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»

Выполнила работу

Автономова Ксения

Академическая группа №J3113

Принято

Иван Владимирович Ходненко

Санкт-Петербург

2024

**Структура отчёта:**

1. Введение

В данной работе поставлена задача поиска всех подмассивов размера K=4, сумма элементов которых равна нулю, в массиве N – до 25 целых чисел. Основной целью работы является разработка алгоритма, способного эффективно находить все такие подмассивы (не только из уникальных чисел), соответствующие условию задачи.

Для достижения поставленной цели мне необходимо выполнить ряд задач:

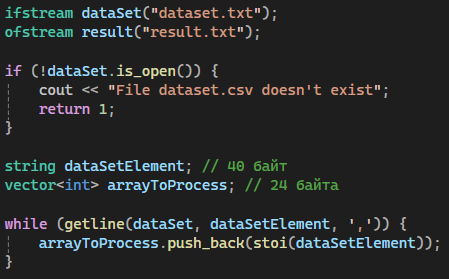
1. определить, как именно будет обрабатываться массив с данными, в котором находятся целые положительные и отрицательные числа, а также 0;
2. написать алгоритм, перебирающий всевозможные варианты подмассивов размера K и вычисляющий их сумму, при этом алгоритмическая сложность должна быть равна или больше, чем O(N^K);
3. придумать, как сохранить и вывести индексы элементов массива N для всех подходящих по условию подмассивов размера K;
4. проверить, что алгоритм соответствует поставленной временной сложности: не занимает более 1 часа при максимальном размере массива N.
5. Теоретическая подготовка

На вход программа будет получать файл dataset.txt, содержащий числа через запятую. Файл dataset.txt будет сгенерирован программой «python generate\_dataset.py --mode numeric --num-rows 25 --min-unique 20 --min-value -10 --max-value 10 --delimiter "," --output-file dataset.txt». Далее файл загружается и формируется массив неуникальных элементов с типом int. Для перебора всевозможных вариантов будут использоваться 4 вложенных цикла (кол-во циклов соответствует K). Сумма каждой последовательности будет проверяться на равенство нулю (по условию задачи), в случае положительного результата индексы элементов будут записаны в файл result.txt в следующем виде: [k1, k2, k3, k4], где k – индекс элемента в массиве. По окончанию выполнения программы в консоль будет выводиться затраченное время на выполнение циклов if (данное время не включает чтение файла и формирование массива).

1. Реализация

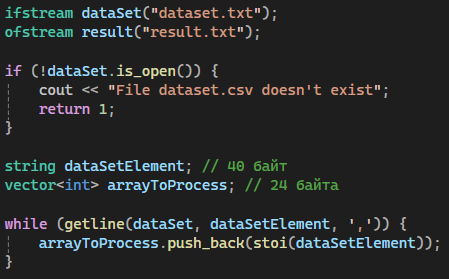
Реализацию я разделила на 3 логических блока.

Первая часть: чтение файла dataset.txt и подготовка массива с данными (изображение №1).



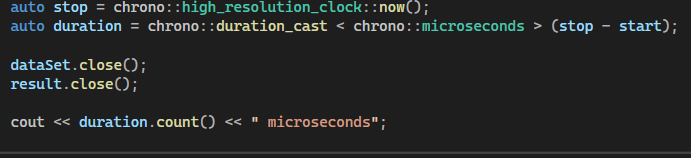
Изображение №1

Вторая часть: непосредственно алгоритм поиска элементов, удовлетворяющих условиям задачи, и вывод их в результирующий файл result.txt (изображение №2).



Изображение №2

Третья часть: вывод информации о продолжительности выполнения алгоритма и закрытие файлов dataset.txt, result.txt (изображение №3).



Изображение №3

1. Экспериментальная часть

Подсчёт по памяти (только для циклов и сложных структур) – условно 64+4\*N байт.

Подсчёт асимптотики (только для циклов и сложных структур) – O(N^4).

Согласно требованиям моей задачи, на вход к моему алгоритму подаётся до 25 элементов. Теоретически заданная сложность задачи составляет O(N^4) и более. Для тестирования алгоритма была собрана статистика, приведенная в таблице №1.

Таблица №1 - Подсчёт сложности реализованного алгоритма

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Размер входного набора | 25 | 50 | 75 | 100 | 125 | 150 |
| Время выполнения программы, с | 0,02 | 0,15 | 0,73 | 2,1 | 4,6 | 9,2 |
| O(N^4), с | 25^4 | 50^4 | 75^4 | 100^4 | 125^4 | 150^4 |

График представляющий визуально удобный формат данных из таблицы №1 представлен на изображении №4

Изображение №4 - График работы алгоритма

Я, проанализировав как работает программа на разном количестве входного набора, заметила, что время выполнения программы с массивом, состоящим из меньше 25 значений, практически незаметно увеличивается: программа стабильно выполняется меньше, чем за 1 секунду. Исходя из этого, я решила увеличить входной размер набора, чтобы посмотреть, как будет зависеть время от размера массива. Из таблицы и графика можно заметить, что при увеличении входного набора значений явно увеличивается затрата времени. Это неудивительно, ведь с увеличением количества элементов в массиве, алгоритму требуется обрабатывать значительно большее количество возможных вариантов, что приводит к росту времени выполнения моей программы.

1. Заключение

В ходе выполнения работы была успешно решена поставленная мною задача, цель была достигнута. Разработанный алгоритм, приведенный мною ранее, продемонстрировал свою эффективность, позволяя находить все подходящие подмассивы, включая массивы с повторяющимися значениями. В качестве направлений для будущего исследования можно выделить: оптимизацию алгоритма для повышения его производительности (при работе с большими массивами), исследование методов поиска подмассивов с другими условиями.

1. Приложения

ПРИЛОЖЕНИЕ А

Полный исходный код программы LAB4.cpp