**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ**



**МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**Факультет Информационных технологий**

***Кафедра Информатики и информационных технологий***

**направление подготовки**

**09.03.02 «Информационные системы и технологии»**

**ЛАБОРАТОРНАЯ РАБОТА № 18**

**Дисциплина:**

Основы алгоритмизации и программирования

**Тема:**

"Вычислительная сложность алгоритмов"

**Выполнил(а): студент(ка) группы 211-7210**

**Салов Д.К.**

(Фамилия И.О.)

**Дата, подпись** 23.12.21

(Дата) (Подпись)

**Проверил: \_\_*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_\_\_\_\_\_\_***

(Фамилия И.О., степень, звание) **(Оценка)[[1]](#endnote-1)**

**Дата, подпись** \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  ***\_\_\_\_\_\_\_\_\_\_\_***

(Дата) (Подпись)

**Замечания: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Москва**

**2021**

**Цель:**  Получить практические навыки анализа сложности алгоритмов.

**Задания:**

Необходимо выполнить и оформить описание следующих пунктов:

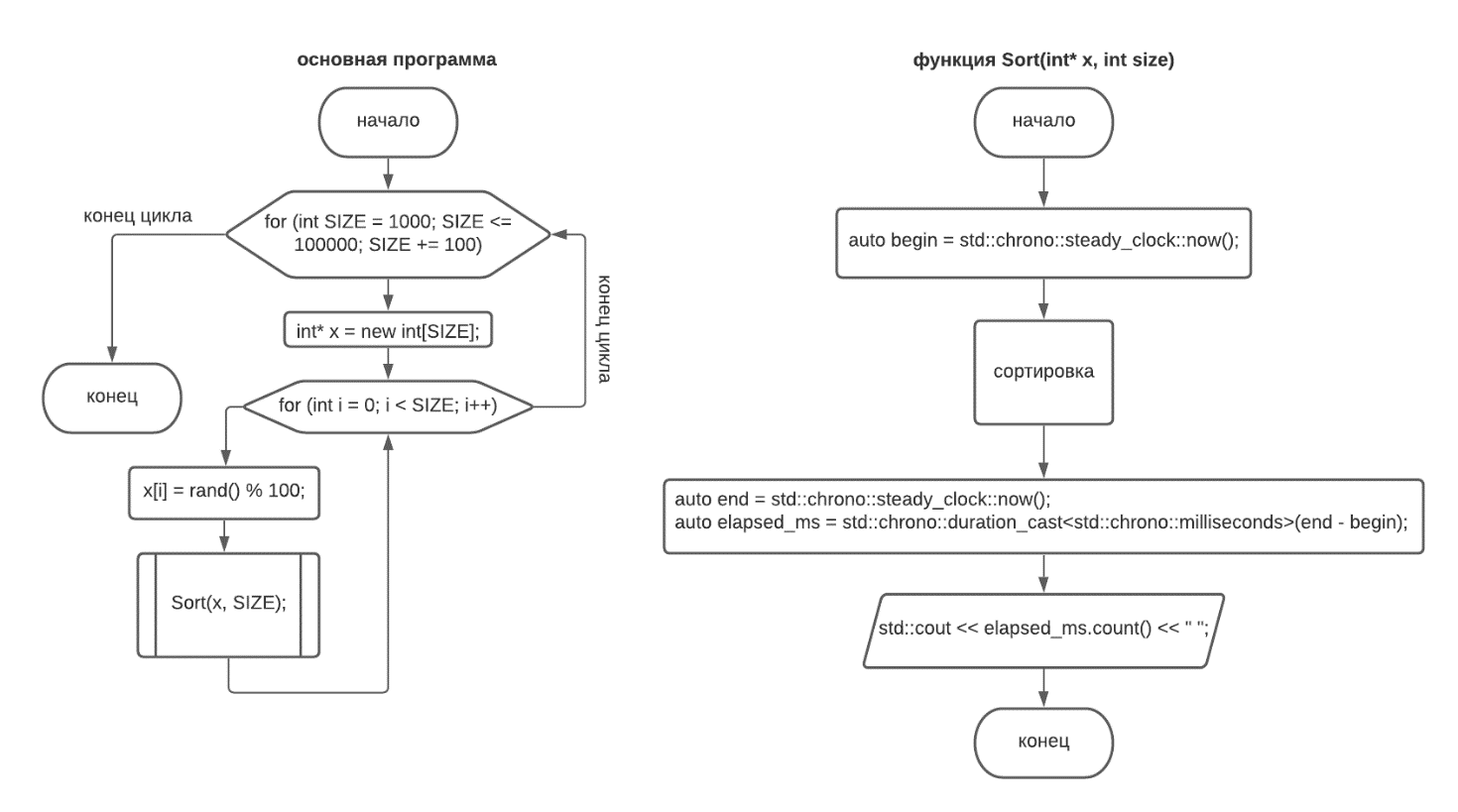
1. Представить ранее рассмотренные алгоритмы (лабораторные 2-17)

2. Выполнить анализ сложности каждого из алгоритмов.

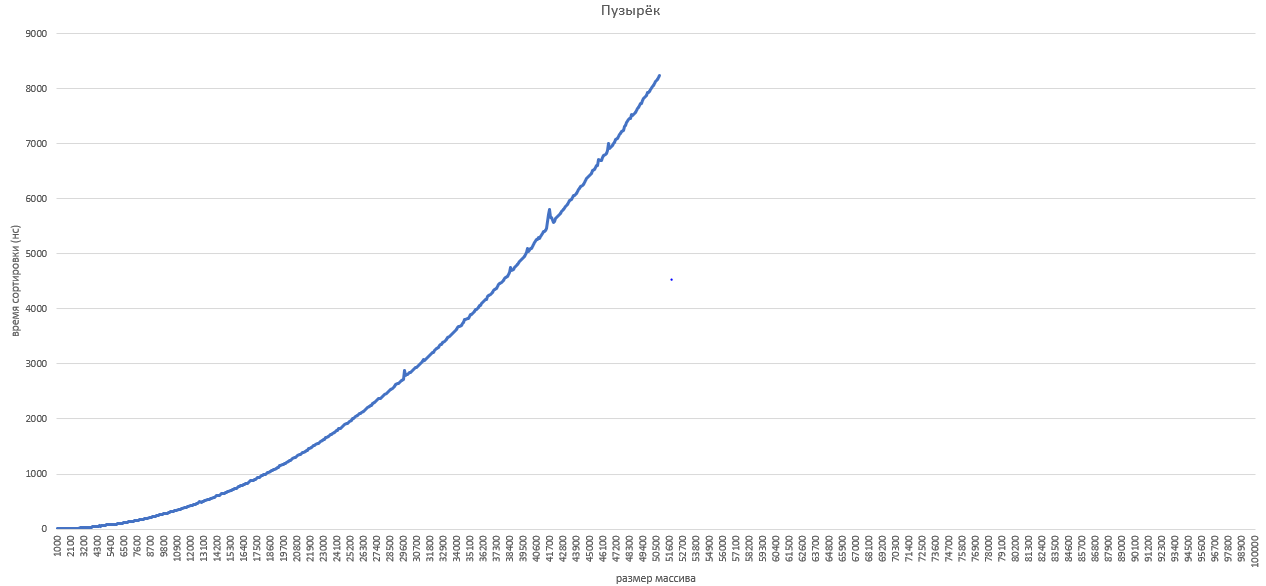
**Методика измерений:**

По логике из следующей блок-схемы пишем программы для каждого варианта сортировки. Разница будет только в самом сортировщике. Программа выдаёт через пробел время сортировки массива в миллисекундах. Результат копируем из консоли в MS Excel и строим график зависимости времени сортировки от размера массива.

**Блок-схемы:**



**Сортировка «Пузырьком»:**

*Время в миллисекундах, а не в наносекундах. На графике опечатка*

*Дальше сортировка занимала слишком много времени, я остановил программу.*

#include <chrono>

#include <iostream>

#include <ctime>

using namespace std;

void bubbleSort(int\* x, int size) { // собсна пузырёчек

auto begin = std::chrono::steady\_clock::now();

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

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

if (x[n] > x[n + 1]) {

int y = x[n];

x[n] = x[n + 1];

x[n + 1] = y;

}

}

}

auto end = std::chrono::steady\_clock::now();

auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin);

std::cout << elapsed\_ms.count() << " "; //вывод времени сортировки

}

int main() {

for (int SIZE = 1000; SIZE <= 100000; SIZE += 100) {

int\* x = new int[SIZE];

for (int i = 0; i < SIZE; i++) { //заполнение массива

x[i] = rand() % 100;

//cout << x[i];

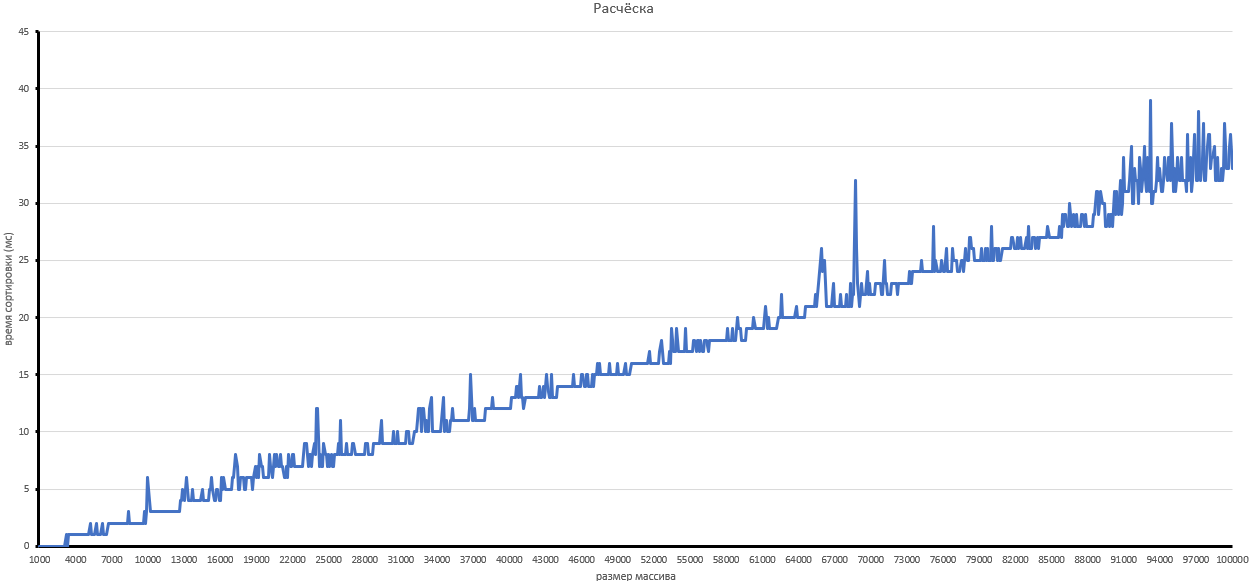
}

bubbleSort(x, SIZE);

}

}

**Сортировка «расчёской»:**



#include <iostream>

#include <chrono>

#include <iostream>

#include <ctime>

using namespace std;

void Sort(int\* x, int size) { // собсна расчёска

auto begin = std::chrono::steady\_clock::now();

constexpr double factor = 1.2473309;

int step = size - 1;

while (step >= 1) {

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

if (x[i] > x[i + step]) {

std::swap(x[i], x[i + step]);

}

}

step /= factor;

}

auto end = std::chrono::steady\_clock::now();

auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin);

std::cout << elapsed\_ms.count() << " ";

}

int main() {

for (int SIZE = 1000; SIZE <= 100000; SIZE += 100) {

int\* x = new int[SIZE];

for (int i = 0; i < SIZE; i++) { //заполнение массива

x[i] = rand() % 100;

//cout << x[i];

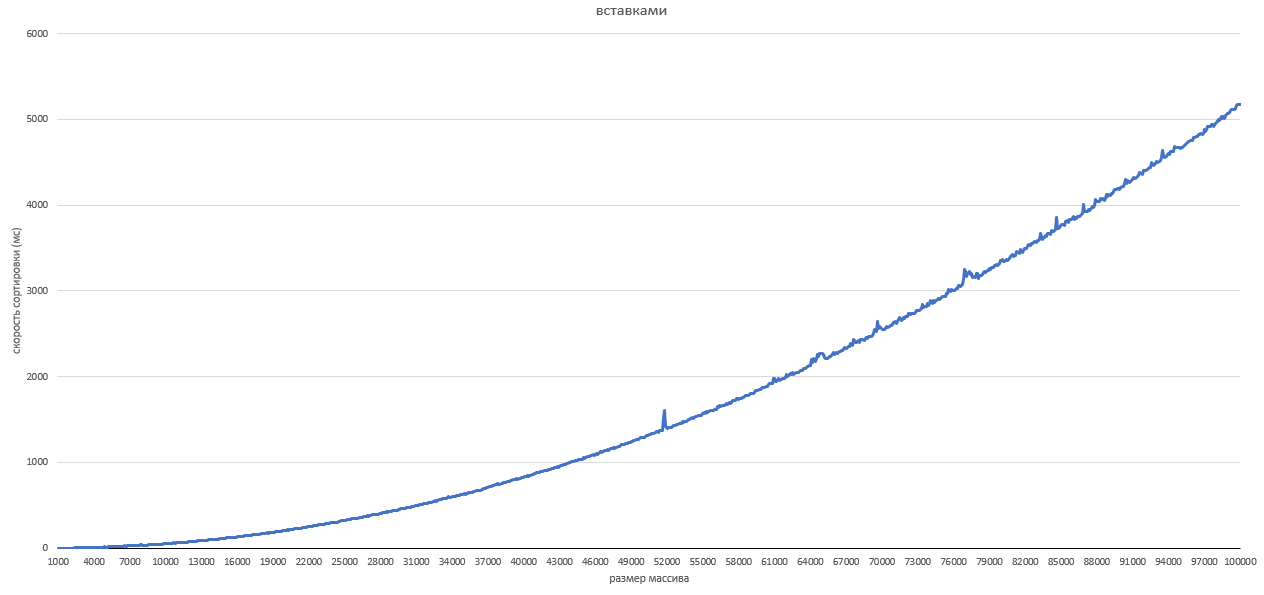
}

Sort(x, SIZE);

}

}

**Сортировка «вставкой»:**



#include <iostream>

#include <chrono>

#include <iostream>

#include <ctime>

using namespace std;

void Sort(int\* x, int size) { // собсна пузырёчек

auto begin = std::chrono::steady\_clock::now();

int temp, j;

for (int i = 1; i < size; i++) {

temp = x[i];

j = i - 1;

while (temp < x[j] && j >= 0) {

x[j + 1] = x[j];

--j;

}

x[j + 1] = temp;

}

auto end = std::chrono::steady\_clock::now();

auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin);

std::cout << elapsed\_ms.count() << " "; //вывод времени сортировки

}

int main() {

for (int SIZE = 1000; SIZE <= 100000; SIZE += 100) {

int\* x = new int[SIZE];

for (int i = 0; i < SIZE; i++) { //заполнение массива

x[i] = rand() % 100;

//cout << x[i];

}

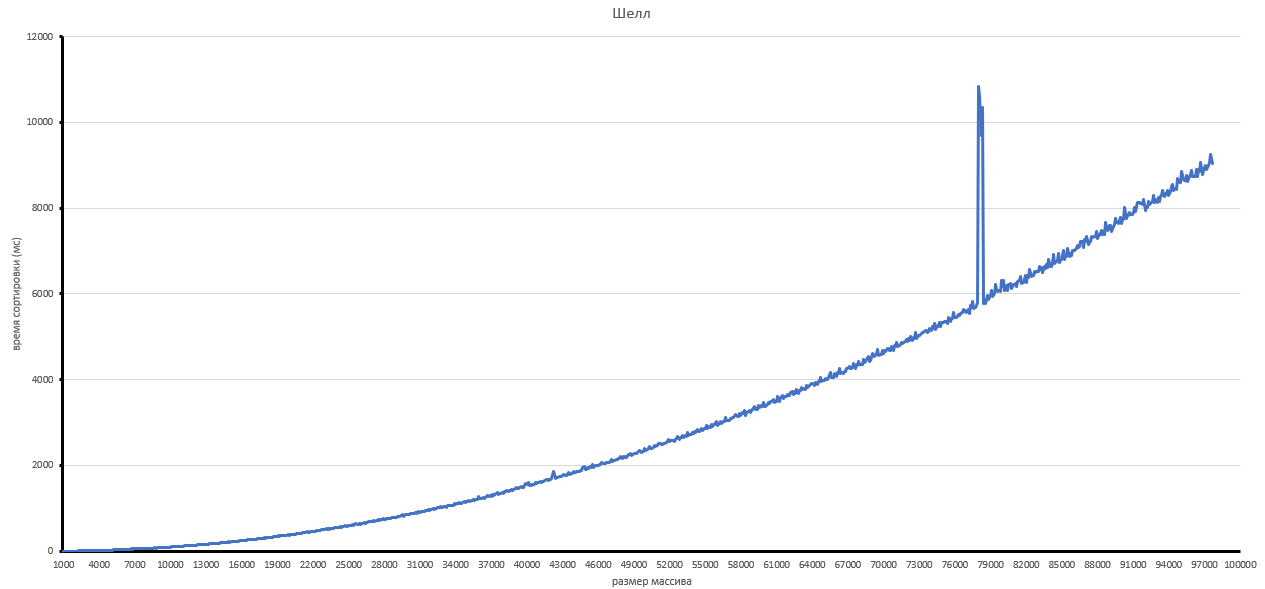
constexpr int size = sizeof(x) / sizeof(int);

Sort(x, SIZE);

}

}

**Сортировка Шелла:**



*Всплеск произошёл при отключении ноутбука от розетки.*

#include <iostream>

#include <chrono>

#include <iostream>

#include <ctime>

using namespace std;

void Sort(int\* x, int size) { // собсна Шелл

auto begin = std::chrono::steady\_clock::now();

int buf; // сохранение значения элемента сортируемого массива

for (int N = size / 2; N > 0; N /= 2) {

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

buf = x[i];

for (; i >= N && buf < x[i - N]; i -= N) {

x[i] = x[i - N];

}

x[i] = buf;

}

}

auto end = std::chrono::steady\_clock::now();

auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin);

std::cout << elapsed\_ms.count() << " "; //вывод времени сортировки

}

int main() {

for (int SIZE = 1000; SIZE <= 100000; SIZE += 100) {

int\* x = new int[SIZE];

for (int i = 0; i < SIZE; i++) { //заполнение массива

x[i] = rand() % 100;

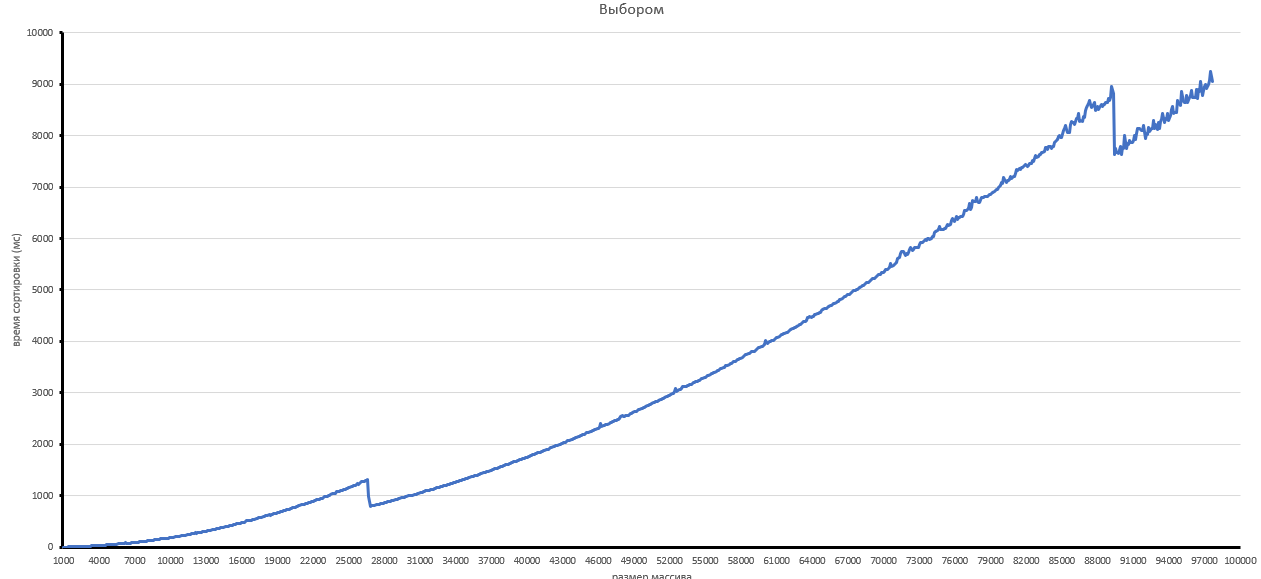
}

Sort(x, SIZE);

}

}

**Сортировка «выбором»:**



#include <iostream>

#include <chrono>

#include <iostream>

#include <ctime>

using namespace std;

void Sort(int\* x, int size) { // сортировочка выбором

auto begin = std::chrono::steady\_clock::now();

int minind = 0;

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

int min = x[i];

for (int j = i; j < size; j++) {

if (x[j] < i) {

minind = j;

min = x[j];

}

}

if (i != minind) {

x[minind] = x[i];

x[i] = min;

}

}

auto end = std::chrono::steady\_clock::now();

auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin);

std::cout << elapsed\_ms.count() << " "; //вывод времени сортировки

}

int main() {

for (int SIZE = 1000; SIZE <= 100000; SIZE += 100) {

int\* x = new int[SIZE];

for (int i = 0; i < SIZE; i++) { //заполнение массива

x[i] = rand() % 100;

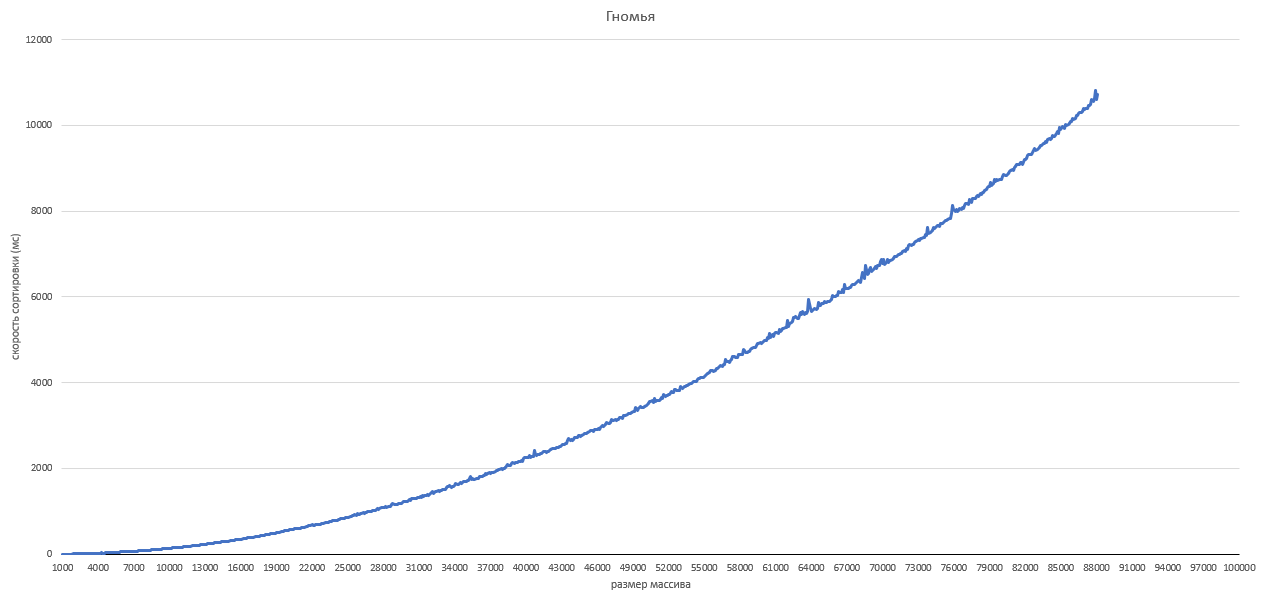
}

Sort(x, SIZE);

}

}

**Сортировка «гномья»:**



#include <iostream>

#include <chrono>

#include <iostream>

#include <ctime>

using namespace std;

void Sort(int\* x, int size) { // гномья сортировка

auto begin = std::chrono::steady\_clock::now();

int i = 1;

while (i < size) {

if ((x[i - 1] <= x[i]) or (i == 0)) {

++i;

}

else {

int tmp = x[i];

x[i] = x[i - 1];

x[i - 1] = tmp;

--i;

}

}

auto end = std::chrono::steady\_clock::now();

auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin);

std::cout << elapsed\_ms.count() << " "; //вывод времени сортировки

}

int main() {

for (int SIZE = 1000; SIZE <= 100000; SIZE += 100) {

int\* x = new int[SIZE];

for (int i = 0; i < SIZE; i++) { //заполнение массива

x[i] = rand() % 100;

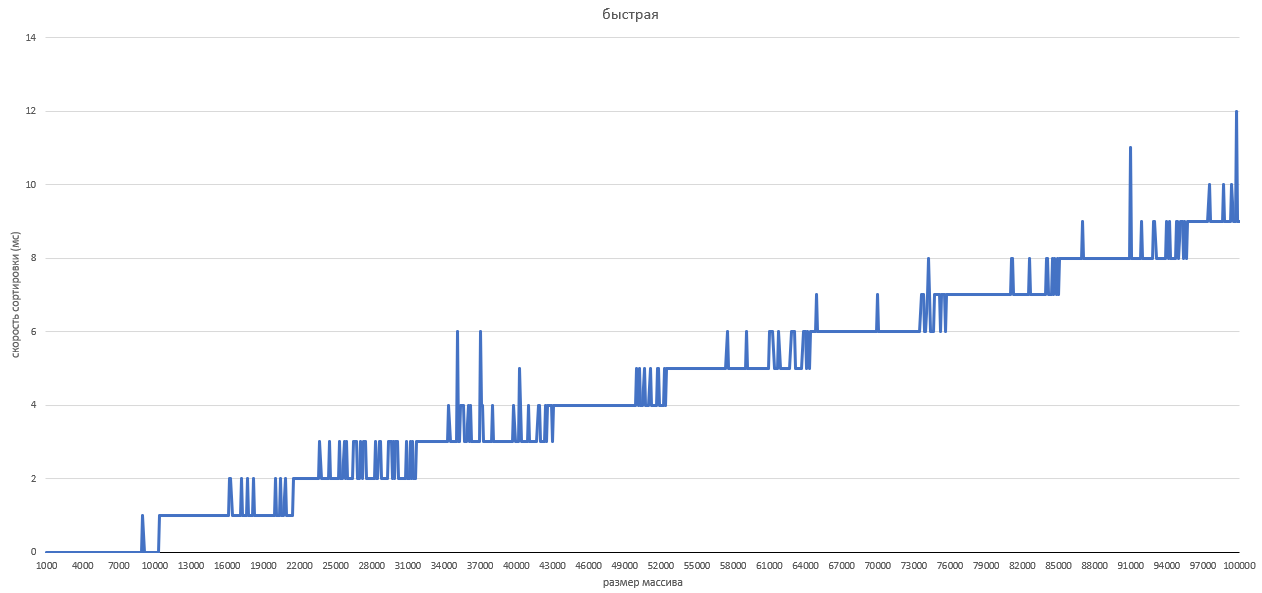
}

Sort(x, SIZE);

}

}

**Быстрая сортировка:**



#include <iostream>

#include <chrono>

#include <iostream>

#include <ctime>

using namespace std;

void qs(int\* arr, int L, int R) {

if (L < R) {

int left = L;

int right = R;

int middle = arr[(left + right) / 2];

do {

while (arr[left] < middle) left++; //сдвиг left вправо

while (arr[right] > middle) right--; //сдвиг right влево

if (left <= right) {

//перестановка

int tmp = arr[left];

arr[left] = arr[right];

arr[right] = tmp;

left++;

right--;

}

} while (left < right);

qs(arr, L, right);

qs(arr, left, R);

}

}

void Sort(int\* x, int size) { // собсна Шелл

auto begin = std::chrono::steady\_clock::now();

qs(x, 0, size - 1);

auto end = std::chrono::steady\_clock::now();

auto elapsed\_ms = std::chrono::duration\_cast<std::chrono::milliseconds>(end - begin);

std::cout << elapsed\_ms.count() << " "; //вывод времени сортировки

}

int main() {

for (int SIZE = 1000; SIZE <= 100000; SIZE += 100) {

int\* x = new int[SIZE];

for (int i = 0; i < SIZE; i++) { //заполнение массива

x[i] = rand() % 100;

}

Sort(x, SIZE);

}

}

**Вывод:** Иногда вывод занимает слишком много времени.

1. [↑](#endnote-ref-1)