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

Тиганов Вадим

Академическая группа №J3112

Принято

Ментор, Дунаев Максим

Санкт-Петербург

2024

**Структура отчёта:**

1. Введение

Цель – разработка алгоритма поиска подмассивов в массиве заданной длины по заданным условиям. Требуемая сложность – не менее O(N^K), где N – длина основного массива, K – длина каждого подмассива. (N = 25, K = 9 для моего варианта задания)

Задачи:

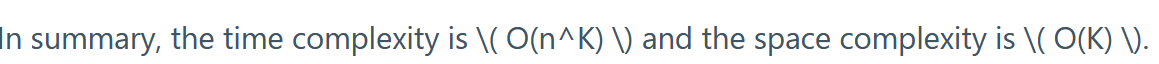
* Изучение способов эффективного (и искусственно неэффективного) поиска подмассивов с определённой суммой.
* Реализация и отладка алгоритма.
* Расчёт алгоритмической сложности программы.
* Сравнение результатов с теоретической оценкой сложности и производительности, замеры временных затрат.

1. Теоретическая подготовка

Использовал типы данных int, float, структуры данных – vector<int> - одномерный массив.

Теоретическое описание алгоритма – Алгоритм использует полный перебор (комбинаторный подход) всех возможных подмассивов длины 9 в массиве длины 25. Для этого применяется рекурсивная функция, которая поочерёдно добавляет элементы в текущий подмассив и проверяет его длину и сумму элементов. Если длина подмассива достигает 9, алгоритм подсчитывает сумму его элементов и сравнивает её с нулем. Если сумма равна нулю, подмассив добавляется в итоговый список. С учетом перебора ожидаемая алгоритмическая сложность задачи – N^K.

В том числе скриншот онлайн проверки сложности:



Ожидаемая сложность – N^K

1. Реализация

Импорт стандартных библиотек:

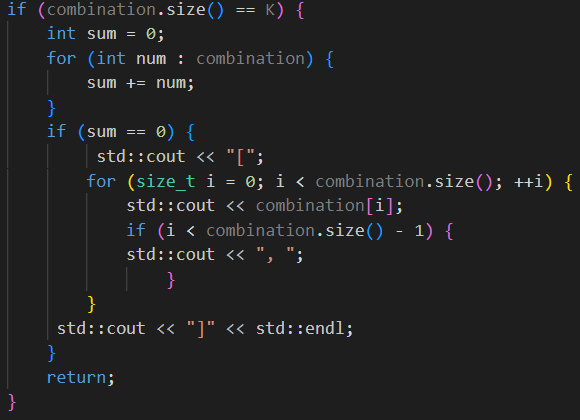
A black background with white text

Description automatically generated

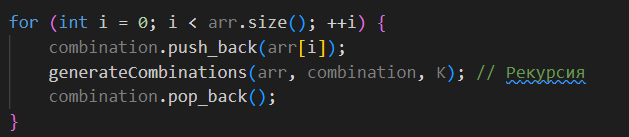
Для работы с вводом-выводом, векторами (массивами), библиотека для подсчета времени работы программы.



Функция перебора: принимает в качестве аргументов заданный массив (изначальный, с длиной 25), массив, в который будут сохраняться комбинации элементов для дальнейшей обработки, число К – длина подмассивов, которые будем искать.



Вывод: если длина подмассива равна 9, а сумма элементов равна 0, то выводим в формате [1, 2, 3, 4, 5, 6, 7, 8, 9]



Наконец, рекурсивная часть: Оставаясь в рамках главного массива, добавляем в конец текущего подмассива новый элемент, вызываем функцию для проверки, удаляем элемент и подставляем новый. И таким образом будут перебираться всевозможные комбинации из 9 элементов в заданном массиве.

A computer screen shot of a program code

Description automatically generated

В главной функции вручную задал массив длины 25, вектор для комбинаций, длину комбинаций, вызвал функцию, а также засек время выполнения с помощью std::chrono

1. Экспериментальная часть

Согласно требованиям моего варианта, на вход моему алгоритму подаётся до 25 элементов.

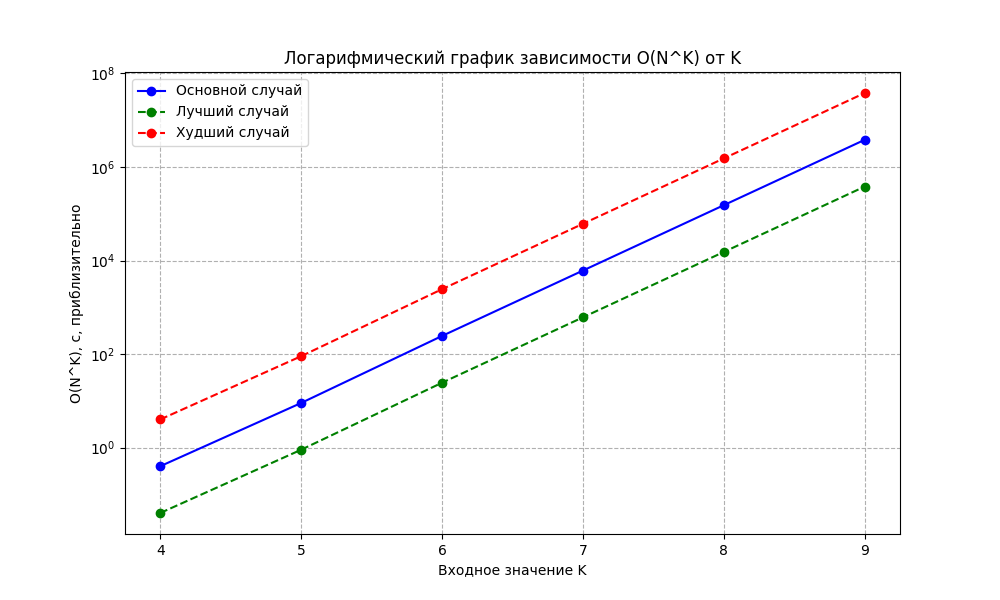
Затраты по памяти составляют 25\*4 + 9\*4 + 24 + 24 = 176 байт оперативной памяти, т.к. она затрачивается только на основной вектор и вектор подмассива, что достаточно немного. (Вес пустого вектора = 24, каждый элемент типа int занимает 4 байта)

Теоретически заданная сложность задачи составляет O(N^K) и более. Для тестирования алгоритма была собрана статистика, приведенная в таблице №1.

Таблица №1 - Подсчёт сложности реализованного алгоритма

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Размер входного набора, K | 25, 4 | 25, 5 | 25, 6 | 25, 7 | 25, 8 | 25, 9 |
| O(N^K), с, приблизительно | 0.4 | 9 | 244 | 6 103 | 152 587 | 3 814 697 |

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



Изображение №1 - График работы алгоритма

Далее необходимо привести анализ графика и таблицы.

Неожиданным стало огромное время выполнения программы, которое, однако, совпадает с расчетами вручную и оценкой с использованием нейросети Chat-GPT. Уже после 7 элементов в подмассивах с учетом сложности N^K временные затраты составляют более часа, исходя из того, что имеет 25^7 итераций, каждая из которых занимает в среднем 1 мкс. Для значений 8 и 9 время крайне стремительно возрастает, что делает невозможным подсчет опытным путем, пришлось использовать только теоретические расчеты времени.

1. Заключение

Главный вывод, сделанный мной за время и после выполнения работы – важность оптимизации времени работы программы и написания эффективных алгоритмов.

Как показала практика, даже для данного условия, что подмассивы длины 9 в массиве длины 25 должны перебираться не более часа, программа справиться не смогла ввиду высокой сложности – N^K, что было задано по условию. Программа может работать быстрее, но сложность уменьшится, что не соответствует условию.

Получил навыки и практику в теоретической оценке времязатратности алгоритма и затрат по памяти, применил на практике комбинаторные принципы для написания алгоритма.

В будущем приобретенные знания послужат для высокой осведомленности при написании оптимальных алгоритмов, потому что для больших объемов данных комбинаторные и другие сложные алгоритмические подходы не эффективны.

ПРИЛОЖЕНИЕ А

Листинг кода файла submassives.cpp

#include <iostream>

#include <vector>

#include <chrono>

void generateCombinations(const std::vector<int>& arr, std::vector<int>& combination, int K) {

        if (combination.size() == K) {

            int sum = 0;

            for (int num : combination) {

                sum += num;

            }

            if (sum == 0) {

                std::ios::sync\_with\_stdio(0);

                std::cin.tie(0); // Для быстрого вывода

                std::cout << "[";

                for (size\_t i = 0; i < combination.size(); ++i) {

                    std::cout << combination[i];

                    if (i < combination.size() - 1) {

                    std::cout << ", ";

                        }

                }

            std::cout << "]" << std::endl;

            }

            return;

        }

        for (int i = 0; i < arr.size(); ++i) {

            combination.push\_back(arr[i]);

            generateCombinations(arr, combination, K); // Рекурсия

            combination.pop\_back();

        }

}

int main() {

    std::vector<int> arr = {1, -1, 2, -2, 3, -3, 0, 4, -4, 5, -5, 6, -6, 7, -7, 8, -8, 9, -9, 10, -10, 11, -11, 12, -12};

    int K = 9; // Длина подмассива

    std::vector<int> combination;

    auto start\_time = std::chrono::high\_resolution\_clock::now();

    generateCombinations(arr, combination, K);

    auto end\_time = std::chrono::high\_resolution\_clock::now();

    std::chrono::duration<double> elapsed = end\_time - start\_time;

    std::cout << "Время выполнения: " << elapsed.count() << " секунд." << std::endl;

    return 0;

}