|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | |
| 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:

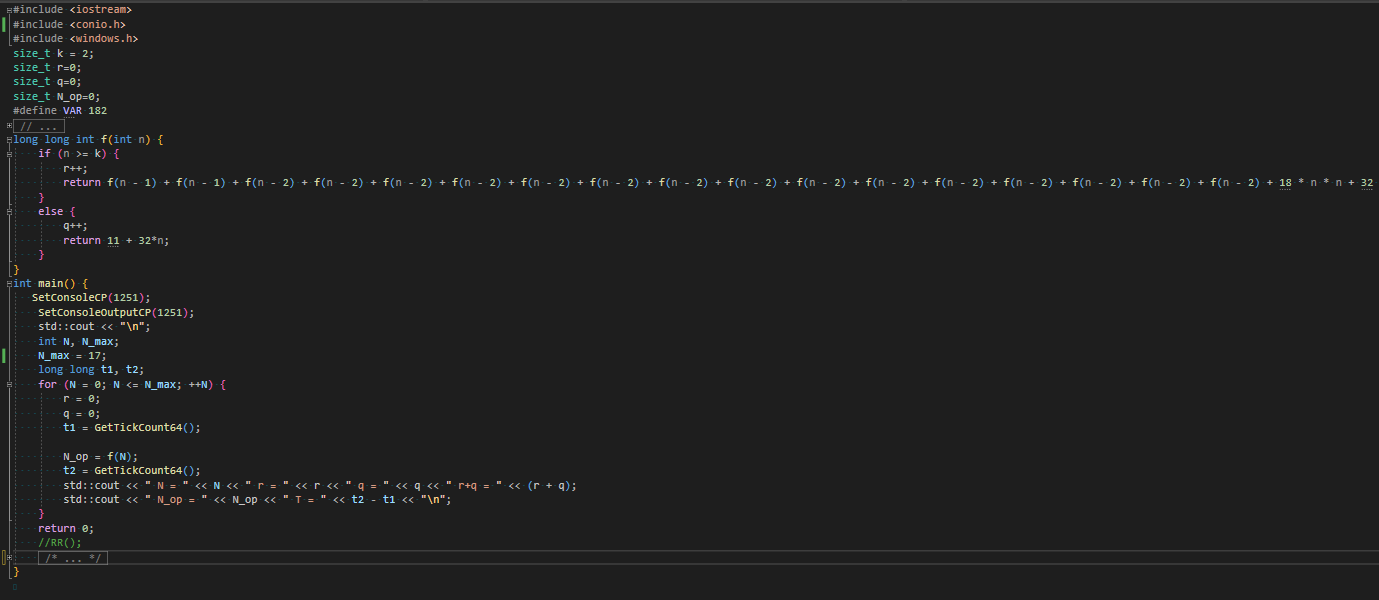
|  |
| --- |
| Анализ сложности рекурсивного алгоритма |
|  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Студент: |  | |  | 29.10.2024 |  | Хвальчев С. П.Сю |
|  | | *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.**

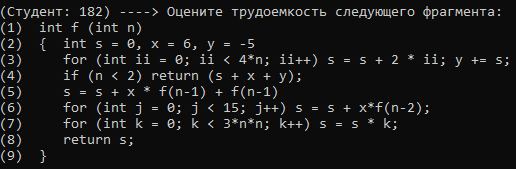


Часть 1-3

**Рекурсивная функция на языке С(С++) генерируется программой-заданием в ходе выполнения задания практической работы по теме анализ рекурсивной программы.**

**Пример.**

Пусть файл задания сформировал следующий фрагмент:



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

|  |  |
| --- | --- |
|  | (1) +1 |
| (2) +3 |
| (3) |
| (4) |
| (5) |
| (6) |
| (7) |
| (8) |

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

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

Таким образом, для указанного фрагмента функция роста имеет вид:

Или, используя рекуррентную запись обобщенного члена, имеем

при начальных значениях .

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

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

Решение линейного неоднородного рекуррентного уравнения ищется как

где – решение соответствующего линейного однородного рекуррентного уравнения;

– 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

Рассмотрим левую часть, приравнивая ее к нулю

Оно представляет собой искомое линейное однородное рекуррентное уравнение 2-го порядка.

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

или для обобщенного члена

что для линейного однородного рекуррентного уравнения с коэффициентами , соответствует

или в каноническом виде

где

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 , 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

характеристическим многочленом называется многочлен комплексной переменной

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

Для нашей задачи характеристический полином имеет вид

Корнями будут значения

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

Для удобства последующих вычислений коэффициенты обозначим греческими буквами без индексов:

Обобщенное решение нашей задачи будет иметь вид:

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

Если правая часть представима в виде

где – некоторый многочлен от аргумента *n* c коэффициентами

,

l – некоторое число (l ≠ 0),

то для линейного неоднородного рекуррентного уравнения вида

в котором – некоторые числа, , существует частное решение вида

1. , где – некоторые числа, если *l – НЕ* является корнем характеристического многочлена соответствующего линейного однородного уравнения из левой части, то есть *l* ≠ *xi*;
2. , где – некоторые числа, если *l* ≠ 0 *–* корень кратности *r* характеристического многочлена соответствующего линейного однородного уравнения из левой части, то есть *l* = *xi*.

Поскольку среди значений корней характеристического уравнения нет значения 1 (), то есть 1 (единица) не является корнем характеристического уравнения второго порядка, то можно положить *l* = 1, что позволит получить справа полином, а значение *m* ограничим значением порядка полинома из правой части неоднородного линейного рекуррентного уравнения, т.е. *m* = 2. Частное решение будем искать в виде полинома порядка *m* (*m* = 2):

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

Подставим эти значения в правую часть полученного ранее выражения

переписав его так, чтобы оно существовало для всех натуральных *n* > 0, то есть реализовав подстановку *n* ← *n*+2

:

приведем подобные слагаемые

Приравняем полученное выражение к правой части

Умножим обе чести для удобства на –1

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

Следовательно,

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

=

Найденный обобщенный член *F*(n), должен соответствовать любому значению последовательности, включая и начальные значения *F*(0) = 11 и *F*(1)=43, то есть

,

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

*.*

***Сделаем проверку.***

1. Выражение

должно удовлетворять самому рекуррентному соотношению:

Проверим. Действительно

Подставим выражения в левую часть

,

Что в точности соответствует правой части рекуррентного уравнения.

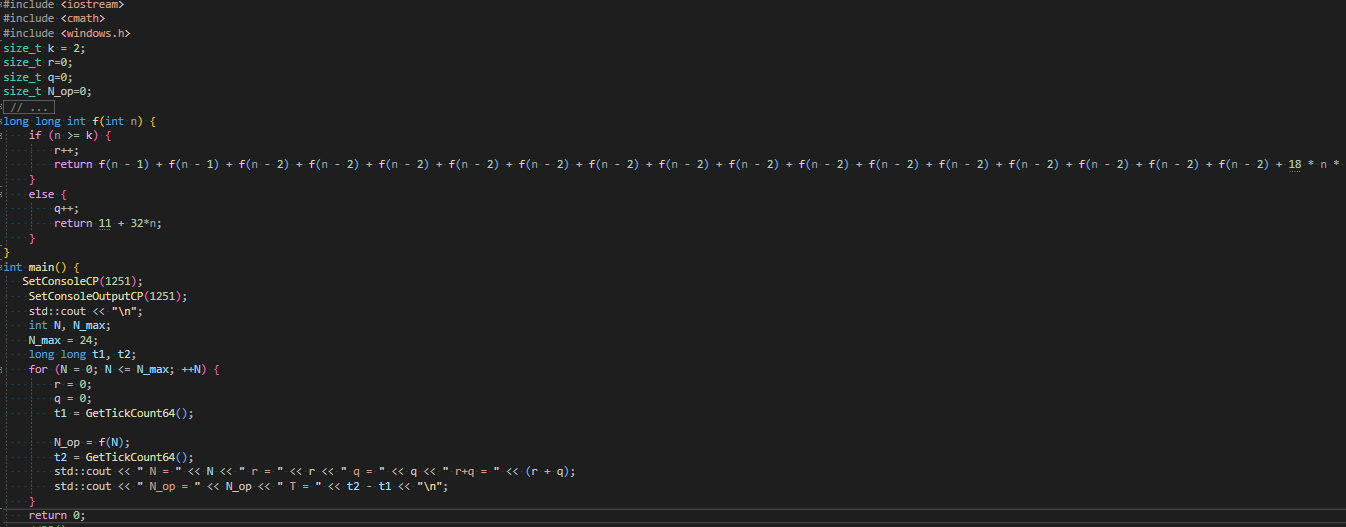
1. Проверим, что решение удовлетворяет начальным значениям

Таким образом, функция роста трудоемкости выполнения рекурсивной функции составляет

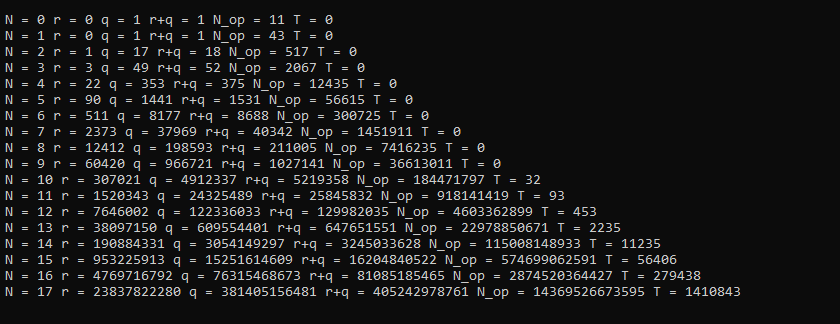
для сведения, асимптотическая оценка ее составляет

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

которую необходимо экспериментально подтвердить. Для этого внесем следующие коррективы в прилагаемый к заданию текст программы (строки 17 и 24), указав трудоемкости вызовов функции-задания из формулы для внутреннего листа и листьев дерева рекурсии.



Результат эксперимента – запуск программы, представлен на следующем скриншоте



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=10987:

Заметим, что переменные *r* и *q* корректно подсчитывают количество внутренних узлов и листьев соответственно дерева рекурсии, выражение которой указано в строке 17, при начальных значениях, указанных в строке 24. 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.

|  |  |
| --- | --- |
|  | (1) +0 |
| (2) +1 |
| (3) +0 |
| (4) +0 |
| (5) |
| (6) |
| (7) |
| (8) |

(здесь нули указаны для наглядности внесенных изменений)

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

Или, используя рекуррентную запись обобщенного члена, имеем

при начальных значениях .

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

Как уже показывалось, решение линейного неоднородного рекуррентного уравнения будем искать как сумму решения соответствующего линейного однородного рекуррентного уравнения и некоторого *частного решения* , т.е.

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

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

Решение также будем искать с помощью корней характеристического полинома вида

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

с учетом найденных корней:

Если правая часть линейного неоднородного рекуррентного соотношения представима в виде

где – некоторый многочлен от аргумента *n* c коэффициентами,

l – некоторое число (l ≠ 0),

то для линейного неоднородного рекуррентного уравнения вида

в котором – некоторые числа, , существует частное решение вида , где – некоторые числа, если *l – НЕ* является корнем характеристического многочлена соответствующего линейного однородного уравнения из левой части, то есть *l* ≠ *xi*.

Замерим, что при *l =* 1; *d* = 1; *a*, *b*…*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

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

получаем

Откуда

Найдем , поскольку должно удовлетворять начальным условиям, то

,

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

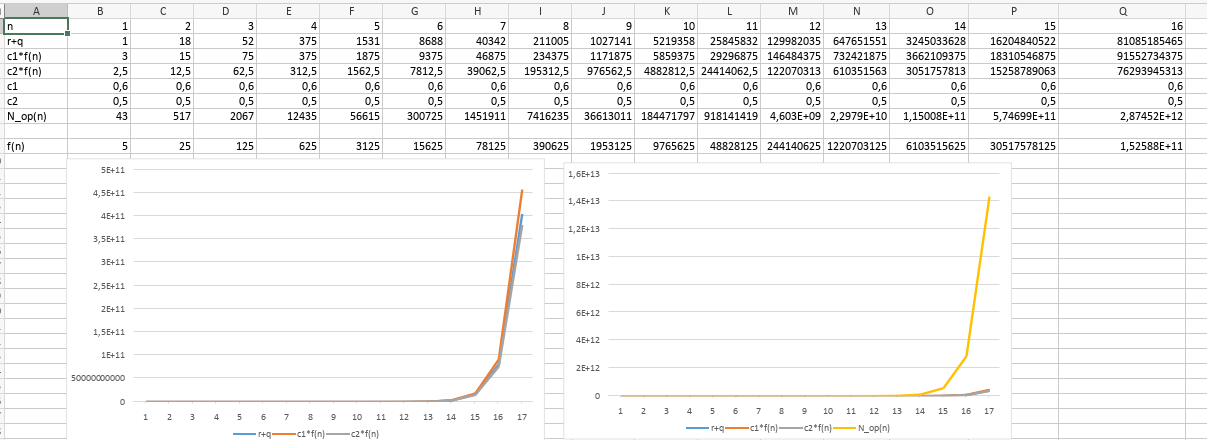
Например,

что соответствует четвертой строке скриншота результата работы

*N* = **4** *r* = 22 *q* = 353 *r*+*q* = 375**,**

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

Часть 4



f(n)=Θ()

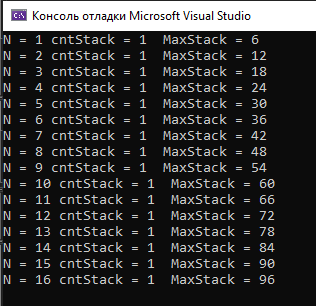
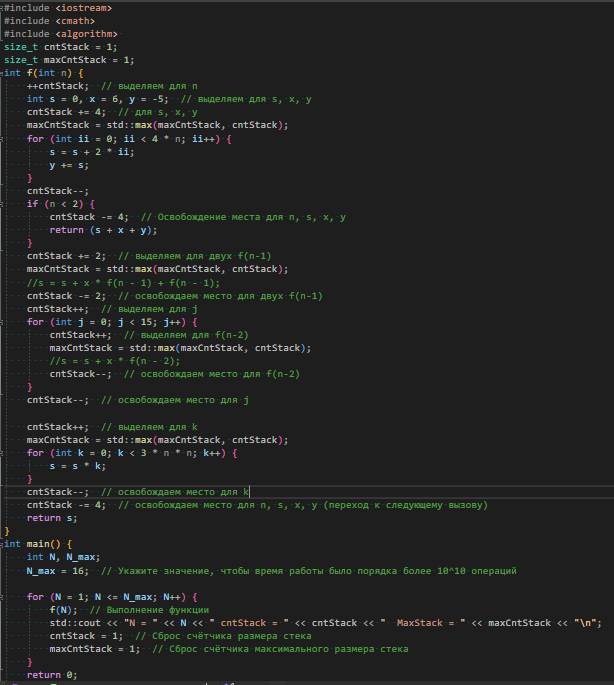
Часть 5.

В моей таблице нет значения 20 <r(N)+q(N)<40 ,поэтому я возьму максимально приближенное к данному диапазону. В моём случае это N=3, так как r(N)+q(N)=59(порядок рекурсии равен 2!). Для значения F(N-1) и F(N) изобразим дерево рекурсии.

В по ссылке будут два изображения для N=3 и N=4

Смотреть фото:https://github.com/MaEsTr0oF/Algoritm/tree/main/task2

Часть 6.



Вывод программыВыв

Формулу оценки требуемого размера стека: 6n .

Часть 7.

### **Вывод::**

В ходе выполнения данной работы я освоил методы анализа рекурсивных функций и научился применять линейные неоднородные рекуррентные уравнения для оценки трудоемкости алгоритмов. В первую очередь, мною был изучен процесс построения функции роста трудоемкости для рекурсивного алгоритма на основе его пооператорного анализа. Это позволило определить, какие операции и сколько раз будут выполняться на каждом этапе работы алгоритма.

В рамках работы был проведен полный анализ рекуррентной зависимости, включающий поиск обобщенного решения уравнения, нахождение характеристического многочлена, вычисление корней и подстановку начальных условий. С помощью данных методов удалось вывести общее решение функции трудоемкости F(n) и провести проверку полученной формулы, чтобы убедиться в правильности и соответствии всех расчетов.

Я также провел проверку с использованием экспериментальных данных, чтобы удостовериться, что расчетные значения трудоемкости для определенных n согласуются с результатами, полученными в ходе запуска программы. Это позволило мне на практике подтвердить правильность теоретического анализа и оценить асимптотическую сложность алгоритма, что является важным аспектом при изучении алгоритмической сложности программ.

**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.*