z(1 + \alpha z + \cdots) = 1 + \alpha z G(z)$.

Рассмотрим алгоритм, который позволяет найти решение рекуррентного соотношения путем использования производящих функций. Пусть нам задана последовательность $\{a_n\}$. Тогда необходимо:

1. Записать уравнение, выражающее a_n через другие элементы последовательности. Полученное уравнение должно оставаться справедливым для всех целых n с учетом того, что $a_{-1} = a_{-2} = \cdots = 0$.

2. Умножить обе части уравнения на z^n и просуммировать по всем n. В левой части получится сумма $\sum_n a_n z^n = G(z)$. Правую часть следует преобразовать так, чтобы она превратилась в другое выражение, включающее $G(z)$.

3. Решить полученное уравнение относительно $G(z)$, тем самым получив замкнутое выражение для $G(z)$.

4. Разложить $G(z)$ в степенной ряд и взять коэффициент при z^n, который является замкнутым видом для a_n.

В качестве примера рассмотрим последовательность Фибоначчи. Имеем

$$f_n = \begin{cases} 0, & n \le 0, \\ 1, & n = 1, \\ f_{n-1} + f_{n-2}, & n > 1. \end{cases}$$

Шаг 1 требует записать заданное уравнение в виде единственного уравнения без ветвлений. Таким образом

$$f_n = f_{n-1} + f_{n-2} + [n = 1]$$

Шаг 2 предлагает преобразовать уравнение для f_n в уравнение для $G(z)$. В этом случае последовательность преобразований будет иметь вид:

$$G(z) = \sum_n f_n z^n = \sum_n f_{n-1} z^n + \sum_n f_{n-2} z^n + \sum_n [n = 1] z^n =$$
$$= \sum_n f_n z^{n+1} + \sum_n f_n z^{n+2} + z = zG(z) + z^2 G(z) + z,$$

откуда следует, что $G(z) = \frac{1}{1-z-z^2}$.

Разложим полученную дробь на сумму двух дробей, т. е. $G(z) = \frac{A}{1-\alpha z} + \frac{B}{1-\beta z}$, где A, B, α, β – константы. С учетом этого замкнутый вид для $G(z)$записываем так:

$$G(z) = A \sum_{n \ge 0} (\alpha z)^n + B \sum_{n \ge 0} (\beta z)^n = \sum_{n \ge 0} (A \cdot \alpha^n + B \cdot \beta^n) z^n. \qquad (1.13)$$

Непосредственно разложение на сумму двух дробей можно получить, воспользовавшись методом «неопределенных коэффициентов». Опуская арифметические выкладки, получим

$$A = +1/\sqrt{5}\,, B = -1/\sqrt{5}\,,$$
$$\alpha = \left(1 + \sqrt{5}\right)/2, \beta = (1 - \sqrt{5})/2.$$

Следовательно, замкнутый вид для функции $G(z)$ можно записать как

$$G(z) = \sum_{n\geq 0}(\frac{1}{\sqrt{5}}(\varphi^n - (1-\varphi)^n)z^n, \varphi = \frac{1+\sqrt{5}}{2}.$$

Из полученного соотношения делаем вывод о том, что замкнутый вид для последовательности Фибоначчи выглядит следующим образом:

$$f_n = \frac{1}{\sqrt{5}}(\varphi^n - (1-\varphi)^n), \varphi = \frac{1+\sqrt{5}}{2}.$$

Основная теорема о рекуррентных соотношениях

Пусть задано рекуррентное соотношение следующего вида:

$$\begin{cases} T(1) = c, \\ T(n) = a \cdot T(n/b) + d(n),\ n > 1, \end{cases} \tag{1.14}$$

где a, b, c – некоторые положительные константы, $d(n)$ – заданная функция, которую принято называть *управляющей функцией*.

Отметим, что соотношение (1.14) можно применить только тогда, когда n является целой степенью числа b. Но если функция $T(n)$ достаточно гладкая, то полученное решение можно будет распространить на другие значения n.

Воспользуемся методом итераций для получения замкнутого вида рекуррентного соотношения (1.14). Для этого запишем несколько значений искомой функции $T(n)$:

$T(1) = c,$

$T(b) = aT(1) + d(b) = ac + d(b),$

$T(b^2) = aT(b) + d(b^2) = a^2c + ad(b) + d(b^2),$

…

$$T(b^k) = a^k c + a^{k-1}d(b) + a^{k-2}d(b^2) + \cdots + d(b^k) = a^k c + \sum_{j=0}^{k-1} a^j d\left(b^{k-j}\right).$$

В последнем равенстве первое и второе слагаемое называются *однородным* и *частным* решениями рекуррентного соотношения (1.14).

Для определенного класса управляющих функций можно найти точное решение рекуррентного соотношения (1.14).

Определение 1.5. Функция $d(n)$ целочисленного аргумента является *мультипликативной*, если для всех положительных целых чисел m и n справедливо $d(mn) = d(m)d(n)$. В частности, мультипликативной является функция $d(n) = n^p$.

Пусть в выражении для $T(b^k)$ управляющая функция является мультипликативной. Тогда для частного решения можно записать

$$\sum_{j=0}^{k-1} a^j d(b^{k-j}) = \sum_{j=0}^{k-1} a^j \left(d(b)\right)^{k-j} = d(b)^k \sum_{j=0}^{k-1} \left(\frac{a}{d(b)}\right)^j = \frac{a^k - d(b)^k}{\frac{a}{d(b)} - 1}.$$

Последнее выражение имеет смысл, если $a \neq d(b)$. В случае же равенства $a = d(b)$ имеем

$$d(b)^k \sum_{j=0}^{k-1} \left(\frac{a}{d(b)}\right)^j = k \cdot d(b)^k = k \cdot a^k.$$

Выполнив замену $n = b^k$, получим следующую теорему.

Теорема 2. Решением рекуррентного соотношения (1.11), где $d(n)$ – мультипликативная функция, является

$$T(n) = \begin{cases} c\, n^{\log_b a} + \dfrac{n^{\log_b a} - n^{\log_b d(b)}}{\dfrac{a}{d(b)} - 1}, & a \neq d(b) \\ c\, n^{\log_b a} + \log_b n \cdot n^{\log_b a}, & a = d(b). \end{cases} \qquad (1.15)$$

Теорема 2 позволяет находить точные решения, однако формула решения выглядит достаточно громоздко. Попробуем несколько «огрубить» результаты теоремы. Заметим, что имеют место следующие три ситуации:

1. Если $a > d(b)$, то порядок частного решения будет $O(n^{\log_b a})$. В этом случае однородное и частное решения имеют одинаковый порядок роста, не зависящий от функции $d(n)$.

2. Если $a < d(b)$, то порядок частного решения будет $O(n^{\log_b d(b)})$. В этом случае частное решение превышает однородное. Когда $d(n) = n^p$, порядок частного решения будет $O(n^p)$.

3. Если $a = d(b)$, то порядок частного решения превосходит порядок однородного на $\log_b n$. Когда $d(n) = n^p$, порядок частного решения равен $O(n^p \log n)$.

Предположим, что управляющая функция в рекуррентном соотношении (1.14) задана не точно, а в виде $O(n^p)$. Так как $O((nm)^p) = O(n^p m^p)$, то при анализе рекуррентного соотношения можно использовать технику, применявшуюся при выводе теоремы 2.

Теорема 3[*]. Пусть дано рекуррентное соотношение вида

$$T(n) = aT(n/b) + O(n^p).$$

Тогда

$$T(n) = \begin{cases} O(n^p), & p > \log_b a, \\ O(n^p \log_b n), & p = \log_b a, \\ O\left(n^{\log_b a}\right), & p < \log_b a. \end{cases}$$

[*] Более строгий вариант этой теоремы был сформулирован и доказан Дж. Бентли, Д. Хакен и Дж. Саксом в 1980 году.

Приведем некоторые другие рекуррентные соотношения, возникающие при анализе рекурсивных алгоритмов. Если рекурсия вызывает аддитивное уменьшение размерности задачи, то используется соотношение

$$\begin{cases} T(0) = c, \\ T(n) = aT(n-b) + d(n),\ n \geq 1. \end{cases} \tag{1.16}$$

Иногда рекурсия ведет к «квадратичному» уменьшению сложности. В этом случае возникает соотношение

$$\begin{cases} T(2) = c, \\ T(n) = a\sqrt{n}T\left(\sqrt{n}\right) + d(n),\ n \geq 3. \end{cases} \tag{1.17}$$

Для решения приведенных соотношений (1.16, 1.17) в некоторых частных случаях могут быть применены соображения, использовавшиеся при решении рекуррентного соотношения (1.14).

1.3. Задачи для самостоятельного решения

1. Расположить следующие функции в порядке степени роста:

а) n; б) $\sqrt{n}$; в) $\log n$; г) $\log\log n$; д) $\log^2 n$; е) $\frac{n}{\log n}$; ж) $\sqrt{n}\log^2 n$; з) $\left(\frac{1}{3}\right)^n$; и) 8.

2. Требуется оценить время выполнения следующей программы:

Процедура 1.3.1. Pascal Triangle

```
01:  Pascal Triangle Impl(n)
02:        C[0, 0] = 1
03:        for i = 1 to n do
04:              C[i, 0] = 1
05:              for j = 1 to i do
06:                    C[i, j] = C[i - 1, j] + C[i - 1, j - 1]
```

3. Требуется оценить время выполнения следующей программы:

Процедура 1.3.2. GCD

```
01:  GCD Impl(a, b)
02:        while (a ≠ 0) and (b ≠ 0) do
03:              if a > b then
04:                    a = a mod b
05:              else
06:                    b = b mod a
07:        return (a + b)
```

4. Решить рекуррентные соотношения:

а) $\begin{cases} a_0 = 1, a_1 = 8, \\ a_n = a_{n-1} + 6a_{n-2}; \end{cases}$ б) $\begin{cases} a_0 = 2, a_1 = 5, \\ a_n = 7a_{n-1} - 12a_{n-2}; \end{cases}$ в) $\begin{cases} a_0 = 1, a_1 = 0, \\ a_n = 4a_{n-1} - 4a_{n-2}; \end{cases}$

г) $\begin{cases} a_0 = 1, \\ a_n = a_{n-1} + n; \end{cases}$ д) $\begin{cases} a_0 = 4, a_1 = 5, \\ a_n = 5a_{n-1} - 6a_{n-2} + 6 \cdot 3^n; \end{cases}$ е) $\begin{cases} a_0 = 2, \\ a_n = 3a_{n-1} + 2^n. \end{cases}$

5. Используя соответствующие рекуррентные уравнения, найти суммы:

а) $1^2+2^2+...+n^2$; б) $1^3+2^3+...+n^3$; в) $k^1+k^2+...+k^n$;

г) $1+4+...+(3n-2)$; д) $1\cdot 2+...+n\cdot(n+1)$; е) $1^2\cdot 2+...+n^2\cdot(n+1)$.

6. Найдите замкнутый вид для последовательности Каталана, описываемой рекуррентным соотношением вида

$$\begin{cases} c_0 = 1, \\ c_n = c_0 c_{n-1} + c_1 c_{n-2} + \cdots + c_{n-1} c_0, \ n \geq 1. \end{cases}$$

7. Возвести заданное число A в натуральную степень n за как можно меньшее количество умножений.

8. Определить количество способов, которыми можно уплатить N рублей купюрами различного достоинства $n_1, n_2, \ldots, n_m$, беря не более одной купюры каждого достоинства.

9. Подсчитать количество разбиений выпуклого n-угольника ($n \geq 3$) на треугольники диагоналями, не пересекающимися внутри многоугольника.

10. Число из $2n$ цифр называется *счастливым билетом*, если сумма первых n цифр равна сумме последних n цифр. Найти количество счастливых билетов при заданном значении n.

11. Рассматриваются цепочки, которые состоят из символов (a, b, c, d, e). Четыре пары cd, ce, ed и ee никогда не встречаются в рассматриваемых цепочках, однако любая цепочка, не содержащая этих запрещенных пар (назовем такие цепочки *хорошими*), возможна. Требуется определить число хороших цепочек длины n.