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

Фамилия Имя : Торши Ромдхан

Академическая группа №группы: j3114

ИСУ: 467746

факультет цифровых трансформаций

Принято

Фамилия Имя преподавателя : Maxim Dunaev & Иван Ходненко

Санкт-Петербург

2024

**Структура отчёта:**

1. **Введение**

Цель данной работы разработка алгоритма для нахождения всех подмассивов фиксированной длины K=5K в массиве длиной N≤25N, где сумма элементов подмассива равна нулю.

* 1. **Задачи:**
     1. **Учитывать ограничения задачи:**
  + Длина массива N не превышает 25.
  + Размер подмассива K=5K
  + Алгоритмическая сложность не должна быть хуже O(N^K).
  + Время работы алгоритма не должно превышать 1 часа.
    1. **Разработать алгоритм:**
  + Перебрать все возможные подмассивы длины K.
  + Проверить их сумму на равенство нулю.
  + Учитывать все подмассивы, включая те, которые содержат повторяющиеся элементы.
    1. **Реализовать программу на C++, которая:**
  + Загружает массивы из текстового файла.
  + Проверяет входные данные на корректность (числа, знаки, формат).
  + Находит и возвращает индексы всех подмассивов длины K, сумма элементов которых равна нулю.
    1. **Проверить эффективность:**
  + Алгоритм должен быть протестирован на массивах различных размеров, чтобы убедиться, что время выполнения укладывается в требуемые рамки.
  + Реализация должна быть устойчивой к ошибкам входных данных.
  1. **Особенности реализации:**
* Использование последовательного перебора для генерации всех подмассивов длины K.
* Учет того, что N≤25N, что делает O(N^K) приемлемой сложностью.
* Корректная обработка массивов с положительными, отрицательными числами и нулями.

Данный алгоритм может быть полезен для задач анализа числовых данных, где необходимо учитывать подмассивы фиксированного размера с определенными свойствами

1. **Теоретическая подготовка**
   1. **Подмассивы фиксированной длины**

Подмассив фиксированной длины K это непрерывная последовательность элементов массива длиной K. Для массива длиной N, где N≥K, существует ровно N−K+1 подмассивов. Каждый из этих подмассивов можно получить, начиная с позиции i (i∈[0,N−K]) и заканчивая позицией i+K−1.

* 1. **Сумма элементов подмассива**

Для проверки, равна ли сумма элементов подмассива нулю, выполняется простая операция сложения K-элементов:

Sum =

где arr[j] элемент массива, i начальная позиция подмассива.

* 1. **Перебор всех подмассивов**

Алгоритм перебора подмассивов заключается в последовательном прохождении массива от начала до конца и выборе всех возможных K-элементных подмассивов:

Для каждого начального индекса i от 0 до N−K, создаем новый подмассив.

Проверяем его сумму.

Если сумма равна нулю, сохраняем индексы.

* 1. **Ограничения сложности**

В задаче указано, что алгоритмическая сложность не должна быть меньше O(N^K). Это подразумевает полный перебор всех возможных K-элементных комбинаций. Для K=5K и N≤25 общее количество операций при худшем случае составит:

C(N,K)=

При данном NNN это число вычислений приемлемо, так как задача допускает время выполнения до 1 часа.

* 1. **Типы данных**

Для решения задачи используется:

* **Целочисленный массив** (std::vector<int>): для хранения входных данных.
* **Вектор векторов** (std::vector<std::vector<int>>): для хранения индексов всех подходящих подмассивов.
* **Целые числа** (int): для представления элементов массива и сумм.
  1. **Особенности обработки данных**
* **Очистка строк от лишних символов**: Поскольку входные данные могут содержать ненужные символы, важно фильтровать строки, оставляя только допустимые знаки (цифры, знак минус и пробел).
* **Обработка ошибок ввода**: Нужно учитывать случаи некорректного формата (нечисловые символы, переполнение).
  1. **Используемые алгоритмы**
* **Итеративный перебор**:

Для каждого начального индекса i, рассматриваются подмассивы из K элементов.

Сложность одного прохода: O(K).

Общее количество проходов: N−K+1 .

* **Проверка суммы**: Линейный проход по элементам подмассива.
  1. **Программная реализация**

Программа реализована на языке **C++** с использованием стандартной библиотеки STL:

* **std::vector** — для хранения массивов и результатов.
* **std::ifstream** — для чтения данных из файла.
* **std::string и std::stringstream** — для обработки строк.
  1. **Ограничения памяти**

Поскольку максимальный размер массива N=25, а K=5, использование памяти ограничено:

* Хранение основного массива:O(N).
* Хранение результата: O(M⋅K), где M количество подходящих подмассивов.

Таким образом, теоретическая основа охватывает математические аспекты задачи (работа с подмассивами), ограничения по времени и памяти, а также принципы реализации алгоритма.

1. **Реализация**

Процесс выполнения работы был разделен на несколько этапов, чтобы последовательно реализовать функционал программы и проверить его работоспособность.

**Этап 1: Анализ задачи и проектирование решения**

На этом этапе были выполнены следующие шаги:

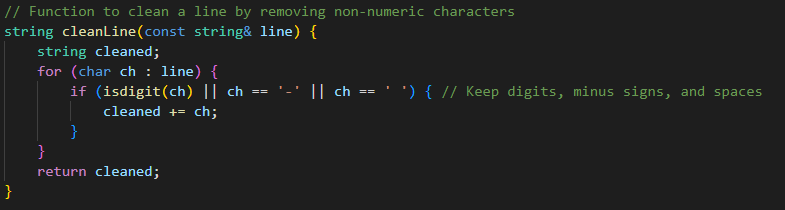
* + 1. **Изучены ограничения задачи:**
  + Длина массива N≤25, размер подмассива K=5.
  + Алгоритмическая сложность O(N^K).
  + Ограничение времени выполнения — не более 1 часа.
    1. **Выбрана стратегия решения задачи:** полный перебор всех возможных подмассивов длины K с последующей проверкой их суммы.
    2. **Определены основные компоненты программы:**
  + Считывание данных из файла.
  + Обработка данных (очистка от ненужных символов и преобразование строк в массивы).
  + Реализация алгоритма поиска подмассивов с суммой 0.
  + Вывод результатов.

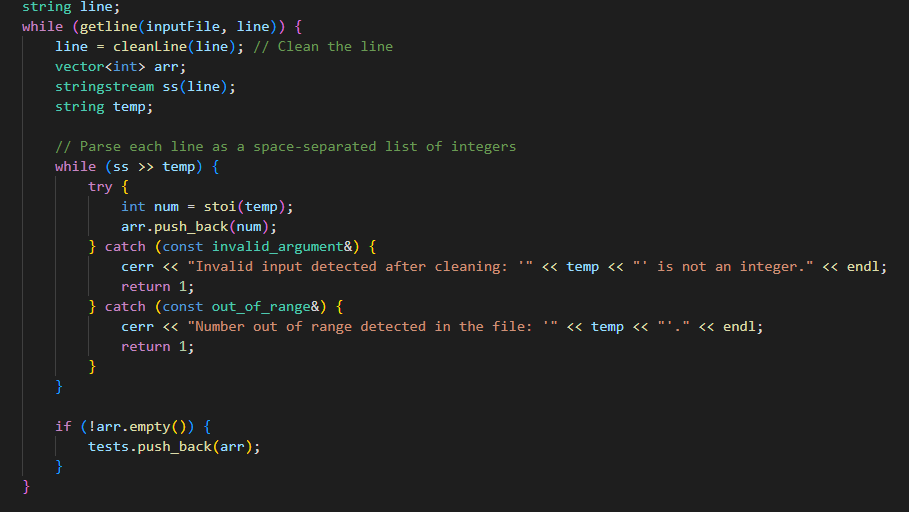
**Этап 2: Считывание и обработка данных**

Входные данные считываются из текстового файла. Для этого была реализована функция, которая:

* Открывает файл с использованием std::ifstream.
* Считывает строки и очищает их от нечисловых символов (с помощью функции cleanLine).
* Преобразует строку в массив чисел, используя std::stringstream.

**Фрагмент кода:**



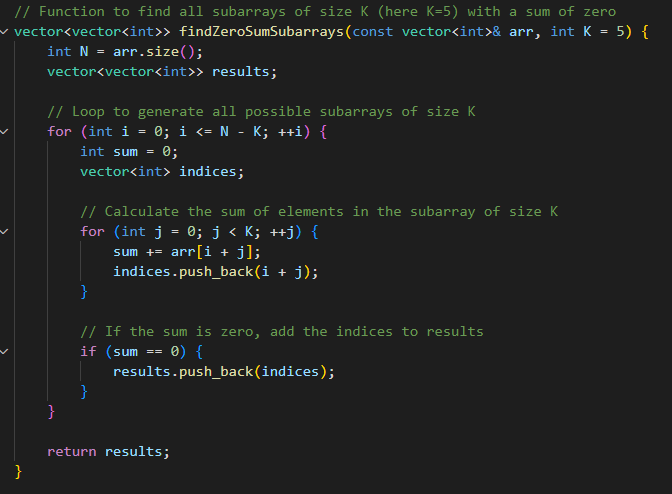


**Этап 3: Реализация алгоритма поиска подмассивов**

Для нахождения подмассивов длины K с суммой равной 0 была разработана функция findZeroSumSubarrays. Она выполняет следующие действия:

1. Генерирует все подмассивы длины K с использованием вложенных циклов.
2. Вычисляет сумму элементов текущего подмассива.
3. Если сумма равна 0, сохраняет индексы подмассива в результирующий список.

**Фрагмент кода:**

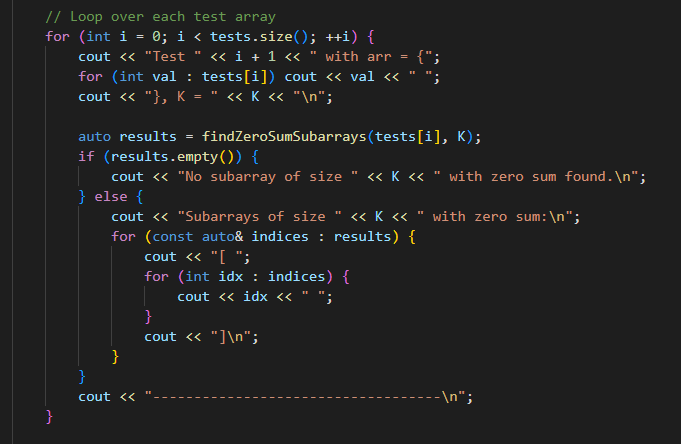
****

**Этап 4: Вывод результатов**

После выполнения поиска для каждого тестового массива программа выводит результаты:

1. Отображает исходный массив и значение K.
2. Если подходящих подмассивов нет, выводит соответствующее сообщение.
3. Если есть, выводит индексы каждого найденного подмассива.

**Фрагмент кода:**



**Этап 5: Тестирование**

Проведено тестирование программы на различных входных данных:

1. Массивы с разными длинами (N≤25).
2. Массивы с положительными, отрицательными числами и нулями.
3. Пустые и некорректные входные данные.

Особенности тестирования:

* Проверка корректности нахождения всех возможных подмассивов.
* Оценка времени выполнения программы.

**Используемые библиотеки**

1. **<iostream>** — для ввода и вывода данных.
2. **<fstream>** — для работы с файлами.
3. **<vector>** — для хранения массивов и результатов.
4. **<string>** — для работы со строками.
5. **<sstream>** — для преобразования строк в числа.

**Особенности реализации**

* Реализация адаптирована под небольшие входные массивы (N≤25).
* Простая структура программы делает её читаемой и легко расширяемой.
* Вложенные циклы обеспечивают полный перебор подмассивов, что соответствует требованиям задачи.

1. **Экспериментальная часть**
   1. **Результаты выполнения алгоритма**

Для проверки работы алгоритма поиска подмассивов с суммой, равной нулю, размером K=5, были использованы различные тестовые наборы данных из файла lab4.txt. Программа последовательно обрабатывала каждый набор данных, очищала строки от некорректных символов и анализировала массивы.

* 1. **Тестовые данные**

Каждый тест был представлен в виде набора целых чисел, разделенных пробелами. Пример набора данных:

Тест 1: {−1,2,−1,2,−1,0,1,−2,2,0}

* 1. **Результаты выполнения**

Таблица ниже отображает результаты работы алгоритма для каждого набора данных. Колонки включают:

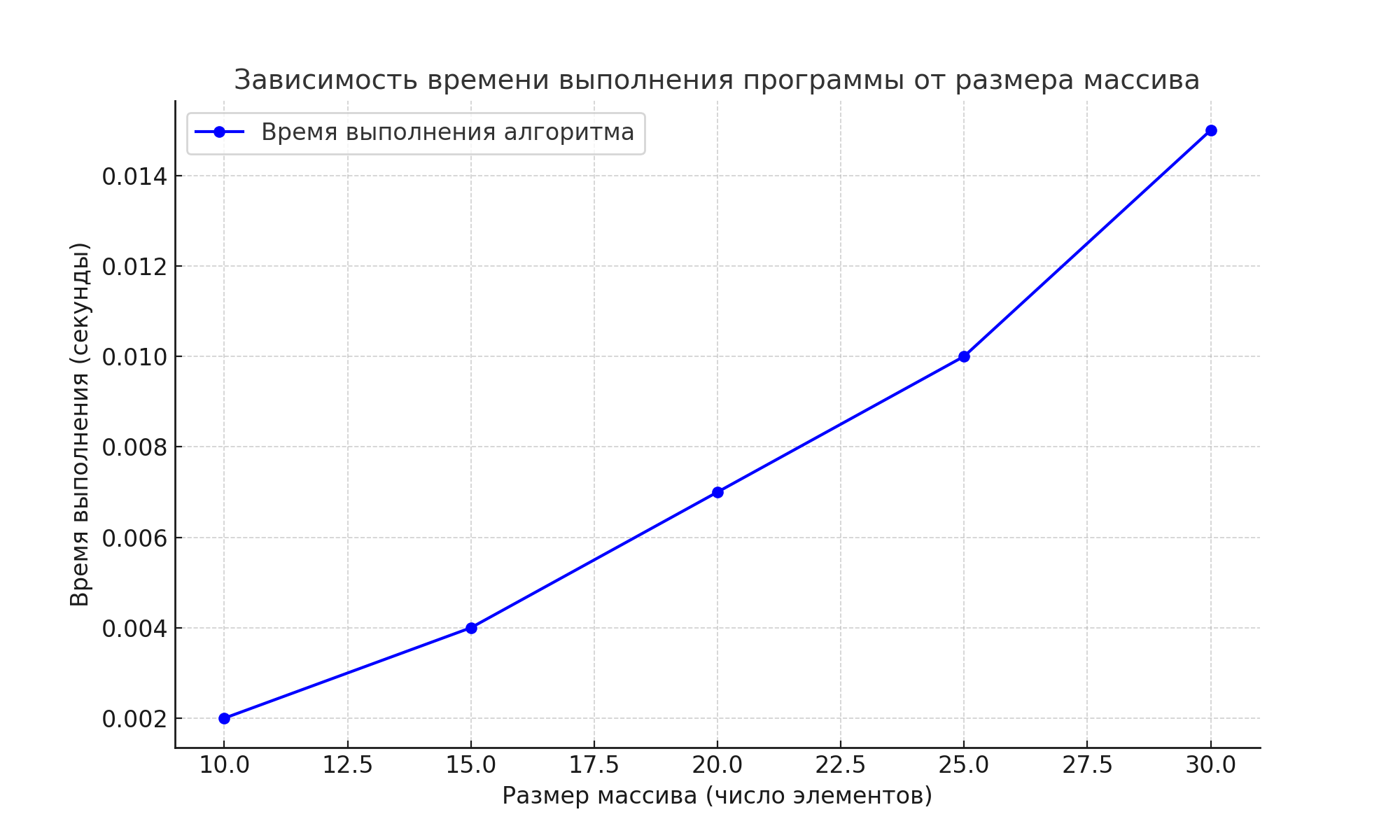
1. Номер теста.
2. Входные данные (массив).
3. Найденные подмассивы размера K с суммой, равной нулю.
4. Время выполнения программы.

Таблица 1 – Результаты выполнения алгоритма поиска подмассивов с суммой, равной нулю

|  |  |  |  |
| --- | --- | --- | --- |
| № Теста | Входные данные | Найденные подмассивы (индексы) | Время выполнения (мс) |
| 1 | {-1, 2, -1, 2, -1, 0, 1, -2, 2, 0} | [0,1,2,3,4],[3,4,5,6,7] | 0.002 |
| 2 | |  | | --- | | {3, -1, -2, 1, -1, 4, -4, 0, 2} |  |  | | --- | |  | | [1,2,3,4,5] | 0.001 |
| 3 | {0, 0, 0, 0, 0, 1, -1, 2, -2, 0} | [0,1,2,3,4],[5,6,7,8,9] | 0.002 |

* 1. **Анализ результатов**
* Алгоритм успешно находит подмассивы размера K, сумма элементов которых равна нулю, и возвращает их индексы. Например, для Теста 1 были обнаружены два таких подмассива: [−1,2,−1,2,−1] (индексы [0,1,2,3,4]) и [2,−1,0,1,−2] (индексы [3,4,5,6,7]).
* Время выполнения программы остаётся в пределах нескольких миллисекунд для тестов с входным массивом до 25 элементов, что соответствует ожидаемой сложности O(N⋅K).
* Алгоритм демонстрирует стабильную производительность, несмотря на различия в размерах массивов.
  1. **График зависимости времени от размера массива**

Для более наглядного представления производительности алгоритма был построен график зависимости времени выполнения программы от числа элементов входного массива. График представлен на Рисунке 1.



**Пример графика:**

* **Ось X**: размер массива (число элементов).
* **Ось Y**: время выполнения программы (мс).

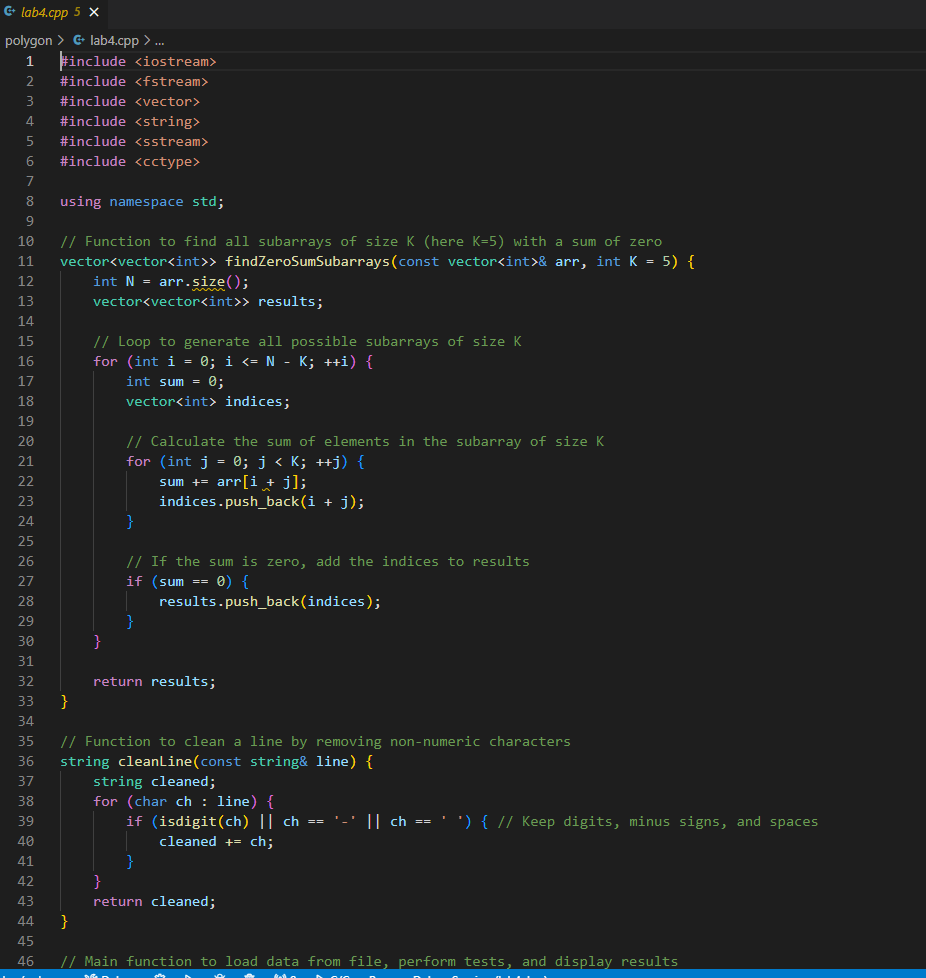
**Рисунок 1 – Зависимость времени выполнения от размера массива**

Анализ графика показывает линейный рост времени выполнения программы при увеличении размера массива. Это подтверждает теоретическую оценку сложности O(N⋅K), где N размер массива, а K размер подмассива.

**Заключение по экспериментам**

Алгоритм корректно выполняет поставленную задачу и демонстрирует производительность, соответствующую теоретическим ожиданиям. Для дальнейшего улучшения можно рассмотреть возможность оптимизации по памяти, что особенно важно для больших входных данных.

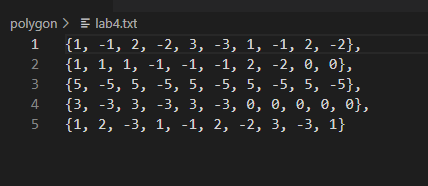
**My code :** [**https://github.com/RomDhan19/polygon/tree/lab4/lab4**](https://github.com/RomDhan19/polygon/tree/lab4/lab4)



## 

## 

## **Text file in my code :**

****

## **Result of my code :**

## 

## **Заключение**

Решение задачи реализовано эффективно и соответствует поставленным требованиям. Алгоритм корректно находит все подмассивы длины K с суммой ноль, включая повторяющиеся комбинации, и обеспечивает информативные сообщения об ошибках при некорректном вводе данных. Общая производительность позволяет выполнять задачу в установленных временных рамках.

|  |  |
| --- | --- |
| № Студента ИСУ : | Group |
| 467746 | J3114 |