ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 4

«Поиск подмассивов размера K в массиве размера N, сумма элементов которых равна нулю»

Выполнил работу:

Баранов Владимир Александрович

Академическая группа: №J3112

Принято:

Дунаев Максим Владимирович

Санкт-Петербург

2024

1. ВВЕДЕНИЕ

Цель работы: исследовать и реализовать алгоритм поиска подмассивов фиксированного размера K = 10 в заданном массиве целых чисел, сумма элементов которых равна нулю.

Задачи:

1. Изучить необходимые теоретические аспекты для решения задачи, включая работу с массивами и методы поиска подмассивов.

2. Разработать и реализовать алгоритм для поиска всех подмассивов заданного размера K = 10 с суммой элементов равной 0.

2. ТЕОРЕТИЧЕСКАЯ ПОДГОТОВКА

В данной задаче мы работаем с массивом целых чисел. Массивы позволяют хранить данные одного типа и обращаться к каждому элементу по индексу, что упрощает реализацию алгоритмов поиска и перебора элементов.

Для выполнения работы были использованы тип массивов vector и библиотека iostream. Тип данных vector из стандартной библиотеки C++ удобен тем, что позволяет работать с динамическими массивами, размер которых может изменяться во время выполнения программы. Библиотека iostream, в свою очередь, предоставляет возможности для ввода и вывода данных, что позволяет пользователю взаимодействовать с программой.

Основной алгоритм, применяемый в данной работе, — это полный перебор. Метод полного перебора заключается в том, что мы последовательно рассматриваем все возможные подмассивы длины 10 и проверяем их на соответствие условиям задачи. В частности, для каждого подмассива проверяется, равна ли сумма его элементов нулю. Этот подход гарантирует, что мы не пропустим ни одного подмассива, который может удовлетворять условию задачи, однако может потребовать значительных вычислительных ресурсов для больших массивов.

3. РЕАЛИЗАЦИЯ

Первым шагом была проинициализирована функция findArrayWithZeroSums, которая на вход принимает массив в формате std::vector. Эта функция отвечает за поиск всех подмассивов длины 10, сумма элементов которых равна 0.

Для реализации поиска был написан алгоритм полного перебора через 10 циклов for. Каждый последующий цикл после первого начинает перебирать индексы на один больше предыдущего, таким образом мы избегаем повторную проверку одних и тех же значений массива. Если сумма значений подмассива длины 10 равна нулю, то заносим индексы данного массива в массив с результатами res.

4. ЭКСПЕРИМЕНТАЛЬНАЯ ЧАСТЬ

Рассмотрим результаты разработанного алгоритма. Для оценки производительности алгоритма была рассчитана асимптотика алгоритма, использование памяти и замерена скорость выполнения алгоритма с помощью встроенной библиотеки chrono.

Подсчет памяти производился для наихудшего случая, когда на вход подается 25 элементов. Самый большой интерес вызывает массив, содержащий результат выполнения алгоритма: структура вектора в векторе занимает 24 * 2 = 48 байт, количество всех возможных значений будет равно биноминальному коэффициенту из длины равной 10 и 25 возможных элементов, умноженных на 4 (тип int), то есть подсчитав, в итоге получим 3 268 908 байт.

Благодаря 10 вложенным циклам for итоговая асимптотика алгоритма составляет O(N^10).

Согласно требованиям моего варианта, на вход алгоритму подается до 25 элементов. Теоретическая сложность задачи составляет O(N^10) в худшем случае, что является минимальным условием задачи. Для практической оценки алгоритма была составлена таблица №1 зависимостей времени выполнения кода от размера набора входных данных. Для удобства визуального восприятия был составлен график (см. рисунок №1).

Размер входного набора	10	15	20	25	30	35	40
Время выполнения реализованного алгоритма, с	0.001	0.002	0.006	0.061	0.371	1.8	6.89
O(N^9), с	0.001	0.002	0.005	0.032	0.152	0.637	2.05
O(N^11), с	0.001	0.002	0.006	0.081	0.662	4.515	21.1

Таблица № 1 – подсчет зависимостей времени выполнения кода от размера набора входных данных.

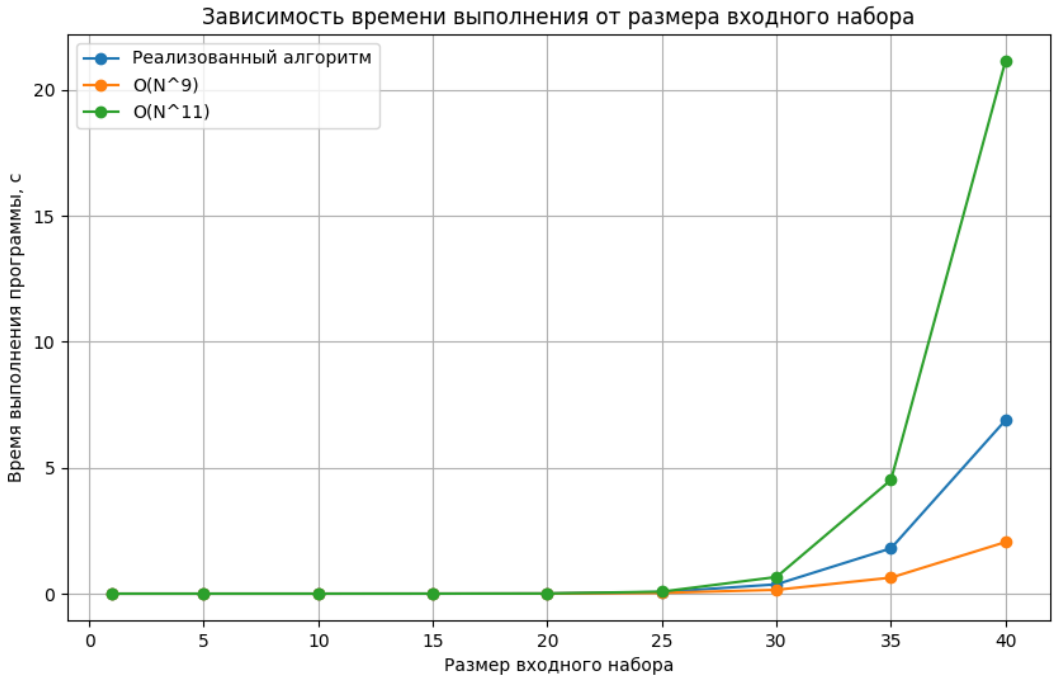

Рисунок №1 График зависимости времени выполнения от размера входного набора.

5. ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы был реализован алгоритм поиска подмассивов фиксированного размера K = 10 в заданном массиве целых чисел, сумма элементов которых равна нулю. Цель работы была достигнута с помощью тестирования алгоритма на различных входных наборах данных.

Практические результаты подтвердили оценку сложности алгоритма, которая составляет O(N^10). Время выполнения алгоритма существенно возрастает с увеличением числа элементов, что накладывает ограничения на его применение для больших массивов. Тем не менее, разработанный алгоритм гарантирует нахождение всех подмассивов, удовлетворяющих заданному условию, благодаря использованию метода полного перебора.

В рамках данной задачи была достигнута цель использования алгоритма полного перебора. Использование данного алгоритма в реальных не приведет ни к чему хорошему, для этой задачи стоит задуматься над оптимизацией.

ПРИЛОЖЕНИЕ А

Листинг алгоритма лабораторной работы №4

```
#include <iostream>
#include <vector>

std::vector<std::vector<int>>
findArrayWithZeroSums(std::vector<int>& arr) {
    int n = arr.size(); // 4 байта; O(1)
    int k = 10; // 4 байта; O(1)
    std::vector<std::vector<int>> res; // 24*2 = 48 байта на
инициализацию векторов + (4 * C(n, 10)), где C(n, 10)
максимальное количество перестановок по биному Ньютона длины 10;
O(C(n, 10))

    for (int i = 0; i < n - 9; i++) { // 4 байта
        for (int j = i + 1; j < n - 8; j++) { // 4 байта
            for (int k = j + 1; k < n - 7; k++) { // 4 байта
                for (int a = k + 1; a < n - 6; a++) { // 4 байта
                    for (int b = a + 1; b < n - 5; b++) { // 4
байта
                        for (int c = b + 1; c < n - 4; c++) { //
4 байта
                            for (int d = c + 1; d < n - 3; d++) {
// 4 байта
                                for (int e = d + 1; e < n - 2;
e++) { // 4 байта
                                    for (int f = e + 1; f < n -
1; f++) { // 4 байта
                                        for (int g = f + 1; g <
n; g++) { // 4 байта
                                            if (arr[i] + arr[j] +
arr[k] + arr[a] + arr[b] + arr[c] + arr[d] + arr[e] + arr[f] +
arr[g] == 0) { // O(1) + O(1) + O(1) + O(1) + O(1) + O(1) + O(1)
+ O(1) + O(1) + O(1) = O(1)
                                                res.push_back({i,
j, k, a, b, c, d, e, f, g}); // O(1)
                                                // Сложность
цикла O(N^10)
                                            }
                                        }
                                    }
                                }
                            }
                        }
                    }
                }
            }
        }
    }
```

```
                    }
                }
            }
        }
    }

    return res;
}

int main() {
    std::vector<int> arr = {3, -7, 4, 1, -1, 0, 5, -5, 6, -6}; //
25 * 4 = 100 байт максимум; O(1) // {3, -7, 4, 1, -1, 0, 5, -5,
6, -6, 2, -2, 0, 0, 3, -3, 7, -4, -1, 1, 8, -8, -3, 2, -2}

    std::vector<std::vector<int>> res =
findArrayWithZeroSums(arr); // 24*2 = 48 байта на инициализацию
векторов + (4 * C(n, 10)), где C(n, 10) максимальное количество
перестановок по биному Ньютона длины 10; O(1)

    for (const auto& indices : res) {
        std::cout << "[";
        for (size_t i = 0; i < indices.size(); ++i) {
            std::cout << indices[i];
            if (i < indices.size() - 1) {
                std::cout << ", ";
            }
        }
        std::cout << "]" << std::endl;
    }
    /*
    Итого по асимптотике: O(N^10)
    Итого по памяти: 4*12 + 48+(4*C(n, 10)) + 100 байт
    */
    return 0;
}
```