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

Журавлев Андрей Дмитриевич

Москва 2025

**Вариант 9**

Формулировка задания:

Я создаю код для 3 алгоритмов:  
 Поразрядная сортировка

Поразрядная 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/Duhaporostoy/algoritmizatsiya>

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 <utility>

void bubbleSort(std::vector<int>& arr) {

int n = arr.size();

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

bool swapped = false;

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

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

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

swapped = true;

}

}

if (!swapped) break;

}

}

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

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

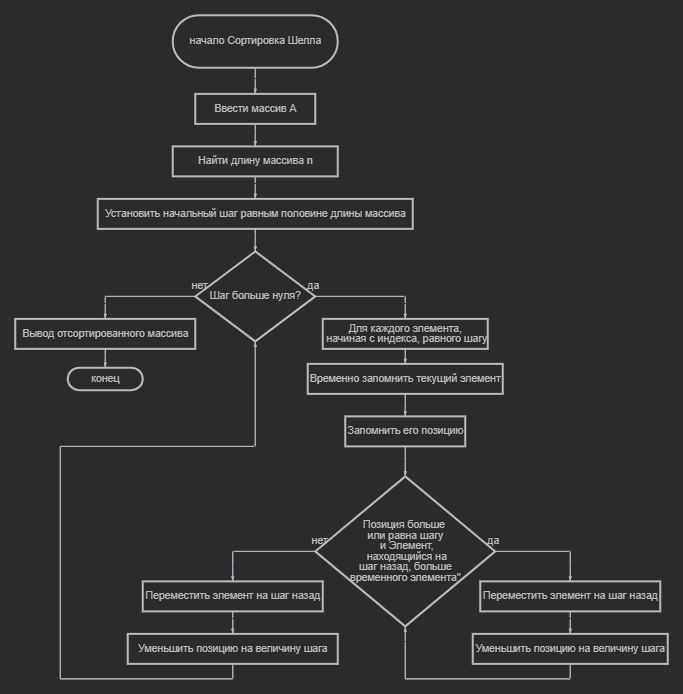
std::cout << arr[i] << " ";

}

std::cout << "\n";

}

2. Сортировка Шелла  
  
Блок-схема:



Листинг кода:  
// shell\_sort.cpp

#include <iostream>

#include <vector>

void shellSort(std::vector<int>& arr) {

int n = arr.size();

int step = n / 2;

while (step > 0) {

for (int i = step; i < n; ++i) {

int temp = arr[i];

int j = i;

while (j >= step && arr[j - step] > temp) {

arr[j] = arr[j - step];

j -= step;

}

arr[j] = temp;

}

step /= 2;

}

}

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

for (int x : arr) {

std::cout << x << " ";

}

std::cout << std::endl;

}

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>

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

if (low >= high) return;

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

int pivot = arr[randomIndex];

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

int lt = low;

int gt = high;

int i = low + 1;

while (i <= gt) {

if (arr[i] < pivot) {

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

i++;

lt++;

}

else if (arr[i] > pivot) {

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

gt--;

}

else {

i++;

}

}

quickSort3Way(arr, low, lt - 1);

quickSort3Way(arr, gt + 1, high);

}

void printQuickArray(const std::vector<int>& arr) {

for (int num : arr) {

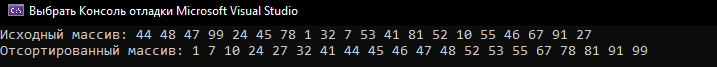
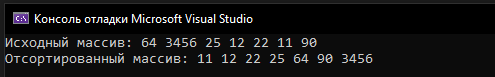
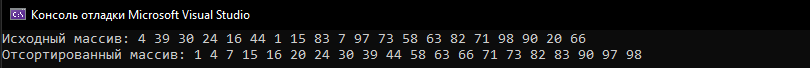
std::cout << num << " ";

}

std::cout << std::endl;

}

Примеры входных / выходных данных:

1. Пузырьковая сортировка  
  
  
  
  
  
2. Поразрядная сортировка  
  
  
3. Быстрая сортировка  


Асимптотическая оценка алгоритмов:

где:

a≥1 — количество подзадач,

b>1 — фактор уменьшения размера задачи,

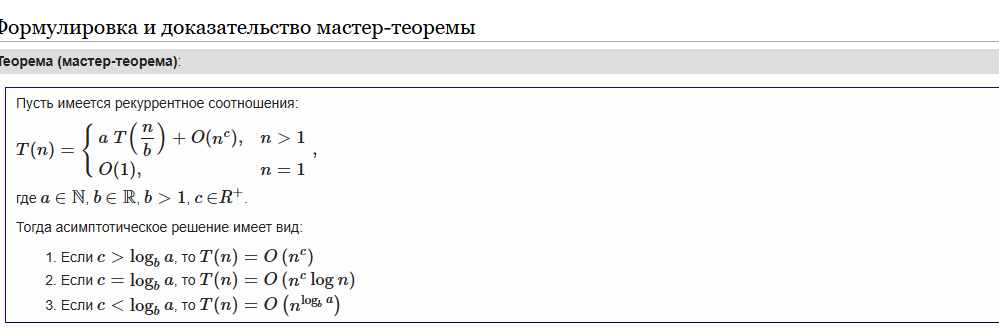
f(n) — стоимость деления задачи и объединения результатов.

Для быстрой сортировки можно рассмотреть 3 случия:

Лучший случай, это когда опорный элемент делит массив ровно пополам:

* a=2 — две подзадачи
* b=2 — размер уменьшается в 2 раза
* f(n)= (n) — проход по массиву для разделения

критический показатель:



C совподает с критическим показателям, значит подходит 2 пункт

Средний случай:

Можно привести схожесть интегралов:

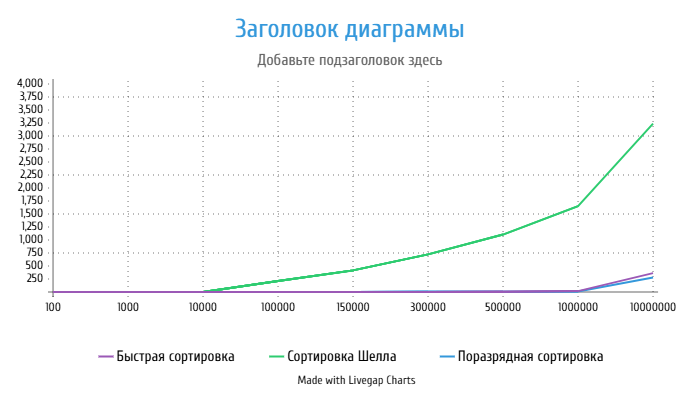
Другой случий где опорный элемент всегда минимальный или максимальный

Это не подходит под форму мастер-теоремы, так как не делится на фиксированное число подзадач

Если решить это рекурсию вручную, то получим

|  |  |  |  |
| --- | --- | --- | --- |
| Алгоритм | Лучший случай | Средний случай | Худший случай |
| Пузырьковая | O(n) | O(n^2) | O(n^2) |
| Шелла | O(d \* n) | O(n log²n) | O(d \* n) |
| Быстрая | O(n \* log n) | O(n \* log n) | O(n^2) |

Замеры на моём компъютере в секундах:



Уточнение :