ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 4

«Кластеризация массива посредством полного перебора всех комбинаций значений этого массива»

Выполнил работу

Мавров Артём

Академическая группа №J3112

Принято

Ассистент (квалификационная категория "ассистент"), Дунаев Максим

Санкт-Петербург

2024

**Структура отчёта:**

1. Введение

Цель лабораторной работы — на практике изучить влияние высокой сложности NP-полных задач на время выполнение.

Задачи – изучить решение NP-полных задач, реализовать комбинаторный алгоритм, создать измерить время работы алгоритма, написать отчёт.

1. Теоретическая подготовка

Для реализации этой лабораторной работы потребовалось

* знание контейнеров vector, array
* умение работы с рекурсивными алгоритмами для решения комбинаторных задач
* знание файловых и строковых потоков
* знание библиотеки ctime, random
* знание передачи функций и элементов по ссылке
* умение считать сложность алгоритма
* умение счёта выделенной оперативной памяти

1. Реализация

* Реализация алгоритма нахождения метрики набора кластеров
* Реализация рекурсивного алгоритма перебора всех возможных
* Реализация тестирующей системы

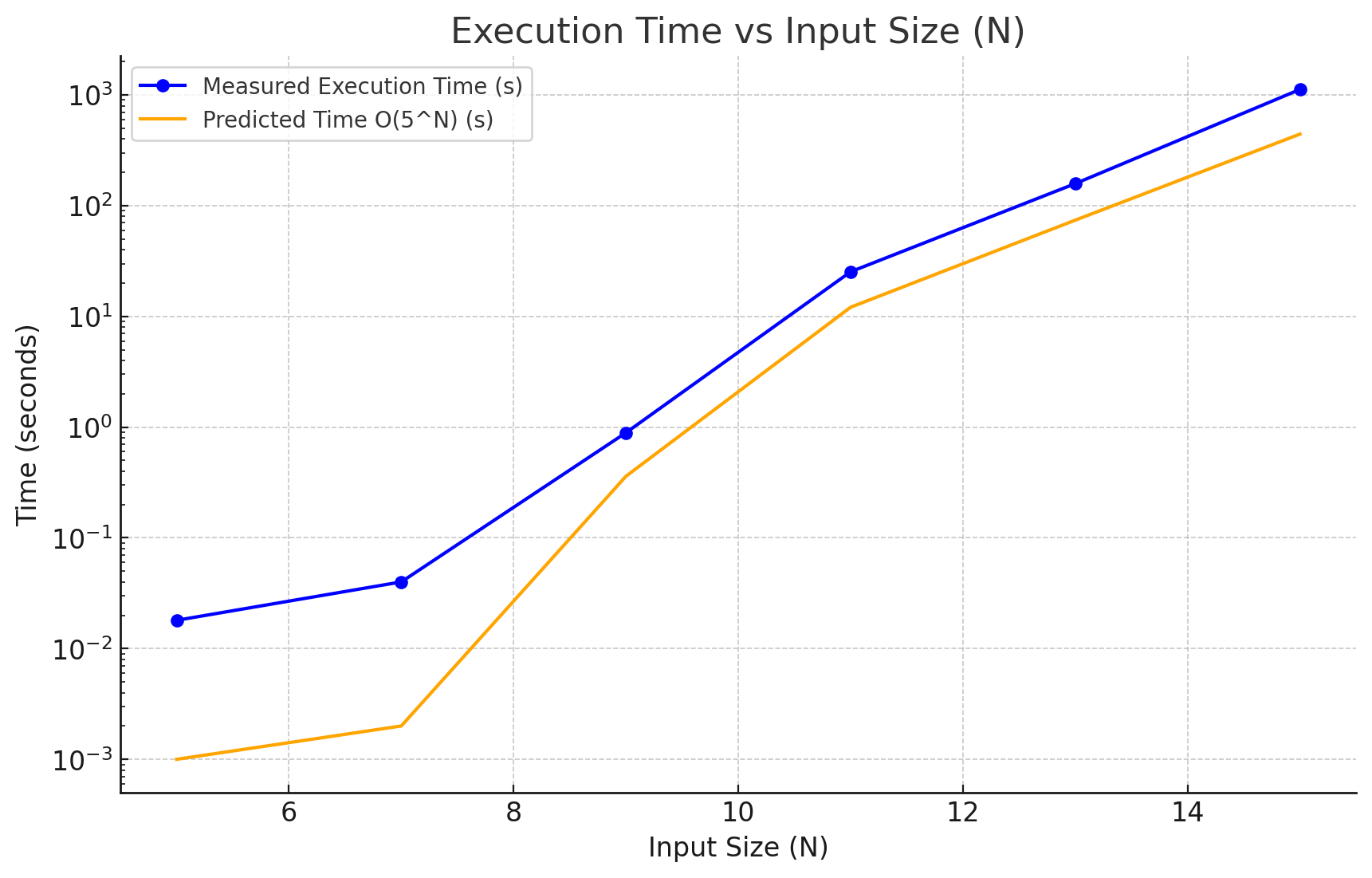
1. Экспериментальная часть

Согласно требованиям моего варианта, на вход к моему алгоритму подаётся до 25 элементов, которые кластеризуются на 5 кластеров. Теоретически заданная сложность задачи составляет O(5^N) и более, где N – количество элементов. Для тестирования алгоритма была собрана статистика, приведенная в таблице №1.

Таблица №1 - Подсчёт сложности реализованного алгоритма

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Размер входного набора | 5 | 7 | 9 | 11 | 13 | 15 |
| Время выполнения программы, с | 0.018 | 0,04 | 0,89 | 25,3 | 158,2 | 1121 |
| O(5^N), с | 0,001 | 0,002 | 0,36 | 12,1 | 74 | 442,9 |

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



Изображение №1 - График работы алгоритма

По графикам заметно, что время фактической работы программы близка к ожидаемой. Отличия вероятно происходят из-за неоптимального подсчёта метрики.

1. Заключение

В ходе выполнения работы мною был реализован алгоритм кластеризации исходного массива на 5 кластеров. Цель работы была достигнута путём полного перебора всех вариантов кластеров и отбор лучшего путём выбора наименьшей метрики. Полученные результаты приблезительно совпадают с теоретическими оценками сложности алгоритма.

В качестве дальнейших исследований можно предложить оптимизацию алгоритма путём сортировки исходного массива, чтобы отбросить заведомо плохие варианты. Так-же можно рассмотреть методы без перебора кластеров.

1. Приложения

ПРИЛОЖЕНИЕ А

Листинг кода файла lab\_4.cpp

#include <iostream>

#include <vector>

#include <array>

#include <cmath>

#include <random>

#include <ctime>

using namespace std;

double min\_metric = 1e9; // 8 байт

array<vector<double>, 5> ans; //5 \* 24 байта

int n; //4 байта

pair <short int, vector <double>> test\_maker(){

random\_device rd;

mt19937 gen(rd());

uniform\_real\_distribution<double> dist(-2147483647, 2147483646);

short int n = (rand() \* time(0)) % 16; //ограничения были понижены с 25 до 16, тк 5^25 будет выполнятся слишком долго; 2 байта

if (n < 5){

n = 5;

}

n = 9;

vector <double> elements(n); //24 байта

for (short int i = 0; i < n; i++){

double random\_element = dist(gen); //8 байт

elements[i] = random\_element;

}

return make\_pair(n, elements); // 24 + 4 байта

}

void print\_clusters(array<vector<double>, 5> clusters){

for (short int i = 0; i < 5; i++){

cout << "[";

for (short int j = 0; j < clusters[i].size(); j++){ //2 байта

if (j == clusters[i].size() - 1){

cout << clusters[i][j] << "]";

}else{

cout << clusters[i][j] << ", ";

}

}

if (i != 4){

cout << ", ";

}

}

}

double mean(const vector<double> &v){

double sum = 0; //8 байт

for (const double &x: v){ //O(N), N - длинна кластера

sum += x;

}

return sum / v.size();

}

double cluster\_metric(const vector<double> &v){

double metric = 0; //8 байт

const double current\_mean = mean(v); //8 байт

for (const double &x: v){

metric += std::abs((x - current\_mean)); //O(N), N - длинна кластера

}

return metric;

}

double clusters\_metric(const array<std::vector<double>, 5> &cluster\_distribution){

double metric = 0; //8 байт

for (const vector<double> &cluster: cluster\_distribution){ //O(N), N - длинна кластера

metric += cluster\_metric(cluster);

}

return metric;

}

void enumerate\_clusters(array<vector<double>, 5> clusters, const vector<double> elements, int index){ //O(5^N), N - длинна кластера

if (index == elements.size()) {

double size\_metric = clusters\_metric(clusters); //8 байт

if (min\_metric > size\_metric){

min\_metric = size\_metric;

ans = clusters;

}

return;

}

for (short int i = 0; i < 5; i++) { //2 байта

clusters[i].push\_back(elements[index]);

enumerate\_clusters(clusters, elements, index + 1);

clusters[i].pop\_back();

}

}

int main() {

pair <short int, vector <double>> input\_data = test\_maker();

n = input\_data.first;

vector<double> elements = input\_data.second;

array<vector<double>, 5> clusters;

enumerate\_clusters(clusters, elements, 0);

print\_clusters(ans);

return 0;

}

//всего исаользовано памяти с учётом рекурсии 228 \* 5^N, N - длинна кластера

//общая сложность O(5^N \* L), N - длинна кластера, L - сложность подсчёта метрики