МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания № 7

Тема:

«Рекурсивные алгоритмы и их реализация»

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Боргачев Т.М.

Группа: ИНБО-10-23

Москва – 2024

СОДЕРЖАНИЕ

Цель: получить знания и практические навыки по разработке и реализации рекурсивных процессов.

1 ФОРМУЛИРОВКА ЗАДАЧИ

1.1 Задание 1

Дан массив из n элементов вещественного типа. Вычислить среднее значение всех элементов массива.

Требования к выполнению первой задачи варианта:

- приведите итерационный алгоритм решения задачи;
- реализуйте алгоритм в виде функции и отладьте его;
- определите теоретическую сложность алгоритма;
- опишите рекуррентную зависимость в решении задачи;
- реализуйте и отладьте рекурсивную функцию решения задачи;
- определите глубину рекурсии, изменяя исходные данные;
- определите сложность рекурсивного алгоритма, используя метод подстановки и дерево рекурсии;
- приведите для одного из значений схему рекурсивных вызовов;
- разработайте программу, демонстрирующую выполнение обеих функций, и покажите результаты тестирования.

1.2 Задание 2

Создание связанного стека из n элементов.

Требования к выполнению второй задачи варианта:

- рекурсивную функцию для обработки списковой структуры согласно варианту. Информационная часть узла – простого типа – целого;
- для создания списка может быть разработана простая или рекурсивная функция по желанию (в тех вариантах, где не требуется рекурсивное создание списка);
- определите глубину рекурсии
- определите теоретическую сложность алгоритма
- разработайте программу, демонстрирующую работу функций и покажите результаты тестов.

2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

2.1 Задание 1

2.1.1 Описание алгоритма – рекуррентная зависимость

Блок-схема итерационного алгоритма представлена на рис.1.

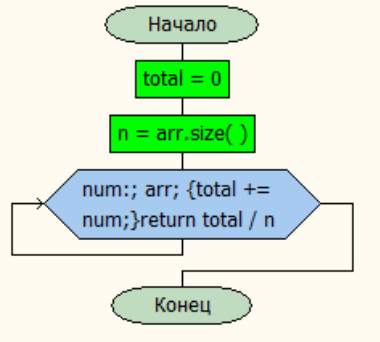

Рисунок 1 – Блок схема итерационного алгоритма

Количество выполнений операций в алгоритме зависит от количества элементов в массиве n, тогда теоретическая сложность алгоритма равна O(n).

Этот же алгоритм можно представить в виде рекуррентного, потому что существует рекуррентная зависимость результата от n. Тогда блок-схема рекурсивного алгоритма будет выглядеть в соответствии с рис.2.

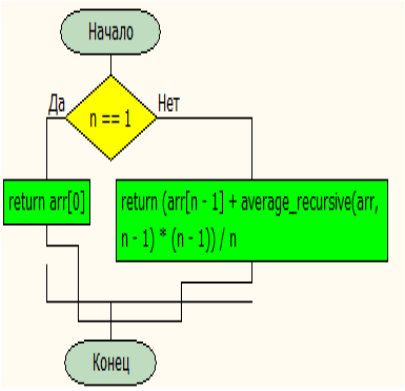

Рисунок 2 – Блок-схема рекурсивного алгоритма

Глубина рекурсии этого алгоритма равна n вызовов.

$$T(n) = a\,T(\frac{n}{c}) + bn; n > 1$$

В данном случае a=0, a <c, b=1, тогда T(n) = O(n).

Схема рекурсивных вызовов представлена на рис. 3.

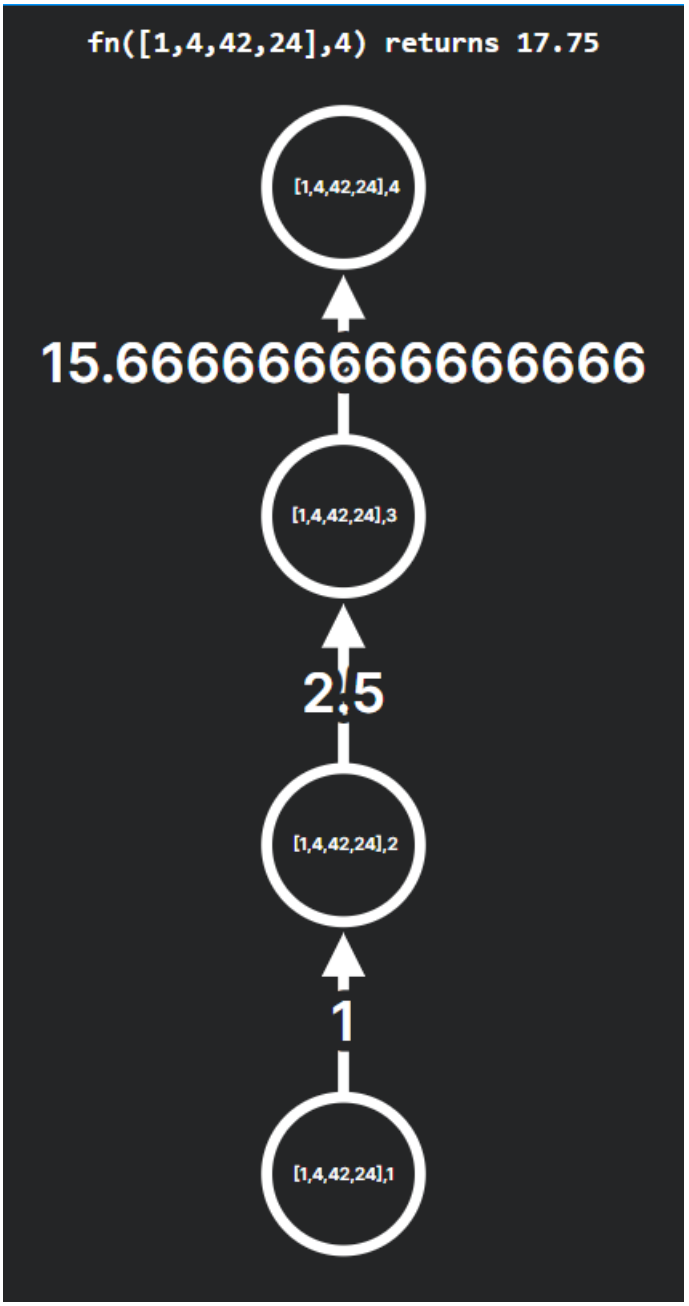

Рисунок 3 – Схема рекурсивных вызовов

2.1.2 Коды используемых функций

Код для итеративной реализации алгоритма представлен на рис. 4.

```
double average_iterative(const vector<double>& arr) {
    double total = 0;
    int n = arr.size();
    for (double num : arr) {
        total += num;
    }
    return total / n;
}
```

Рисунок 4 – Итеративная реализация алгоритма

Код для рекурсивной реализации алгоритма представлен на рис. 5.

```
// Рекурсивное решение
double average_recursive(const vector<double>& arr, int n) {
    if (n == 1) {
        return arr[0];
    }
    else {
        return (arr[n - 1] + average_recursive(arr, n - 1) * (n - 1)) / n;
    }
}
```

Рисунок 5 – Рекурсивная реализация алгоритма

Основной алгоритм программы для проверки работы алгоритмов представлен на рис. 6.

```
vector<double> arr = { 1.5, 2.5, 3.5, 4.5, 5.5 };
cout << "Average (iterative): " << average_iterative(arr) << endl;
cout << "Average (recursive): " << average_recursive(arr, arr.size()) << endl;
```

Рисунок 6 – Код для теста

2.1.3 Тестирование алгоритма

Результат работы обоих алгоритмов представлен на рис. 7.

```
C:\Users\borga\source\repos\
Average (iterative): 3.5
Average (recursive): 3.5
```

Рисунок 7 – Тестирование алгоритмов

2.2 Задание 2

2.2.1 Описание алгоритма – рекуррентная зависимость

Алгоритм предполагает принятие количества записи и целое число для каждой записи – информации хранения узла, в качестве входных данных. По входным данным должен строиться однонаправленный список, называемый стэком.

После задания каждого узла, алгоритм должен вывести стэк на экран.

2.2.2 Коды используемых функций

Структура данных для стэка представлена на рис. 8.

```
struct Node {
    int data;
    Node* next;
};
```

Рисунок 8 – Структура узла стэка

Код, реализующий функции добавления узла в стэк, создание и вывод стэка, представлен на рис. 9.

```
// Добавление узла в стэк
Node* push(Node* head, int data) {
    Node* new_node = new Node();
    new_node->data = data;
    new_node->next = head;
    head = new_node;
    return head;
}

// Рекурсивное создание стека
void create_stack_recursive(Node*& stack, int n) {
    if (n == 0) {
        return;
    }
    int data;
    cout << "Enter data for node: ";
    cin >> data;
    stack = push(stack, data);
    create_stack_recursive(stack, n - 1);
}

// Вывод стека
void display_stack(Node* stack) {
    while (stack != nullptr) {
        cout << stack->data << " -> ";
        stack = stack->next;
    }
    cout << "nullptr" << endl;
}
```

Рисунок 9 – Код реализации функций стэка

Основной алгоритм программы для тестирования алгоритма представлен на рис. 10.

```
int n;
cout << "Enter the number of nodes: ";
cin >> n;
Node* stack = nullptr;
create_stack_recursive(stack, n);
cout << "Stack: ";
display_stack(stack);
```

Рисунок 10 – Основной алгоритм программы для второго задания

2.2.3 Задачи задания 2

Глубина рекурсии равна количеству узлов n – вызовов.

$$T(n) = a\,T(\frac{n}{c}) + bn; n > 1$$

В данном случае a=0, a <c, b=1, тогда T(n) = O(n) – теоретическая сложность алгоритма.

2.2.4 Тестирование алгоритма

Результаты тестирования алгоритма представлены на рис. 11.

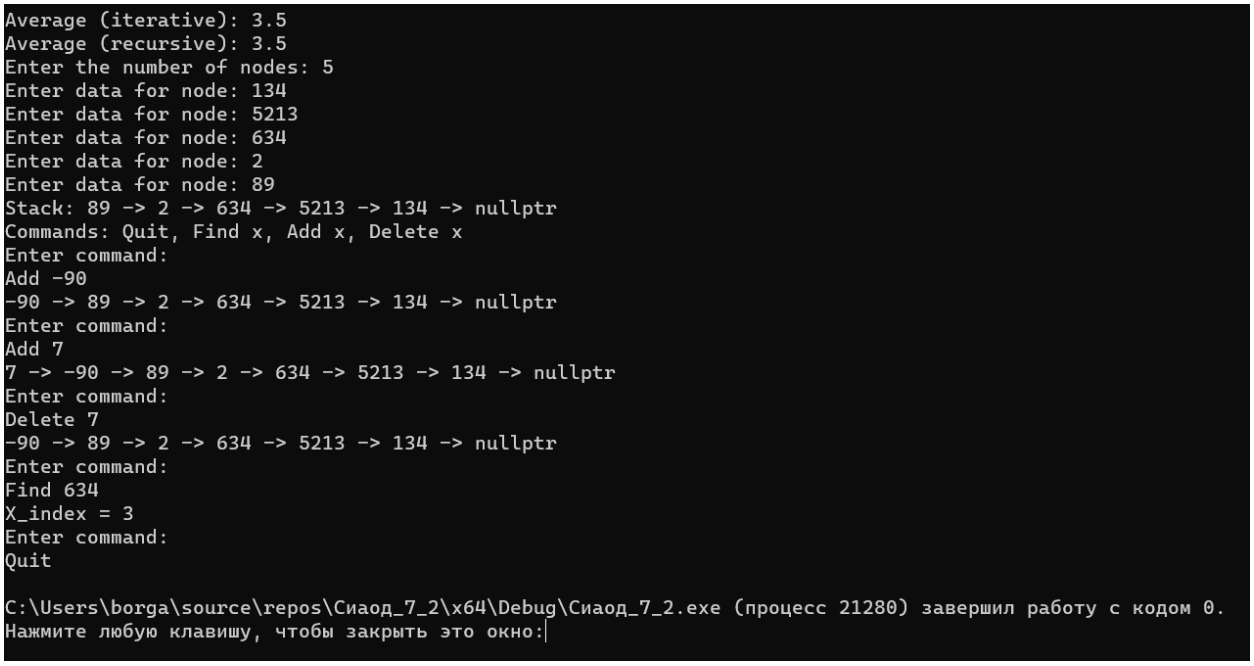

Рисунок 11 – Тестирование алгоритма

3 ВЫВОДЫ

Таким образом, все алгоритмы для обоих заданий работают корректно. В ходе выполнения задания были определены глубины рекурсии для алгоритмов, а также их теоретические сложности.

Были получены знания и практические умения работы с рекурсивными функциями.

Реализованы различные алгоритмы реализации рекурсивных проектов.

4 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Сартаков М.В., ПР-1.1 (Теоретическая сложность алгоритма)М., МИРЭА — Российский технологический университет – 12 с. - URL: https://online-edu.mirea.ru/pluginfile.php?file=%2F1042738%2Fmod_assign%2Fintroattachment%2F0%2FПР1.1%20%28Теоретическая%20сложность%20алгоритма%29.pdf&forcedownload=1 (дата обращения: 15.02.2024). - Режим доступа: Электронно-облачная система – Cloud MIREA РТУ МИРЭА. - Текст: электронный.

2. Рысин М.Л., Сартаков М.В., Туманова М.Б., Введение в структуры и алгоритмы обработки данных. Ч. 1 - учебное пособие, 2022, МИРЭА – Российский технологический университет. – 2022, 109с. – URL: file:///C:/Users/borga/Downloads/Рысин%20М.Л.%20и%20др.%20Введение%20в%20структуры%20и%20алгоритмы%20обработки%20данных.%20Ч.%201%20-%20учебное%20пособие,%202022.pdf (дата обращения: 15.02.2024). – Режим доступа: Электронно-облачная система – Cloud MIREA РТУ МИРЭА. - Текст: электронный.