МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ОБРАЗОВАНИЯ
«МИРЭА – РОССИЙСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»
РТУ МИРЭА

Институт ИКБ

Специальность (направление): 09.03.02 (информационные системы и технологии)

Кафедра: КБ-3 «Разработка программных решений и системного программирования»

Дисциплина: «Алгоритмы и структуры данных»

Практическая работа
на тему:

Анализ сложности рекурсивного алгоритма

Студент: _____ 29.10.2024 Крашенинников М.В.
подпись *Дата* *инициалы и фамилия*

Группа: БСБО-16-23 Шифр: 23Б0107

Преподаватель: _____ 29.10.2024 Филатов В.В.
подпись *дата* *инициалы и фамилия*

Москва 2024 г.

ЧАСТЬ 1-3

Для построения функции роста трудоемкости данной функции проведем построчно пооператорный анализ данного фрагмента:

(Студент: 107) ----> Оцените трудоемкость следующего фрагмента: (1) int f (int n) (2) { int s = 0, x = 2, y = -5 (3) for (int ii = 0; ii < 4*n; ii++) s = s + 2 * ii; y += s; (4) if (n < 2) return (s + x + y); (5) s = s + x * f(n-1) + f(n-1) (6) for (int j = 0; j < 3; j++) s = s + x*f(n-2); (7) for (int k = 0; k < 4; k += 2) s++; (8) return s; (9) }	(1) + 1
	(2) + 3
	(3) $+1+2+\sum_{ii=1}^{4n}(1+2+3)+1$
	(4) $\{1_{\text{при } n\geq 2}, 4_{\text{при } n<2}\}$
	(5) $+8+2F(n-1)$
	(6) $+1+1+\sum_{j=1}^{3}(1+1+5+F(n-2))$
	(7) $+1+1+\sum_{k=1}^{2}(2+1)$
	(8) + 1

$F(n) = 7+\sum_{ii=1}^{4n}(6)+1+1+8+2F(n-1)+1+1+$
$\sum_{j=1}^{3}(7+F(n-2))+1+1+\sum_{k=1}^{2}(3)+1$=27 + 24n + 2F(n-1) + 21 + 3F(n-2)=
=2F(n-1) + 3F(n-2) + 24n + 48, при $n \geq 2$.

Если условие (4) будет истинным, а это имеет место при n < 2, что соответствует порядку данного рекуррентного соотношения или начальному условию, то трудоемкость данной цепочки складывается из строк (1), (2), (3), (4) или

$$F(n) = 1+3+1+2+\sum_{ii=1}^{4n}(6)+1+4 = 12+24n, \text{ при } n<2.$$

Подставляя значения n = 0 и n = 1 в выражение, получаем начальные значения рекуррентного ряда:

$$F(0) = 12$$
$$F(1) = 12+24 = 36$$

Таким образом, для указанного фрагмента функция роста имеет вид:

$$F(n) = \begin{cases} 2F(n-1)+3F(n-2)+24n+48, \text{при } n\geq 2 \\ 12+24n, \text{при} < 2. \end{cases}$$

Или, используя рекуррентную запись обобщенного члена, имеем

$$F(n) = 2F(n-1)+3F(n-2)+24n+48,$$

при начальных значениях $F(0) = 12, F(1) = 36$.

Оно представляет собой ***линейное неоднородное рекуррентное уравнение первого порядка при заданных начальных условиях***.

Напомним, что решением рекуррентного уравнения является явная форма определения обобщенного члена с помощью некоторой функции $f(n)$, которая удовлетворяет как самому рекуррентному уравнению, так и начальным условиям.

Решение линейного неоднородного рекуррентного уравнения $F_{\text{неодн}}(n)$ ищется как

$$F_{\text{неодн}}(n) = F_{\text{одн}}(n) + F^*(n),$$

где $F_{\text{одн}}(n)$ – решение соответствующего линейного однородного рекуррентного уравнения;

$F^*(n)$ – некоторое *частное решение*.

Вначале получим соответствующее исходному линейное однородное рекуррентное уравнение. Для этого перегруппируем элементы так, чтобы в левой части были слагаемые, выражающие *только* рекуррентную зависимость, а в правой – остальные слагаемые. Получаем

$$F(n) - 2F(n-1) - 3F(n-2) = 24n + 48.$$

Рассмотрим левую часть, приравнивая ее к нулю

$$F(n) - 2F(n-1) - 3F(n-2) = 0.$$

Оно представляет собой искомое линейное однородное рекуррентное уравнение 1-го порядка.

Напомним, что по рекуррентному соотношению может быть сформирована последовательность чисел. Обозначим ее через $\{x_n\}$, причем первым её членом будет x_0. На основе k-штук начальных значений можно вычислить следующий элемент последовательности, то есть

$$x_k = f(x_{k-1}, x_{k-2}, \ldots, x_1, x_0),$$

или для обобщенного члена

$$x_{n+k} = f(x_{n+k-1}, x_{n+k-2}, \ldots, x_{n+1}, x_{n+0}),$$

что для линейного однородного рекуррентного уравнения с коэффициентами $\{p_i\}$, $0 \le i < k$, соответствует

$$x_{n+k} = p_{k-1}x_{n+k-1} + p_{k-2}x_{n+k-2} + p_1x_n + p_{k-1}x_n$$

или в каноническом виде

$$x_{n+k} + p_{k-1}x_{n+k-1} + p_{k-2}x_{n+k-2} + p_1x_n + p_{k-1}x_n = 0,$$

где $\{p_i\}$, $0 \le i < k$ –некторые числа.

Решим его, то есть получим формулу для вычисления обобщенного члена как функцию от n. Причем решение бывает общим и частным.

Частным решением рекуррентного соотношения называется такая последовательность $\{x_n\}$, что при подстановке в это уравнение для каждого n соответствующих элементов этой последовательности получается верное равенство.

Общим решением рекуррентного соотношения называется множество всех частных решений.

Решение линейного однородного рекуррентного уравнения ищется при помощи соответствующего характеристического многочлена.

Для линейного однородного рекуррентного уравнения вида

$$x_{n+k} + p_{k-1}x_{n+k-1} + p_{k-2}x_{n+k-2} + p_1x_n + p_{k-1}x_n = 0$$

характеристическим многочленом называется многочлен комплексной переменной

$$P(x) = x^k + p_{k-1}x^{k-1} + p_{k-2}x^{k-2} + \ldots + p_1x + p_0.$$

Степень характеристического многочлена равная порядку k линейного однородного рекуррентного уравнения. Чтобы найти корни характеристического уравнения, надо его приравнять к нулю

$$x^k + p_{k-1}x^{k-1} + p_{k-2}x^{k-2} + \ldots + p_1x + p_0 = 0.$$

Для нашей задачи характеристический полином имеет вид

$$x^2 - 2x - 3 = 0.$$

Корнями будут значения $x_1 = 3, x_2 = -1.$

Если отсутствуют среди k-штук корней кратные (равные) корни, то общее решение ищется в виде

$$\left\{\alpha_1 x_1^n + \alpha_2 x_2^n + \ldots + \alpha_k x_k^n\right\}.$$

Для удобства последующих вычислений коэффициенты обозначим греческими буквами без индексов:

$$\{\alpha_1, \alpha_2, \ldots, \alpha_k\} \rightarrow \{\alpha, \beta, \gamma, \ldots\}.$$

Обобщенное решение нашей задачи будет иметь вид:

$$F_{\text{одн}}(n) = \alpha x_1^n + \beta x_2^n,$$

$$F_{\text{одн}}(n) = \alpha 3^n + \beta(-1)^n.$$

Решив линейное однородное рекуррентное уравнение, вернемся к неоднородному линейному уравнению.

Если правая часть представима в виде

$$\lambda^n \cdot P(n),$$

где $P(n)$ – некоторый многочлен от аргумента n с коэффициентами

$$\{a_0, a_1, a_2,...,a_m\} \xrightarrow{\text{для удобства вычислений}} \{a, b,c,...\},$$

λ – некоторое число ($\lambda \neq 0$),

то для линейного неоднородного рекуррентного уравнения вида

$$x_{n+k} + \sum_{j=1}^{k} p_{k-j} x_{n+k-j} = \left(a \cdot n^m + b \cdot n^{m-1} + ... + cn + d\right) \cdot \lambda^n$$

в котором $\{a_0, a_1, a_2,...,a_m, p_1, p_2,...,p_k,\lambda\}$ – некоторые числа, $\lambda \neq 0, k \geq 1$, существует частное решение вида

1) $\left(c_0 + c_1 n + c_2 n^2 + ... + c_m n^m\right) \cdot \lambda^n$, где $c_0, c_1, c_2,...,c_m$– некоторые числа, если λ – *НЕ* является корнем характеристического многочлена соответствующего линейного однородного уравнения из левой части, то есть $\lambda \neq x_i$;
2) $n^r \cdot \left(c_0 + c_1 n + c_2 n^2 + ... + c_m n^m\right) \cdot x^n$, где $c_0, c_1, c_2,...,c_m$– некоторые числа, если $\lambda \neq 0$ – корень кратности r характеристического многочлена соответствующего линейного однородного уравнения из левой части, то есть $\lambda = x_i$.

Поскольку среди значений корней характеристического уравнения нет значения 1 ($x_1 = 3, x_2 = -1$), то есть 1 (единица) не является корнем характеристического уравнения второго порядка, то можно положить $\lambda = 1$, что позволит получить справа полином, а значение m ограничим значением порядка полинома $24n + 48$ из правой части неоднородного линейного рекуррентного уравнения, т.е. $m = 1$. Частное решение $F^*(n)$ будем искать в виде полинома порядка m ($m = 1$):

$$F^*(n) = (b \cdot n + c).$$

Тогда следующие члены (n+1) и (n+2) будут иметь вид

$$F^*(n+1) = (b \cdot (n+1) + c),$$
$$F^*(n+2) = (b \cdot (n+2) + c).$$

Подставим эти значения в правую часть полученного ранее выражения

$$F(n) - 2F(n-1) - 3F(n-2) = 24n + 48,$$

переписав его так, чтобы оно существовало для всех натуральных $n > 0$, то есть реализовав подстановку $n \leftarrow n$+2

$F(n+2) - 2F(n+1) - 3F(n) = 0 + 24(n+2) + 48$:

$F(n+2) - 2F(n+1) - 3F(n) = (b\cdot(n+2) + c) - 2(b\cdot(n+1) + c) - 3(b\cdot n + c) = bn + 2b + c - 2(bn + b + c) - 3(b\cdot n + c) =$

$= bn + 2b + c - 2bn - 2b - 2c - 3b\cdot n - 3c =$

приведем подобные слагаемые

$= -4bn - 4c$

Приравняем полученное выражение к правой части

$$-4bn - 4c = 24(n+2) + 48$$

$$-4bn - 4c = 24n + 96$$

$$4bn + 24n + 4c + 96 = 0$$

$$n(b+6) + (c+24) = 0$$

т.к. $n > 0$, то найдем такие b и c, которые совместно обращают каждую скобку в 0:

$$\begin{cases} b + 6 = 0 \\ c + 24 = 0 \end{cases} \begin{cases} b = -6 \\ c = -24 \end{cases}$$

Следовательно,

$$F^*(n) = (-6\cdot n - 24)$$

Как было сказано выше, обобщенный член $F(n)$

$$F(n) = F_{\text{неодн}}(n) = F_{\text{одн}}(n) + F^*(n),$$

$$F(n) = \alpha 3^n + \beta(-1)^n - 6n - 24$$

Найденный обобщенный член F(n), должен соответствовать любому значению последовательности, включая и начальные значения $F(0) = 12$ и $F(1) = 36$, то есть

$$\begin{cases} F(0) = \alpha 3^n + \beta(-1)^n - 6n - 24 \\ F(1) = \alpha 3^n + \beta(-1)^n - 6n - 24 \end{cases}$$

$$\begin{cases} \alpha 3^0 + \beta(-1)^0 - 6\cdot 0 - 24 = 12 \\ \alpha 3^1 + \beta(-1)^1 - 6\cdot 1 - 24 = 36 \end{cases}$$

$$\begin{cases} \alpha + \beta = 36 \\ 3\alpha - \beta = 66 \end{cases}, \quad \begin{cases} \alpha = 36 - \beta \\ \beta = 3\alpha - 66 \end{cases}, \quad \begin{cases} \alpha = 36 - 3\alpha + 66 \\ \beta = 3\alpha - 66 \end{cases},$$

$$\begin{cases} 4\alpha = 102 \\ \beta = 3\alpha - 66 \end{cases}, \quad \begin{cases} \alpha = 25{,}5 \\ \beta = 3\cdot 25{,}5 - 66 \end{cases}, \quad \begin{cases} \alpha = 25{,}5 \\ \beta = 10{,}5 \end{cases}$$

Общее решение, проходящее через заданные начальные условия, будет иметь вид

$$F(n) = 25{,}5 \cdot 3^n + 10{,}5 \cdot (-1)^n - 6n - 24.$$

Сделаем проверку.

1. Выражение

$$F(n) = 25{,}5 \cdot 3^n + 10{,}5 \cdot (-1)^n - 6n - 24$$

должно удовлетворять самому рекуррентному соотношению:

$$F(n+2) - 2F(n+1) - 3F(n) = 24n + 96.$$

Проверим. Действительно

$$F(n) = 25{,}5 \cdot 3^n + 10{,}5 \cdot (-1)^n - 6n - 24$$
$$F(n+1) = 25{,}5 \cdot 3^{n+1} + 10{,}5 \cdot (-1)^{n+1} - 6(n+1) - 24$$
$$F(n+2) = 25{,}5 \cdot 3^{n+2} + 10{,}5 \cdot (-1)^{n+2} - 6(n+2) - 24$$

Подставим выражения в левую часть

$$F(n+2) - 2F(n+1) - 3F(n) =$$
$$= 25{,}5 \cdot 3^{n+2} + 10{,}5 \cdot (-1)^{n+2} - 6(n+2) - 24 -$$
$$- 2 \cdot \left(25{,}5 \cdot 3^{n+1} + 10{,}5 \cdot (-1)^{n+1} - 6(n+1) - 24\right) -$$
$$- 3 \cdot \left(25{,}5 \cdot 3^n + 10{,}5 \cdot (-1)^n - 6n - 24\right) =$$
$$= 25{,}5 \cdot (9 - 2 \cdot 3 - 3) \cdot 3^n + 10{,}5 \cdot \left((-1)^2 - 2 \cdot (-1) - 3\right) \cdot (-1)^n +$$
$$+ (-6n - 12 + 12n + 12 + 18n) + (-1 + 2 + 3) \cdot 24 =$$
$$= 25{,}5 \cdot 0 \cdot 3^n + 10{,}5 \cdot 0 \cdot (-1)^n + 24n + 4 \cdot 24 =$$
$$= 24n + 96,$$

Что в точности соответствует правой части рекуррентного уравнения.

2. Проверим, что решение удовлетворяет начальным значениям

$$F(0) = 25{,}5 \cdot 3^0 + 10{,}5 \cdot (-1)^0 - 6 \cdot 0 - 24 =$$
$$= 25{,}5 + 10{,}5 - 24 = 12$$
$$F(1) = 25{,}5 \cdot 3^1 + 10{,}5 \cdot (-1)^1 - 6 \cdot 1 - 24 = 36$$

Таким образом, функция роста трудоемкости выполнения рекурсивной функции составляет

$$F(n) = f(n) = 25{,}5 \cdot 3^n + 10{,}5 \cdot (-1)^n - 6n - 24;$$

для сведения, асимптотическая оценка ее составляет

$$T(n) = O(f(n)) = O\left(\max\left\{3^n ; (-1)^n ; n; -24\right\}\right) = O(3^n)$$

Напомним, что в ходе анализа исходного текста программы была получена следующая формула роста трудоемкости алгоритма

$$F(n) = \begin{cases} 2F(n-1) + 3F(n-2) + 24n + 48, \text{ при } n \geq 2 \\ 12 + 24n, \text{ при } < 2 \end{cases},$$

которую необходимо экспериментально подтвердить.

```cpp
#include <iostream>
#include "sysinfoapi.h"

int k = 2;
size_t r, q;

long long f(int n) {
    if (n >= k) {
        ++r;
        return f(n-1) + f(n-1) + f(n-2) + f(n-2) + f(n-2) + 24*n + 48;
    } else {
        ++q;
        return 12 + 24*n;
    }
}

int main() {
    int N, N_MAX; N_MAX = 24;
    size_t N_OP;
    long long t1, t2;

    for (N = 0; N <= N_MAX; ++N) {
        r = q = 0;
        t1 = GetTickCount();
        N_OP = f(N);
        t2 = GetTickCount();
        std::cout << "N = " << N << " r = " << r << " q = " << q << " r+q = " << r + q;
        std::cout << " N_op = " << N_OP << " T = " << t2-t1 << std::endl;
    }
}
```

Результат эксперимента – запуск программы, представлен на следующем скриншоте

```
N = 0 r = 0 q = 1 r+q = 1 N_op = 12 T = 0
N = 1 r = 0 q = 1 r+q = 1 N_op = 36 T = 0
N = 2 r = 1 q = 5 r+q = 6 N_op = 204 T = 0
N = 3 r = 3 q = 13 r+q = 16 N_op = 636 T = 0
N = 4 r = 10 q = 41 r+q = 51 N_op = 2028 T = 0
N = 5 r = 30 q = 121 r+q = 151 N_op = 6132 T = 0
N = 6 r = 91 q = 365 r+q = 456 N_op = 18540 T = 0
N = 7 r = 273 q = 1093 r+q = 1366 N_op = 55692 T = 0
N = 8 r = 820 q = 3281 r+q = 4101 N_op = 167244 T = 0
N = 9 r = 2460 q = 9841 r+q = 12301 N_op = 501828 T = 0
N = 10 r = 7381 q = 29525 r+q = 36906 N_op = 1505676 T = 0
N = 11 r = 22143 q = 88573 r+q = 110716 N_op = 4517148 T = 0
N = 12 r = 66430 q = 265721 r+q = 332151 N_op = 13551660 T = 0
N = 13 r = 199290 q = 797161 r+q = 996451 N_op = 40655124 T = 0
N = 14 r = 597871 q = 2391485 r+q = 2989356 N_op = 121965612 T = 0
N = 15 r = 1793613 q = 7174453 r+q = 8968066 N_op = 365897004 T = 16
N = 16 r = 5380840 q = 21523361 r+q = 26904201 N_op = 1097691276 T = 46
N = 17 r = 16142520 q = 64570081 r+q = 80712601 N_op = 3293074020 T = 157
N = 18 r = 48427561 q = 193710245 r+q = 242137806 N_op = 9879222348 T = 468
N = 19 r = 145282683 q = 581130733 r+q = 726413416 N_op = 29637667260 T = 1454
N = 20 r = 435848050 q = 1743392201 r+q = 2179240251 N_op = 88913002092 T = 4546
N = 21 r = 1307544150 q = 5230176601 r+q = 6537720751 N_op = 266739006516 T = 14141
N = 22 r = 3922632451 q = 15690529805 r+q = 19613162256 N_op = 800217019884 T = 42125
N = 23 r = 11767897353 q = 47071589413 r+q = 58839486766 N_op = 2400651059916 T = 114797
N = 24 r = 35303692060 q = 141214768241 r+q = 176518460301 N_op = 7201953180108 T = 321109
```

$$F(2) = 25{,}5 \cdot 3^2 \ + \ 10{,}5 \cdot(-1)^2 \ - 6 \cdot 2 - 24 = 204$$

Покажем согласованность экспериментальных и расчётных данных. Выберем произвольную строку (например, *N*=5), вычислим по формуле решения рекуррентного уравнения и сравним с полученным значением N_op=6132:

$$F(n) = 25{,}5 \cdot 3^n \ + \ 10{,}5 \cdot (-1)^n \ - 6n - 24$$

$$F(5) = 25{,}5 \cdot 3^5 \ + \ 10{,}5 \cdot(-1)^5 \ - 6 \cdot 5 - 24 = 6132$$

Заметим, что переменные *r* и *q* корректно подсчитывают количество внутренних узлов и листьев соответственно дерева рекурсии, выражение которой указано в строке 10 и 12. Соответствующие значения общего числа узлов дерева рекурсии (*r* + *q*) может быть вычислено аналитически (проверено или соотнесено с программным подсчетом), если решить следующее линейное неоднородное уравнение, взяв в качестве базовой операции факт однократного исполнения функции *f*(), т.е.

(Студент: 107) ----> Оцените трудоемкость следующего фрагмента: (1) int f (int n) (2) { int s = 0, x = 2, y = -5 (3) for (int ii = 0; ii < 4*n; ii++) s = s + 2 * ii; y += s; (4) if (n < 2) return (s + x + y); (5) s = s + x * f(n-1) + f(n-1) (6) for (int j = 0; j < 3; j++) s = s + x*f(n-2); (7) for (int k = 0; k < 4; k += 2) s++; (8) return s; (9) }	(1) +1 (2) + 0 (3) + 0 +$\sum_{ii=1}^{4*n}(0 + F(n-1))$ (4) + 0 (5) + 0 +$\sum_{ii=1}^{4*n}(0 + F(n-1))$ (6) + 0 +$\sum_{j=1}^{3}(0 + F(n-2))$ (7) + 0 +$\sum_{k=1}^{2}(0)$ (8) + 0

(здесь нули указаны для наглядности внесенных изменений)

С учетом внесенных изменений для указанного фрагмента функция роста имеет вид:

$$F(n) = \begin{cases} 2F(n-1) + 3F(n-2) + 1, \text{при } n \geq 2 \\ 1, \text{при} < 2. \end{cases}$$

Или, используя рекуррентную запись обобщенного члена, имеем

$$F(n) = 2F(n-1) + 3F(n-2) + 1,$$

при начальных значениях $F(0) = 1, F(1) = 1$.

Как уже указывалось, оно представляет собой ***линейное неоднородное рекуррентное уравнение первого порядка при заданных начальных условиях***.

Как уже показывалось, решение линейного неоднородного рекуррентного уравнения $F_{\text{неодн}}(n)$ будем искать как сумму решения $F_{\text{одн}}(n)$ соответствующего линейного однородного рекуррентного уравнения и некоторого *частного решения* $F^*(n)$, т.е.

$$F_{\text{неодн}}(n) = F_{\text{одн}}(n) + F^*(n),$$

Сгруппировав в левой части члены, представляющие вызов или выполнение функции *f*(), а справа – трудоемкость однократного исполнения тела функции, получаем

$$F(n) - 2F(n-1) - 3F(n-2) = 1.$$

Рассмотрим левую часть, приравнивая ее к нулю, получим линейное однородное рекуррентное уравнение

$$F(n) - 2F(n-1) - 3F(n-2) = 0.$$

Решение также будем искать с помощью корней характеристического полинома вида

$$x^2 - 2x - 3 = 0.$$

Корнями будут значения $x_1 = 3, x_2 = -1$ (заметим, что они те же самые, что были нами найдены ранее, так как структура рекурсивных вызовов не изменилась). Так как корни не кратные ($x_1 \neq x_2$), решение линейного однородного рекуррентного уравнения будем искать в виде

$$F_{\text{одн}}(n) = \alpha x_1^n + \beta x_2^n,$$

с учетом найденных корней:

$$F_{\text{одн}}(n) = \alpha 3^n + \beta(-1)^n .$$

Если правая часть линейного неоднородного рекуррентного соотношения представима в виде

$$\lambda^n \cdot P(n),$$

где $P(n)$ – некоторый многочлен от аргумента *n* с коэффициентами,

λ – некоторое число ($\lambda \neq 0$),

то для линейного неоднородного рекуррентного уравнения вида

$$x_{n+k} + \sum_{j=1}^{k} p_{k-j} x_{n+k-j} = \left(a \cdot n^m + b \cdot n^{m-1} + \ldots + cn + d\right) \cdot \lambda^n$$

в котором $\{a_0, a_1, a_2,...,a_m, p_1, p_2,...,p_k,\lambda\}$ – некоторые числа, $\lambda \neq 0, k \geq 1$, существует частное решение вида $\left(c_0 + c_1 n + c_2 n^2 + ... + c_m n^m\right) \cdot \lambda^n$, где $c_0, c_1, c_2,...,c_m$ – некоторые числа, если λ – *НЕ* является корнем характеристического многочлена соответствующего линейного однородного уравнения из левой части, то есть $\lambda \neq x_i$.

Заметим, что при $\lambda = 1$; $d = 1$; $a, b \ldots c = 0$ выражение сводится к правой части заданного нам рекуррентного соотношения, порядок нашего полинома справа равен 0 => $m = 0$, поэтому частное решение будем искать в виде

$$F^*(n) = (c_0) \cdot 1^n = c_0 = const.$$

Подставляя частное решение в общую формулу, заметив, что в нашем случае частное решение не зависит от *n,* т.е.

$$F^*(n) = F^*(n+1) = F^*(n+2) = c_0,$$

получаем

$$c_0 - 2c_0 - 3c_0 = 1 \;\; => \;\; c_0 = -0{,}25\,.$$

Откуда

$$F_{\text{неодн}}(n) = F_{\text{одн}}(n) + F^*(n) = \alpha 3^n + \beta(-1)^n - 0{,}25.$$

Найдем α и β, поскольку $F_{\text{неодн}}(n)$ должно удовлетворять начальным условиям, то

$$\begin{cases} F(0) = \alpha 3^n + \beta(-1)^n - 0{,}25 \\ F(1) = \alpha 3^n + \beta(-1)^n - 0{,}25 \end{cases}$$

$$\begin{cases} \alpha 3^0 + \beta(-1)^0 - 0{,}25 = 1 \\ \alpha 3^1 + \beta(-1)^1 - 0{,}25 = 1 \end{cases}$$

$$\begin{cases} \alpha + \beta = 1{,}25 \\ 3\alpha - \beta = 1{,}25 \end{cases}, \quad \begin{cases} \alpha = 1{,}25 - \beta \\ \beta = 3\alpha - 1{,}25 \end{cases}, \quad \begin{cases} \alpha = 1{,}25 - 3\alpha + 1{,}25 \\ \beta = 3\alpha - 1{,}25 \end{cases},$$

$$\begin{cases} 4\alpha = 2{,}5 \\ \beta = 3\alpha - 1{,}25 \end{cases}, \quad \begin{cases} \alpha = 0{,}625 \\ \beta = 3 \cdot 0{,}625 - 1{,}25 \end{cases}, \quad \begin{cases} \alpha = 0{,}625 \\ \beta = 0{,}625 \end{cases}$$

Решением рекуррентного уравнения, выражающего функцию роста трудоемкости алгоритма (рост числа вызовов рекурсивной функции) будет выражение

$$F(n) = 0.625 \cdot 3^n + 0.625 \cdot (-1)^n - 0{,}25.$$

Например,

$$F(\mathbf{4}) = 0.625 \cdot 3^4 + 0.625 \cdot (-1)^4 - 0{,}25 = 50{,}625 + 0{,}625 - 0{,}25 = \mathbf{51},$$

что соответствует четвертой строке скриншота результата работы

$N = \mathbf{4}$ $r = 10$ $q = 41$ $r+q = \mathbf{51,}$

$$F(\mathbf{10}) = 0.625 \cdot 3^{10} + 0.625 \cdot (-1)^{10} - 0{,}25 = \mathbf{36906},$$

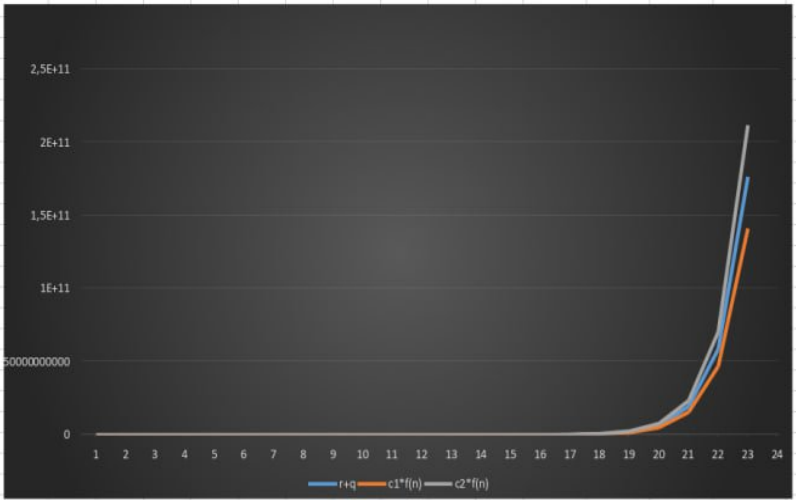
что соответствует десятой строке скриншота результата работы

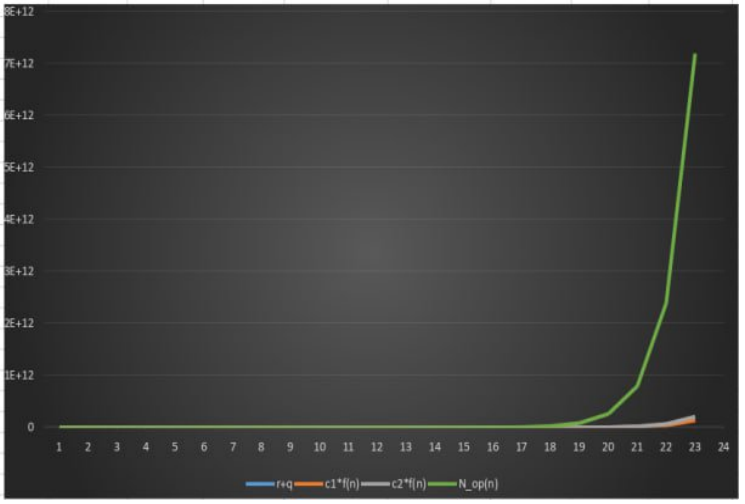
$N = \mathbf{10}$ $r = 7381$ q = 29525 $r+q = \mathbf{36906}$

Аналогично можно подтвердить ***согласованность (совпадение) по всем строкам экспериментальных и расчетных данных***.

ЧАСТЬ 4

n	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
r+q	6	16	51	151	456	1366	4101	12301	36906	110716	332151	996451	2989356	8968066	26904201	80712601	242137806	726413416	2179240251	6537720751	1,9613E+10	5,8839E+10	1,76518E+11
c1*f(n)	4,5	13,5	40,5	121,5	364,5	1093,5	3280,5	9841,5	29525	88574	265720,5	797161,5	2391485	7174453,5	21523360,5	64570081,5	193710245	581130733,5	1743392201	5230176602	1,5691E+10	4,7072E+10	1,41215E+11
c2*f(n)	6,75	20,25	60,75	182,25	546,75	1640,3	4920,8	14762	44287	132860	398580,8	1195742	3587227	10761680,3	32285040,8	96855122,25	290565367	871696100,3	2615088301	7845264902	2,3536E+10	7,0607E+10	2,11822E+11
c1	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5	0,5
c2	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75	0,75
N_op(n)	204	636	2028	6132	18540	55692	167244	501828	2E+06	5E+06	13551660	40655124	1,22E+08	365897004	1097691276	3293074020	9879222348	2,9638E+10	88913002092	2,66739E+11	8,00217E+11	2,4007E+12	7,20195E+12

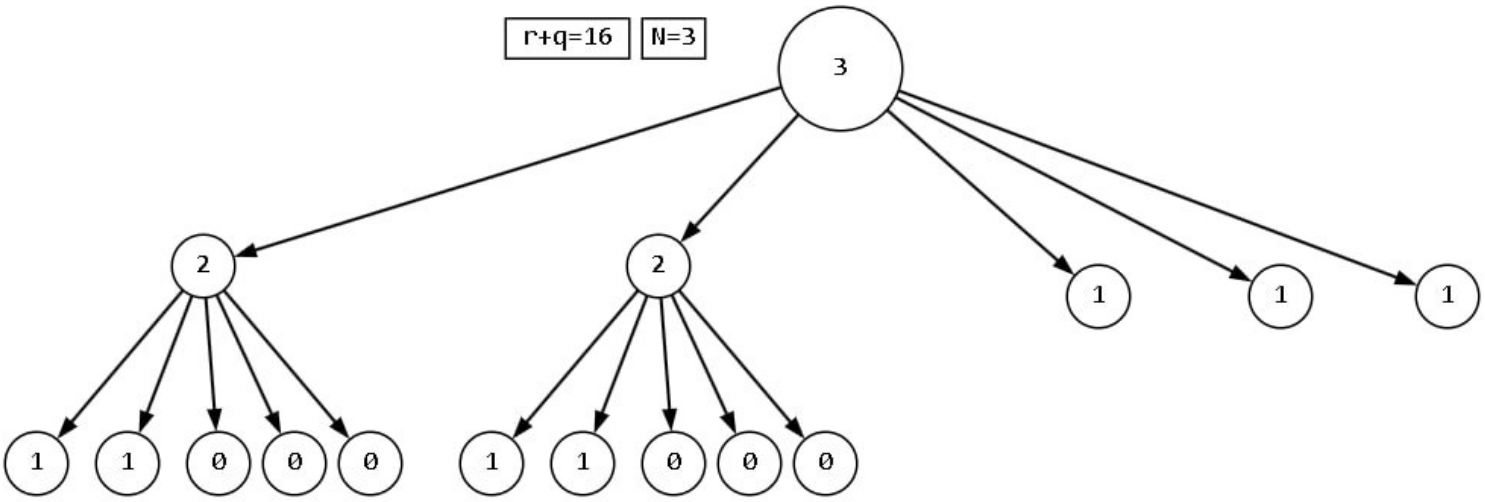


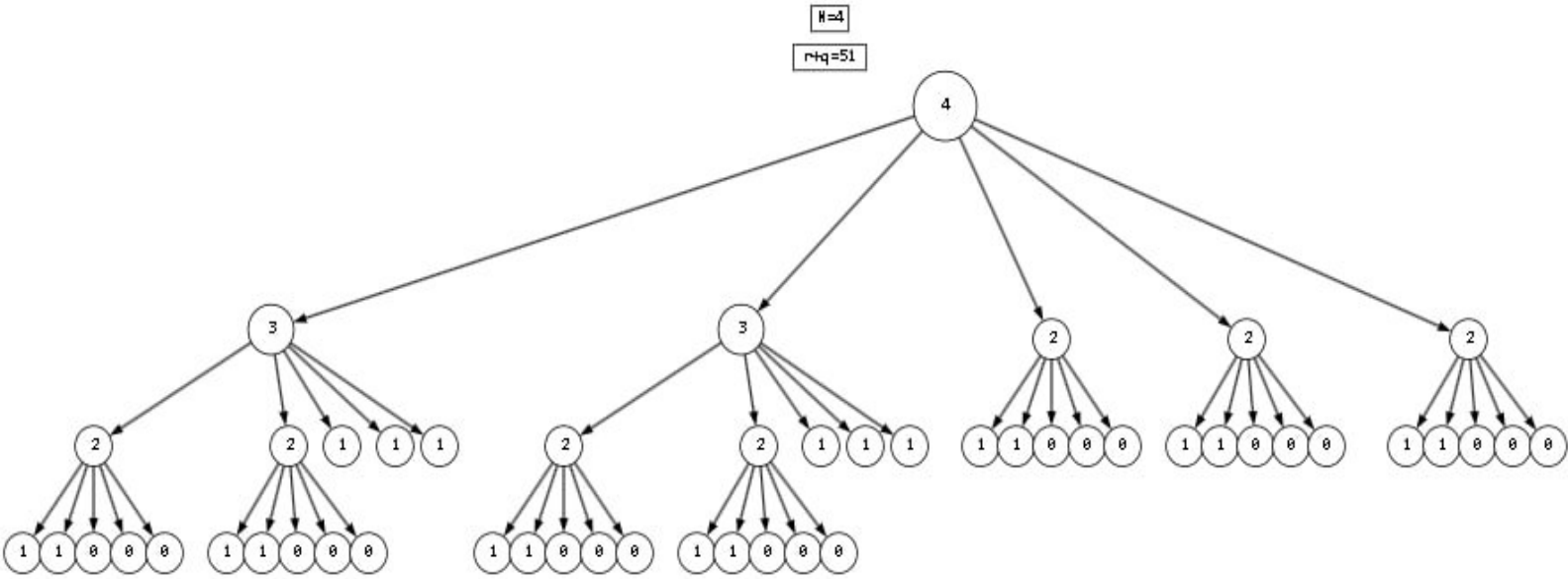


В нашем случае f(n) =Θ(3^n).

ЧАСТЬ 5

Возьмём N=3: r+q=16 и N=4: r+q=51





ЧАСТЬ 6

Немного модернизируем код (закомментируем неиспользуемые части, чтобы вывести только размер стека).

```cpp
#include <iostream>
#include "sysinfoapi.h"

#define VAR 107
#define max(a,b) (((a) > (b)) ? (a) : (b))

int k = 2;
size_t r, q;

int f(int n);

long long t(int n) {
    if (n >= k) {
        ++r;
        return t(n-1) + t(n-1) + t(n-2) + t(n-2) + t(n-2) + 24*n + 48;
    } else {
        ++q;
        return 12 + 24*n;
    }
}
```

```cpp
long long maxStackSize = 1, currStackSize = 1;

int main() {
    int N, N_MAX; N_MAX = 24;
    size_t N_OP;
    long long t1, t2;

    for (N = 1; N <= N_MAX; ++N) {
        // r = q = 0;
        // t1 = GetTickCount();
        // N_OP = t(N);
        // t2 = GetTickCount();
        f(N);
        // std::cout << "N = " << N << " r = " << r << " q = " << q << " r+q = " << r + q;
        // std::cout << " N_op = " << N_OP << " T = " << t2-t1;
        std::cout << "N = " << N << " MAX_STACK_SIZE = " << maxStackSize << std::endl;
        maxStackSize = currStackSize = 1; // за возвращаемое значение
    }
}
```

```
42  int f(int n) {
43      ++currStackSize; // за аргумент n
44      currStackSize += 3; // за 3 инициализации s, x, y
45      int s = 0, x = 2, y = -5;
46      ++currStackSize; // за инициализацию ii
47      maxStackSize = max(currStackSize, maxStackSize);
48      for (int ii = 0; ii < 4 * n; ii++) s = s + 2 * ii;
49      --currStackSize; // высвобождение ii
50
51      y += s;
52      if (n < 2) {
53          currStackSize -= 4; // высвобождаем s, x, y, n
54          return (s + x + y);
55      }
56
57      currStackSize += 2; // f(n-1) дважды
58      maxStackSize = max(currStackSize, maxStackSize);
59      s = s + x * f(n-1) + f(n-1);
60      currStackSize -= 2; // конец f(n-1)
61
62      ++currStackSize; // инициализация j
63      maxStackSize = max(currStackSize, maxStackSize);
```

```
64      for (int j = 0; j < 3; j++) {
65          ++currStackSize; // f(n-2)
66          maxStackSize = max(currStackSize, maxStackSize);
67          s = s + x * f(n-2);
68          --currStackSize; // конец f(n-2)
69      }
70
71      ++currStackSize; // инициализация k
72      maxStackSize = max(currStackSize, maxStackSize);
73      for (int k = 0; k < 4; k += 2) s++;
74
75      currStackSize -= 6; // освобождение s, x, y, k, n и возвращаемого значения
76      return s;
77  }
```

Результат выполнения программы:

```
N = 1 MAX_STACK_SIZE = 6
N = 2 MAX_STACK_SIZE = 12
N = 3 MAX_STACK_SIZE = 18
N = 4 MAX_STACK_SIZE = 24
N = 5 MAX_STACK_SIZE = 30
N = 6 MAX_STACK_SIZE = 36
N = 7 MAX_STACK_SIZE = 42
N = 8 MAX_STACK_SIZE = 48
N = 9 MAX_STACK_SIZE = 54
N = 10 MAX_STACK_SIZE = 60
N = 11 MAX_STACK_SIZE = 66
N = 12 MAX_STACK_SIZE = 72
N = 13 MAX_STACK_SIZE = 78
N = 14 MAX_STACK_SIZE = 84
N = 15 MAX_STACK_SIZE = 90
N = 16 MAX_STACK_SIZE = 96
N = 17 MAX_STACK_SIZE = 102
N = 18 MAX_STACK_SIZE = 108
N = 19 MAX_STACK_SIZE = 114
N = 20 MAX_STACK_SIZE = 120
N = 21 MAX_STACK_SIZE = 126
N = 22 MAX_STACK_SIZE = 132
N = 23 MAX_STACK_SIZE = 138
N = 24 MAX_STACK_SIZE = 144
```

Предполагаю, что формула оценки требуемого размера стека исходя из размерности задачи в нашем случае равна 6*n.

ЧАСТЬ 7

Вывод: в ходе выполнения практической работы «Анализ сложности рекурсивного алгоритма» были выполнены следующие задачи:
- Построение функции роста рекурсивной программы

- Получение функции роста трудоемкости рекурсивной программы как решение рекуррентного уравнения

- Экспериментальное определение трудоемкости рекурсивной программы

- Экспериментальное определение трудоемкости рекурсивной программы на примере определения скорости роста числа узлов дерева рекурсии

- Построение дерева рекурсии

- Оценка пространственной трудоемкости рекурсивной программы

Литература

1. *Альфред В. Ахо, Джон Э. Хопкрофт, Джеффри Д. Ульман. Структуры данных и алгоритмы. – М.: Издательский дом «Вильямс», 2016*
2. *Кормен Т., Лейзерсон Ч., Ривест Р. Алгоритмы: Построение и анализ. – М.: «Вильямс», 2020*
3. *Левитин А.В., Красиков И.В. Алгоритмы: введение в разработку и анализ.: Пер. с англ.– М.:Издательский дом «Вильямс», 2017. – 576.*
4. *Головешкин В.А., Ульянов М.В. Теория рекурсии для программистов. – М.:ФИЗМАТЛИТ, 2006,-296 с.*
5. *Ульянов М.В. Ресурсно-эффективные компьютерные алгоритмы. Разработка и анализ.- М.:ФИЗМАТЛИТ, 2008, 304 с.*
6. *Филатов В.В. Лекции и практики*