МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

“Московский государственный университет геодезии и картографии”

(МИИГАИК)

Факультет геоинформатики и информационной безопасности

Кафедра геоинформационных систем и технологий

**Лабораторная работа №3**

**"Алгоритмы сортировки"**

Проверил: Выполнил:

Лебедев Евгений Денисович Студент группы: 2024-ФГИИБ-ПИ-1б

Центнер Валерий Викторович

Москва 2025

**Вариант 28**

Формулировка задания:

Я создаю код для 3 алгоритмов:  
 Сортировка пузырьком,

Поразрядная сортировка,

Быстрая сортировка.

Также, я проведу асимптотическую оценку каждого из них и укажу замеры работы алгоритмов в виде графиков и таблиц.

Ссылка на GitHub репозиторий с файлами:

<https://github.com/guguker/InfAlgoLebedev/tree/main/second_semestr/homework%203>

Блок-схема каждого из алгоритмов и листинг их кодов:

1. Пузырьковая сортировка  
  
Блок-схема:  
Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.  
  
Листинг кода:  
#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

#include <chrono> // добавили chrono для замеров времени

using namespace std;

using namespace std::chrono;

// так называемый пузырёк сортировка

void bubbleSort(vector<int>& arr) {

int n = arr.size();

for (int i = 0; i < n - 1; i++) {

bool swapped = false; // оптимизация: если ничего не поменялось — массив уже отсортирован

for (int j = 0; j < n - 1 - i; j++) {

if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]);

swapped = true;

}

}

if (!swapped) break;

}

}

// выводим массив

void printFirstElements(const vector<int>& arr, int count = 20) {

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

cout << arr[i] << " ";

}

cout << "\n";

}

int main() {

srand(time(0));

int N = 1000000; // количество случайных чисел

int RANGE = 100; // диапазон значений

vector<int> arr;

arr.reserve(N); // резервируем память заранее

// массив из N случайных чисел от 1 до RANGE

for (int i = 0; i < N; ++i) {

arr.push\_back(rand() % RANGE + 1);

}

cout << "Original array (first 20 elements):\n";

printFirstElements(arr);

// начинаем замер времени

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

bubbleSort(arr);

// заканчиваем замер времени

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

auto duration = duration\_cast<milliseconds>(end - start);

cout << "Sorted array (first 20 elements):\n";

printFirstElements(arr);

cout << "Time taken by bubbleSort: " << duration.count() << " milliseconds" << endl;

return 0;

}

2. Поразрядная сортировка  
  
Блок-схема:  
Изображение выглядит как текст, снимок экрана, диаграмма, дизайн

Контент, сгенерированный ИИ, может содержать ошибки.  
  
Листинг кода:  
#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

#include <chrono> // добавили chrono для замеров времени

using namespace std;

using namespace std::chrono;

// объединяем корзины прям как КРОЛИК:)

// если увидели это - напишите об этом в ответном письме :)

vector<int> combineBuckets(const vector<vector<int>>& buckets) {

vector<int> combinedArray;

for (const auto& bucket : buckets) {

combinedArray.insert(combinedArray.end(), bucket.begin(), bucket.end());

}

return combinedArray;

}

// сортируем по разряду

void sortByDigit(vector<int>& arr, int digitPlace) {

vector<vector<int>> buckets(10);

for (int num : arr) {

int bucketIndex = (num / digitPlace) % 10;

buckets[bucketIndex].push\_back(num);

}

arr = combineBuckets(buckets);

}

// так называемая рэйдикс сортировка

void radixSort(vector<int>& arr) {

int maxVal = \*max\_element(arr.begin(), arr.end());

int digitPlace = 1;

while (maxVal / digitPlace > 0) {

sortByDigit(arr, digitPlace);

digitPlace \*= 10;

}

}

// выводим первые 20 элементов массива для нАГлЯднОСти

void printFirstElements(const vector<int>& arr) {

for (int i = 0; i < 20 && i < arr.size(); ++i) {

cout << arr[i] << " ";

}

cout << endl;

}

int main() {

srand(time(0));

int N = 1000000000; // <- количество элементов массива

int RANGE = 100; // <- диапазон значений

vector<int> arr;

arr.reserve(N); // резервируем память заранее

// массив из N случайных чисел от 1 до RANGE

for (int i = 0; i < N; ++i) {

arr.push\_back(rand() % RANGE + 1);

}

cout << "Original array (first 20 elements):\n";

printFirstElements(arr);

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

radixSort(arr);

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

auto duration = duration\_cast<milliseconds>(end - start);

cout << "Sorted array (first 20 elements):\n";

printFirstElements(arr);

cout << "Time taken by radixSort: " << duration.count() << " milliseconds" << endl;

return 0;

}

3. Быстрая сортировка  
  
Блок-схема:  
Изображение выглядит как текст, снимок экрана, диаграмма, План

Контент, сгенерированный ИИ, может содержать ошибки.

Листинг кода:  
#include <iostream>

#include <vector>

#include <cstdlib>

#include <ctime>

#include <chrono>

using namespace std;

using namespace std::chrono;

// так называемый квиксорт который делит на больше/меньше/равно

void quickSort3Way(vector<int>& arr, int low, int high) {

if (low >= high) return;

// выбираем случайный pivot

int randomIndex = low + rand() % (high - low + 1);

int pivot = arr[randomIndex];

swap(arr[randomIndex], arr[low]);

// начинаем разделение

int lt = low;

int gt = high;

int i = low + 1;

while (i <= gt) {

if (arr[i] < pivot) {

swap(arr[i], arr[lt]);

i++;

lt++;

}

else if (arr[i] > pivot) {

swap(arr[i], arr[gt]);

gt--;

}

else {

i++;

}

}

// рекурсивно сортируем < pivot и > pivot части

quickSort3Way(arr, low, lt - 1);

quickSort3Way(arr, gt + 1, high);

}

// выводим массив

void printArray(const vector<int>& arr) {

for (int num : arr) {

cout << num << " ";

}

cout << endl;

}

int main() {

srand(time(0));

int N = 1000000000; // количество случайных чисел

int RANGE = N; // диапазон значений

vector<int> arr;

arr.reserve(N); // резервируем память заранее

// массив из N случайных чисел от 1 до RANGE

for (int i = 0; i < N; ++i) {

arr.push\_back(rand() % RANGE + 1);

}

cout << "Original array (first 20 elements):\n";

for (int i = 0; i < 20 && i < arr.size(); ++i) {

cout << arr[i] << " ";

}

cout << "\n";

// начинаем замер времени

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

quickSort3Way(arr, 0, arr.size() - 1);

// заканчиваем замер времени

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

auto duration = duration\_cast<milliseconds>(end - start);

cout << "Sorted array (first 20 elements):\n";

// для НАгЛЯдНОсти выводим 20 элементов

for (int i = 0; i < 20 && i < arr.size(); ++i) {

cout << arr[i] << " ";

}

cout << "\n";

cout << "Time taken by quickSort3Way: " << duration.count() << " milliseconds" << endl;

return 0;

}

Примеры входных / выходных данных:

1. Пузырьковая сортировка  
Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.  
  
  
  
  
2. Поразрядная сортировка  
Изображение выглядит как текст, снимок экрана, Шрифт

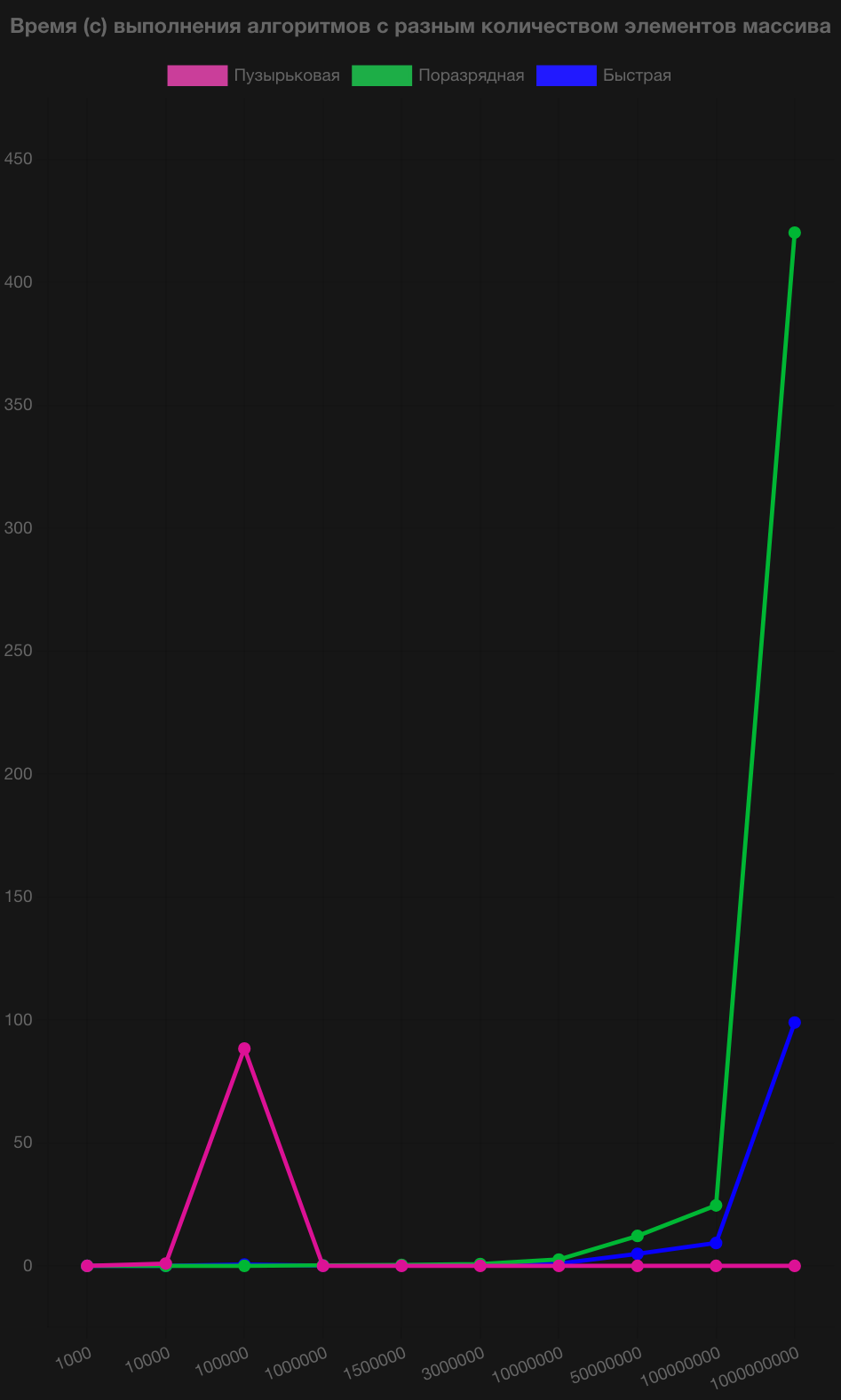
Контент, сгенерированный ИИ, может содержать ошибки.  
  
3. Быстрая сортировка  
Изображение выглядит как текст, снимок экрана, Шрифт

Контент, сгенерированный ИИ, может содержать ошибки.

Асимптотическая оценка алгоритмов:

|  |  |  |  |
| --- | --- | --- | --- |
| Алгоритм | Лучший случай | Средний случай | Худший случай |
| Пузырьковая | O(n) | O(n^2) | O(n^2) |
| Поразрядная | O(d \* n) | O(d \* n) | O(d \* n) |
| Быстрая | O(n \* log n) | O(n \* log n) | O(n^2) |

Замеры на моём ноутбуке:



Уточнение :  
Пузырьковая сортировка в целом НЕ считалась начиная с 1.000.000 элементов массива ибо в таком случае считалось бы несколько часов, что ну слишком долго))