Министерство науки и высшего образования Российской Федерации
Федеральное государственное автономное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

Лабораторная работа № 3
по дисциплине «Анализ алгоритмов»

Тема Графовые модели алгоритмов

Студент Куликов Н.В.

Группа ИУ7-53Б

Преподаватели Волкова Л.Л., Строганов Ю.В., Строганов Д.В.

Москва, 2025

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Цель: на материале графовых моделей алгоритмов выделить участки программ, которые могут быть исполнены параллельно.

Для достижения поставленной цели необходимо было выполнить следующие задачи:

1) описать два алгоритма (рекурсивный и нерекурсивный) решения задачи определения количества единиц в последовательности, состоящей из нулей и единиц, заканчивающейся числом 2;
2) описать реализации алгоритмов четырьмя графовыми моделями — графом управления, информационным графом, операционной историей, информационной историей;
3) указать участки каждой программы, которые могут быть исполнены параллельно, или отсутствие таковых.

1 Аналитическая часть

1.1 Рекурсия

В математике и программировании рекурсия – это метод определения или выражения функции или процедуры посредством той же функции и процедуры. Рекурсию обычно рассматривают в качестве антипода итерации. Соответственно различают два больших класса алгоритмов: итерационные и рекурсивные.

В основе итерационных алгоритмов лежит итерация – многократное повторение одних и тех же действий. Рекурсивный алгоритм – это алгоритм, определяемый через себя. В основе рекурсивных алгоритмов лежит рекурсия. [1]

В контексте программирования рекурсивной называется функция, которая вызывает сама себя. [2]

1.2 Графовые модели программ

Операционное отношение – это отношение между операторами программы, которое определяется фактом выполнения одного оператора непосредственно за другим. Оно задаёт порядок передачи управления между операторами в программе. [3]

Информационное отношение – это отношение между операторами программы, при котором один оператор использует в качестве аргументов результаты выполнения других операторов. Оно определяет зависимость по данным между операторами. [3]

Операционная история – это ориентированный граф, который строится на основе наблюдения за выполнением программы на конкретных входных данных. Вершины соответствуют каждому срабатыванию операторов, а дуги соединяют вершины в соответствии с операционными отношениями. [3]

Информационная история – это ориентированный граф, который строится на основе наблюдения за выполнением программы на конкретных входных данных. Вершины соответствуют каждому срабатыванию операторов, а дуги соединяют вершины в соответствии с информационными отношениями. [3]

Информационный граф – это модель программы, не зависящая от входных данных. Вершины графа соответствуют операторам исходной программы. Две вершины соединяются дугой, если между какими-либо срабатываниями

соответствующих операторов теоретически возможно информационное отношение. [3]

Управляющий граф – это модель программы, не зависящая от входных данных. Вершины графа соответствуют операторам исходной программы. Две вершины соединяются дугой, если текст программы допускает выполнение одного оператора непосредственно за другим, то есть возможно операционное отношение. [3]

1.3 Вывод

В аналитической части были рассмотрены понятия рекурсии, итерационного и рекурсивного алгоритмов, рекурсивной функции. Рассмотрены основные понятия графовых моделей программ.

2 Конструкторская часть

2.1 Функциональные требования

Разработать программное обеспечение для определения количества единиц в последовательности, состоящей из нулей и единиц, заканчивающейся числом 2.

Входные данные:

— пункт меню (целое число от 0 до 3),

— элементы последовательности – целые числа от 0 до 2.

Выходные данные:

Целое число – количество единиц в последовательности.

2.2 Схемы алгоритмов

На рисунках 2.1 – 2.2 представлены схемы алгоритмов.

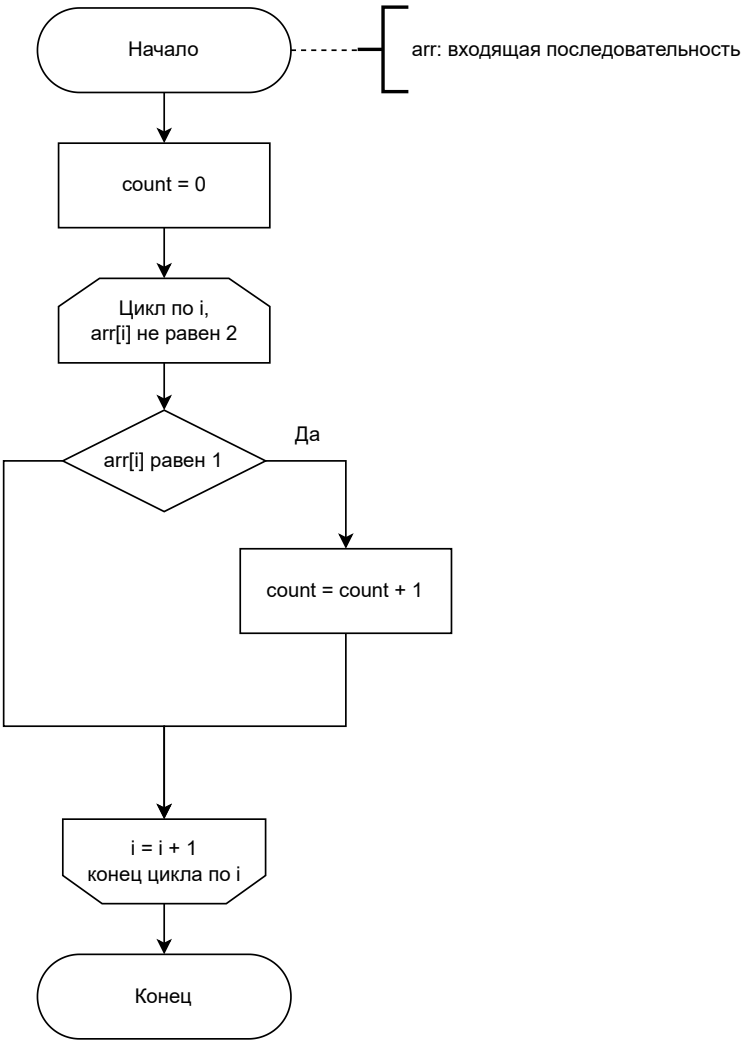

Рисунок 2.1 — Схема нерекурсивного алгоритма

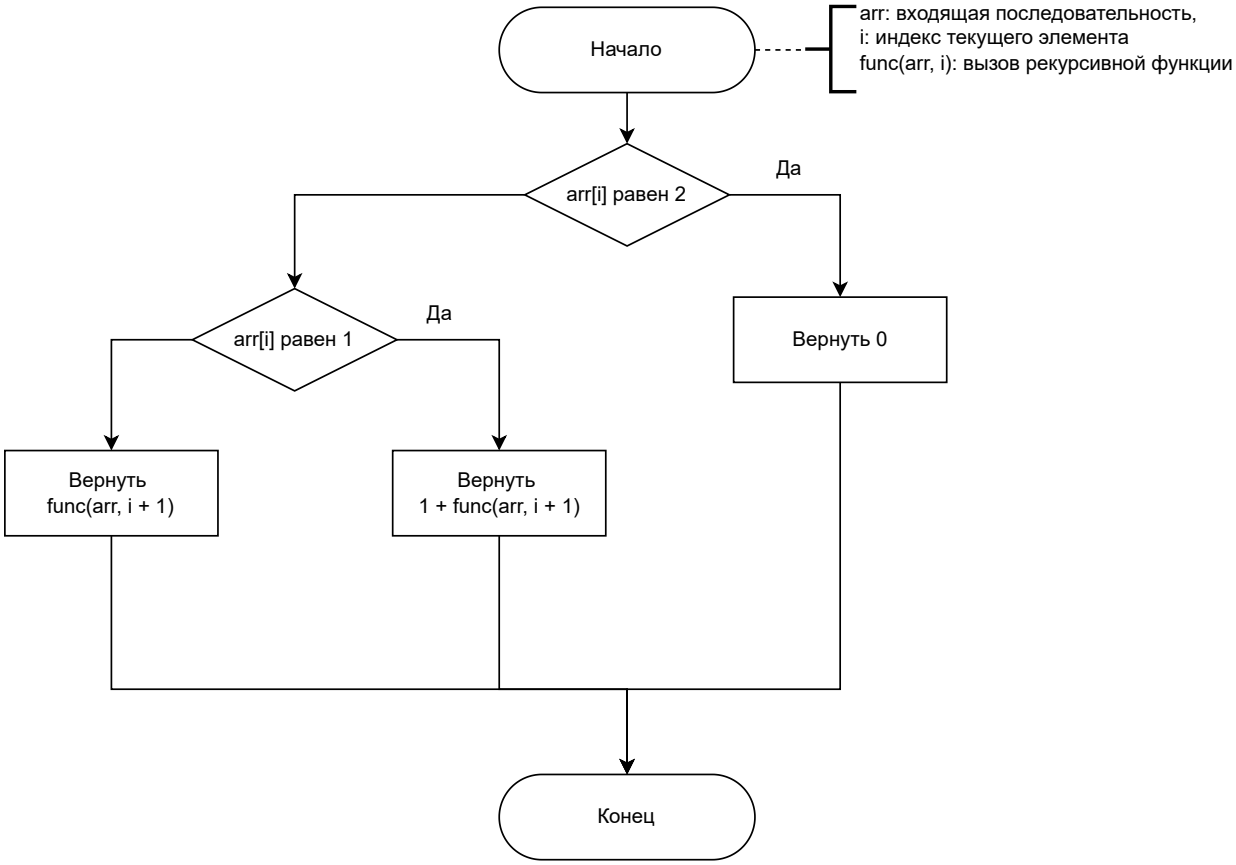

Рисунок 2.2 — Схема рекурсивного алгоритма

2.3 Вывод

В данном разделе были описаны функциональные требования к программе, построены схемы алгоритмов: нерекурсивного и рекурсивного.

3 Технологическая часть

3.1 Средства реализации

Для реализации алгоритмов был выбран язык C. Выбор обусловлен тем, что C – статически типизированный язык программирования, в нём нет сборщика мусора и имеется стандартная библиотека для замера процессорного времени.

Для замера процессорного времени использовалась функция clock() из модуля time. [4]

3.2 Реализация алгоритмов

В листингах (3.1 - 3.2) показаны реализации алгоритмов: рекурсивного и нерекурсивного.

```
size_t count_ones_recursive(const int *arr, size_t i) { // 0
  if (arr[i] == 2)                                      // 1
    return 0;                                           // 2

  if (arr[i] == 1)                                  // 3
    return 1 + count_ones_recursive(arr, i + 1); // 4

  return count_ones_recursive(arr, i + 1); // 5
}
```

Листинг 3.1 — Реализация рекурсивного алгоритма

```
size_t count_ones_iterative(const int *arr) { // 0
  size_t count = 0;                           // 1
  for (size_t i = 0; arr[i] != 2; i++)        // 2
    if (arr[i] == 1)                          // 3
      count++;                                // 4

  return count; // 5
}
```

Листинг 3.2 — Реализация нерекурсивного алгоритма

3.3 Функциональные тесты

В таблице 3.1 представлены результаты функционального тестирования реализаций: рекурсивного и нерекурсивного алгоритмов. Каждая реализация каждого алгоритма прошла тесты успешно.

Таблица 3.1 — Результаты функционального тестирования алгоритмов подсчёта единиц

Входные данные		**Результат**	
Последовательность	**Завершающий символ**	**Ожидаемый**	**Фактический**
$(1, 0, 1, 1, 0)$	2	3	3
$(1, 1, 1, 1, 1)$	2	5	5
$(0, 0, 0, 0, 0)$	2	0	0
(1)	2	1	1
(0)	2	0	0
$()$	2	0	0
(2)	2	0	0
$(1, 0, 1, 0, 1)$	-	Ошибка	Ошибка
$(1, 0, 0, 1)$	3	Ошибка	Ошибка

3.4 Вывод

В этом разделе были описаны средства реализации алгоритмов. Также были продемонстрированы листинги реализаций алгоритмов: рекурсивного и нерекурсивного. Приведены результаты функционального тестирования.

4 Исследовательская часть

4.1 Графовые модели

4.1.1 Граф управления

На рисунках 4.1 – 4.2 представлены графы управления реализаций рекурсивного и нерекурсивного алгоритмов.

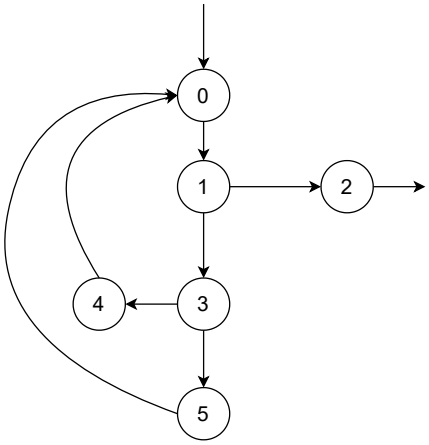

Рисунок 4.1 — Граф управления реализации рекурсивного алгоритма

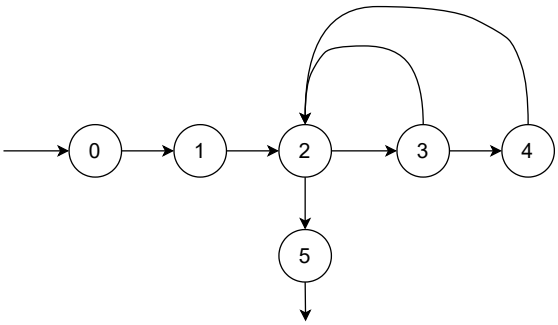

Рисунок 4.2 — Граф управления реализации нерекурсивного алгоритма

4.1.2 Информационный граф

На рисунках 4.3 – 4.4 представлены графы управления реализаций рекурсивного и нерекурсивного алгоритмов.

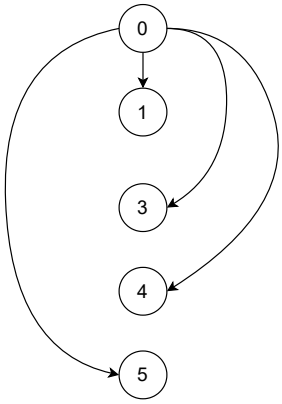

Рисунок 4.3 — Информационный граф реализации рекурсивного алгоритма

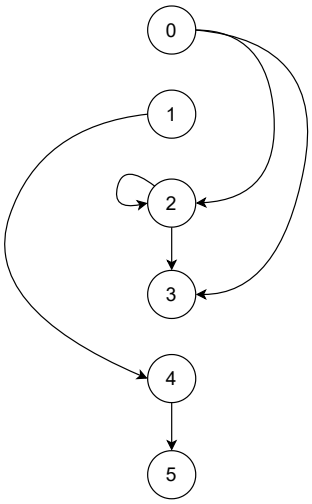

Рисунок 4.4 — Информационный граф реализации нерекурсивного алгоритма

4.1.3 Операционная история

На рисунках 4.5 – 4.6 представлены графы операционных историй реализаций рекурсивного и нерекурсивного алгоритмов при входных данных, соответствующих худшему случаю, то есть когда вся последовательность состоит из единиц. Длина последовательности $N = 5$.

4.1.4 Информационная история

На рисунках 4.7 – 4.8 представлены графы информационных историй реализаций рекурсивного и нерекурсивного алгоритмов при входных данных, соответствующих худшему случаю, то есть когда вся последовательность состоит из единиц. Длина последовательности $N = 5$.

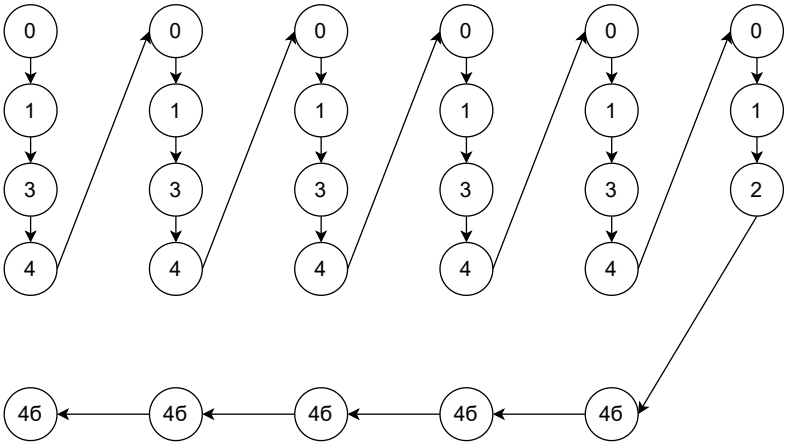

Рисунок 4.5 — Операционная история реализации рекурсивного алгоритма

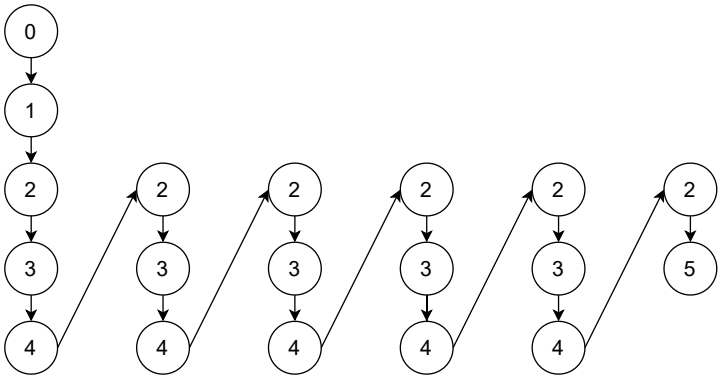

Рисунок 4.6 — Операционная история реализации нерекурсивного алгоритма

4.2 Распараллеливание программ

4.2.1 Реализация рекурсивного алгоритма

Анализируя графы 4.1 – 4.8, нельзя выделить участки реализации рекурсивного алгоритма, которые могут быть исполнены параллельно, так как между всеми операторами есть информационные и операционные отношения.

4.2.2 Реализация нерекурсивного алгоритма

Анализируя графы 4.1 – 4.8, нельзя выделить участки реализации нерекурсивного алгоритма, которые могут быть исполнены параллельно, так как между всеми операторами есть информационные и операционные отношения.

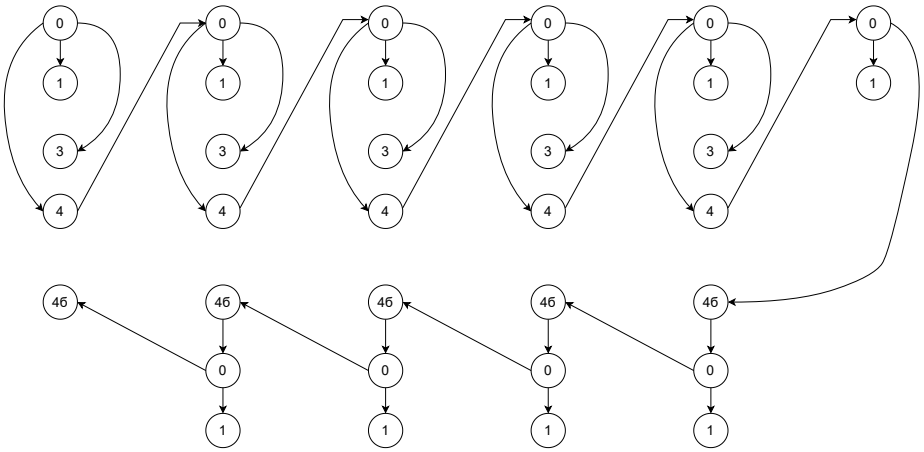

Рисунок 4.7 — Информационная история реализации рекурсивного алгоритма

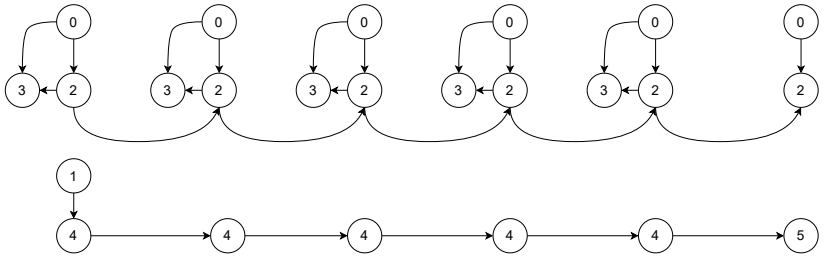

Рисунок 4.8 — Информационная история реализации нерекурсивного алгоритма

4.3 Вывод

В данном разделе были описаны реализации алгоритмов четырьмя графовыми моделями — графом управления, информационным графом, операционной историей, информационной историей. Также были указаны участки каждой программы, которые могут быть исполнены параллельно, или отсутствие таковых.

ЗАКЛЮЧЕНИЕ

В результате лабораторной работы на материале графовых моделей алгоритмов были выделены участки программ, которые могут быть исполнены параллельно.

Выполнены следующие задачи:

1) описаны два алгоритма (рекурсивный и нерекурсивный) решения задачи определения количества единиц в последовательности, состоящей из нулей и единиц, заканчивающейся числом 2;
2) описаны реализации алгоритмов четырьмя графовыми моделями — графом управления, информационным графом, операционной историей, информационной историей;
3) указаны участки каждой программы, которые могут быть исполнены параллельно, или отсутствие таковых.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Быкова В. Математические методы анализа рекурсивных алгоритмов. Журн. СФУ. Сер. Матем. и физ., том 1, выпуск 3, 2008. — С. 236.

2. Свейгарт Э. Рекурсивная книга о рекурсии. — Санкт-Петербург: Изд-во ПИТЕР, 2023. — С. 28.

3. Воеводин В. В. Параллельные вычисления. — Санкт-Петербург: Изд-во БХВ-Петербург, 2002. — С. 327–330.

4. ISO/IEC 9899:1999. 2007. — С. 7.23.2.1.