ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО»

Отчёт по лабораторной работе № 5

«Разработка алгоритмов сортировки»

Выполнил работу

Воробьев И. А

Академическая группа J3111

Принято

Ментор, Вершинин Владислав Константинович

Санкт-Петербург

2024

1. Введение

Цель данной работы заключается в разработке алгоритмов для сортировки массивов различной длинны.

Задачи – изучить алгоритмы сортировки массивов, реализовать алгоритмы, сгенерировать тесты, измерить время работы алгоритмов для различного размера данных, написать отчет

2. Теоретическая часть

Для реализации данной лабораторной работы потребовалось:

* Знание vector
* Умение работы с рекурсивными алгоритмами
* Знание библиотеки измерения времени

3. Реализация

* Ввод значений с клавиатуры
* Реализация сортировки, используя один из трех алгоритмов: Шейкерная сортировка / Быстрая сортировка / Поразрядная сортировка
* Функции для тестирования

4. Экспериментальная часть

Согласно требованиям моего варианта, на вход к каждому алгоритму подаётся до 1000000 элементов. Теоретически заданная сложность задачи составляет O(N^2) для шейкерной сортировки, O(n\*log n) для быстрой сортировки, O(N\*k) для поразрядной, где N - количество элементов начального массива, K – максимальное количество разрядов среди входных чисел. Для тестирования алгоритма была собрана статистика. Часть значений приведена в таблице №1.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Размер входного набора | 1000 | 5000 | 10000 | 100000 | 500000 | 1000000 |
| Время выполнения “Пузырьковой сортировки”, сек | 0,000003 | 0,007 | 0,057 | 20 | 621 | 2488 |
| Время выполнения “Быстрой сортировки”, сек | 0,0002 | 0,001 | 0,001 | 0,018 | 0,152 | 0,266 |
| Время выполнения “Поразрядной сортировки”, сек | 0,000012 | 0,00003 | 0,007 | 0.04 | 0,206 | 0,415 |

Таблица №1 - Подсчёт сложности реализованного алгоритма

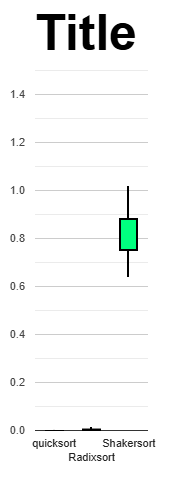
График представляющий визуально удобный формат всех данных представлен на изображении №1, 2.

Изображение 1

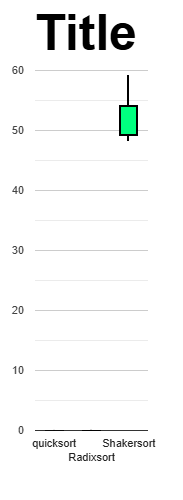
Изображение 2

Для оценки вариабельности значений было выполнено по 50 тестов для трех алгоритмов. Тесты были разделены на две категории: одна группа с вводом 100000 элементов, другая – с вводом 10000 элементов. Результаты тестирования визуализированы с использованием двух boxplot.

Boxplot с вводом 10000 элементов



Boxplot с вводом 100000 элементов



Заключение и выводы

В ходе выполнения работы мною было реализовано три алгоритма: “Шейкерная сортировка”, “Быстра сортировка”, “Поразрядная сортировка” и исследована зависимость между входными данными и временем выполнения для каждого алгоритма.

Алгоритм и тестирующая система были реализованы на языке C++. Генератор тестов на python.

В качестве дальнейших исследований можно предложить исследование зависимости использованной памяти или количества операций сравнения.

Приложение А

Листинг кода файла quicksort.cpp

#include<iostream>

#include<vector>

#include<string>

#include<fstream>

#include<ctime>

#include<iomanip>

int Partition(std::vector<int> &arr, const int l, const int r) {

// 1 + 4 + 4 байта внутри функции

const int x = arr[r]; // +4 байта

int small\_part\_ln = 0; // + 4 байта

for (int i=l; i <= r ; i++) { // O(r-l)

// + 4 байта внутри цикла

// здесь будет выделено максимальное кол-во памяти

// в худшем случаи 9 + 13\*len\_data байт

if (arr[i] <= x) {

std::swap(arr[l+small\_part\_ln], arr[i]);

small\_part\_ln += 1;

}

}

return l + small\_part\_ln - 1;

}

void SortPart(std::vector<int> &arr, const int l, const int r) {

// 1 + 4 + 4 байта внутри функции

if (l >= r) {

return;

}

const int separator = Partition(arr, l, r); // + 4 байта, O(r-l)

SortPart(arr, l, separator - 1);

SortPart(arr, separator + 1, r);

// максимальная глубина рекурсии - len\_data, отсюда в худшем случаи на SortPart будет затрачено 13 \* len\_data байт

}

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

// 1 байт из-за ссылки

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

}

bool tests() {

std::ifstream findata, fin\_ans;

findata.open("unit\_tests\\quicksort\_bad\_test.txt");

fin\_ans.open("unit\_tests\\quicksort\_bad\_test\_answers.txt");

for (int test=0; test < 1000 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

QuickSort(data);

for (int i=0; i < len\_data ; i++) {

fin\_ans >> input\_val;

if (input\_val != data[i]) {

std::cout << "failed on bad test " << i + 1 << '\n';

std::cout << "Program out:\n";

for (int j=0; j < len\_data ; j++) {

std::cout << data[j] << ' ';

}

return false;

}

}

}

findata.close();

fin\_ans.close();

findata.open("unit\_tests\\quicksort\_average\_test.txt");

fin\_ans.open("unit\_tests\\quicksort\_average\_test\_answers.txt");

for (int test=0; test < 1000 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

QuickSort(data);

for (int i=0; i < len\_data ; i++) {

fin\_ans >> input\_val;

if (input\_val != data[i]) {

std::cout << "failed on average test " << i + 1 << '\n';

std::cout << "Program out:\n";

for (int j=0; j < len\_data ; j++) {

std::cout << data[j] << ' ';

}

return false;

}

}

}

findata.close();

fin\_ans.close();

return true;

}

void time\_tests() {

std::fstream findata("time\_tests\\time\_tests0.txt");

std::ofstream out("time\_proc\_quick.txt");

for (int test=0; test < 400 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

QuickSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

out << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

}

findata.close();

findata.open("time\_tests\\time\_tests1.txt");

for (int test=0; test < 300 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

QuickSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

out << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

}

findata.close();

findata.open("time\_tests\\time\_tests2.txt");

for (int test=0; test < 150 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

QuickSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

out << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

}

findata.close();

findata.open("time\_tests\\time\_tests3.txt");

for (int test=0; test < 150 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

QuickSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

out << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

}

findata.close();

out.close();

}

void box\_tests() {

std::fstream findata("box\_tests\\random\_tests.txt");

std::ofstream out("box\_tests\\time\_box\_quick.txt");

for (int test=0; test < 50 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

QuickSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(10) << execution\_time << '\n';

out << std::setprecision(10) << execution\_time << '\n';

}

findata.close();

out.close();

findata.open("box\_tests\\random\_tests\_long.txt");

out.open("box\_tests\\time\_box\_quick2.txt");

for (int test=0; test < 50 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

QuickSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(10) << execution\_time << '\n';

out << std::setprecision(10) << execution\_time << '\n';

}

findata.close();

out.close();

}

int main() {

std::vector<int> data(0); //24 + 4 \* len\_data байт

int len\_data, input\_val; // 8 байт

std::cin >> len\_data;

for (int i=0; i < len\_data ; i++) { //O(len\_data)

//+4 байта в цикле

std::cin >> input\_val;

data.push\_back(input\_val);

}

QuickSort(data);

for (int i=0; i < len\_data ; i++) {

//+4 байта в цикле

std::cout << data[i] << ' ';

}

return 0;

}

Приложение B

Листинг кода файла shakersort.cpp

#include<iostream>

#include<vector>

#include<string>

#include<fstream>

#include<ctime>

#include<iomanip>

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

// 1 байт из-за ссылки

for (int iter=0; iter < arr.size()/2 ; iter++) {

//+4 байта в цикле

bool is\_edit = false; //1 байт

for (int i=iter; i < (arr.size() - iter - 1) ; i++) { //O(len\_data - iter)

//+4 байта в цикле

// здесь будет затрачено больше всего памяти 9 байт

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

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

is\_edit = true;

}

}

for (int i=(arr.size() - iter - 1); i > iter ; i--) { //O(len\_data - iter)

//+4 байта в цикле

// здесь будет затрачено больше всего памяти 9 байт

if (arr[i - 1] > arr[i]) {

std::swap(arr[i - 1], arr[i]);

is\_edit = true;

}

}

if (!is\_edit) {

return;

}

// итерация работает за O(len\_data - iter)

// итого все итерации отработают за O(len\_data \* len\_data) в худшем случаи

}

}

bool tests() {

std::ifstream findata, fin\_ans;

findata.open("unit\_tests\\shakersort\_bad\_test.txt");

fin\_ans.open("unit\_tests\\shakersort\_bad\_test\_answers.txt");

for (int test=0; test < 1000 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

ShakerSort(data);

for (int i=0; i < len\_data ; i++) {

fin\_ans >> input\_val;

if (input\_val != data[i]) {

std::cout << "failed on bad test " << i + 1 << '\n';

std::cout << "Program out:\n";

for (int j=0; j < len\_data ; j++) {

std::cout << data[j] << ' ';

}

return false;

}

}

}

findata.close();

fin\_ans.close();

findata.open("unit\_tests\\shakersort\_average\_test.txt");

fin\_ans.open("unit\_tests\\shakersort\_average\_test\_answers.txt");

for (int test=0; test < 1000 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

ShakerSort(data);

for (int i=0; i < len\_data ; i++) {

fin\_ans >> input\_val;

if (input\_val != data[i]) {

std::cout << "failed on average test " << i + 1 << '\n';

std::cout << "Program out:\n";

for (int j=0; j < len\_data ; j++) {

std::cout << data[j] << ' ';

}

return false;

}

}

}

findata.close();

fin\_ans.close();

return true;

findata.open("unit\_tests\\shakersort\_good\_test.txt");

fin\_ans.open("unit\_tests\\shakersort\_good\_test\_answers.txt");

for (int test=0; test < 1000 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

ShakerSort(data);

for (int i=0; i < len\_data ; i++) {

fin\_ans >> input\_val;

if (input\_val != data[i]) {

std::cout << "failed on good test " << i + 1 << '\n';

std::cout << "Program out:\n";

for (int j=0; j < len\_data ; j++) {

std::cout << data[j] << ' ';

}

return false;

}

}

}

findata.close();

fin\_ans.close();

return true;

}

void time\_tests() {

std::fstream findata("time\_tests\\time\_tests0.txt");

std::ofstream out("time\_proc\_shaker.txt");

for (int test=0; test < 400 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

ShakerSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

out << std::setprecision(20) << execution\_time << '\n';

}

findata.close();

findata.open("time\_tests\\time\_tests1.txt");

for (int test=0; test < 300 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

ShakerSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

out << std::setprecision(20) << execution\_time << '\n';

}

findata.close();

findata.open("time\_tests\\time\_tests2.txt");

for (int test=0; test < 150 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

ShakerSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

out << std::setprecision(20) << execution\_time << '\n';

}

findata.close();

findata.open("time\_tests\\time\_tests3.txt");

for (int test=0; test < 150 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

ShakerSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

out << std::setprecision(20) << execution\_time << '\n';

}

findata.close();

out.close();

}

void box\_tests() {

std::fstream findata("box\_tests\\random\_tests.txt");

std::ofstream out("box\_tests\\time\_box\_shaker.txt");

for (int test=0; test < 50 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

ShakerSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(10) << execution\_time << '\n';

out << std::setprecision(10) << execution\_time << '\n';

}

findata.close();

out.close();

findata.open("box\_tests\\random\_tests\_long.txt");

out.open("box\_tests\\time\_box\_shaker2.txt");

for (int test=0; test < 50 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

ShakerSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(10) << execution\_time << '\n';

out << std::setprecision(10) << execution\_time << '\n';

}

findata.close();

out.close();

}

int main() {

std::vector<int> data(0); //24 + 4 \* len\_data байт

int len\_data, input\_val; // 8 байт

std::cin >> len\_data;

for (int i=0; i < len\_data ; i++) { //O(len\_data)

//+4 байта в цикле

std::cin >> input\_val;

data.push\_back(input\_val);

}

ShakerSort(data);

for (int i=0; i < len\_data ; i++) {

//+4 байта в цикле

std::cout << data[i] << ' ';

}

return 0;

}

Приложение C

Листинг кода файла radixsort.cpp

#include<iostream>

#include<vector>

#include<string>

#include<fstream>

#include<ctime>

#include<iomanip>

int get\_radix(int a) {

// + 4 байта переменная a

int ans = 1; // + 4 байта

a /= 10;

while (a) { // O(кол-во разрядов a)

ans += 1;

a /= 10;

}

// здесь будет испольоваться больше всего памяти - 8 + 8 + 8 + 24 + 10\*24 + 4\*len\_data + 1 байт

return ans;

}

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

// + 1 байт &arr

std::vector<std::vector<int>> radixes = std::vector<std::vector<int>> (10, std::vector<int>(0)); // 24 + 10\*24 + 4\*len\_data байт

int max\_ln = 0, k = 1; // + 8 байт

for (int i=0; i < arr.size() ; i++) { // O(len\_data \* макс кол-во разрядов)

// + 4 байта внутри цикла

max\_ln = std::max(max\_ln, get\_radix(arr[i]));

}

for (int r=0; r < max\_ln ; r++) {

// + 4 байта внутри цикла

for (int i=0; i < arr.size() ; i++) { // O(len\_data)

// + 4 байта внутри цикла

radixes[(arr[i]/k) % 10].push\_back(arr[i]);

}

arr.clear();

for (int digit=0 ; digit < 10 ; digit++) { // O(len\_data)

// + 4 байта внутри цикла

for (int i=0; i < radixes[digit].size() ; i++) {

// + 4 байта внутри цикла

arr.push\_back(radixes[digit][i]);

}

}

radixes = std::vector<std::vector<int>> (10, std::vector<int>(0));

k \*= 10;

// цикл отработает за O(max\_ln \* len\_data)

}

//В итоге вся функция отработает за O(макс кол-во разрядов \* len(data))

}

bool tests() {

std::ifstream findata, fin\_ans;

findata.open("unit\_tests\\radixsort\_bad\_test.txt");

fin\_ans.open("unit\_tests\\radixsort\_bad\_test\_answers.txt");

for (int test=0; test < 1000 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

RadixSort(data);

for (int i=0; i < len\_data ; i++) {

fin\_ans >> input\_val;

if (input\_val != data[i]) {

std::cout << "failed on bad test " << i + 1 << '\n';

std::cout << "Program out:\n";

for (int j=0; j < len\_data ; j++) {

std::cout << data[j] << ' ';

}

return false;

}

}

}

findata.close();

fin\_ans.close();

findata.open("unit\_tests\\radixsort\_average\_test.txt");

fin\_ans.open("unit\_tests\\radixsort\_average\_test\_answers.txt");

for (int test=0; test < 1000 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

RadixSort(data);

for (int i=0; i < len\_data ; i++) {

fin\_ans >> input\_val;

if (input\_val != data[i]) {

std::cout << "failed on average test " << i + 1 << '\n';

std::cout << "Program out:\n";

for (int j=0; j < len\_data ; j++) {

std::cout << data[j] << ' ';

}

return false;

}

}

}

findata.close();

fin\_ans.close();

return true;

}

void time\_tests() {

std::fstream findata("time\_tests\\time\_tests0.txt");

std::ofstream out("time\_proc.txt");

for (int test=0; test < 400 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

RadixSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

out << std::setprecision(20) << execution\_time << '\n';

}

findata.close();

findata.open("time\_tests\\time\_tests1.txt");

for (int test=0; test < 300 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

RadixSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

out << std::setprecision(20) << execution\_time << '\n';

}

findata.close();

findata.open("time\_tests\\time\_tests2.txt");

for (int test=0; test < 150 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

RadixSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

out << std::setprecision(20) << execution\_time << '\n';

}

findata.close();

findata.open("time\_tests\\time\_tests3.txt");

for (int test=0; test < 150 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

RadixSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(20) << execution\_time << '\n';

out << std::setprecision(20) << execution\_time << '\n';

}

findata.close();

out.close();

}

void box\_tests() {

std::fstream findata("box\_tests\\random\_tests.txt");

std::ofstream out("box\_tests\\time\_box\_radix.txt");

for (int test=0; test < 50 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

RadixSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(10) << execution\_time << '\n';

out << std::setprecision(10) << execution\_time << '\n';

}

findata.close();

out.close();

findata.open("box\_tests\\random\_tests\_long.txt");

out.open("box\_tests\\time\_box\_radix2.txt");

for (int test=0; test < 50 ; test++) {

int len\_data;

int input\_val;

std::vector<int> data(0);

findata >> len\_data;

for (int i=0; i < len\_data ; i++) {

findata >> input\_val;

data.push\_back(input\_val);

}

unsigned long long t\_start = clock();

RadixSort(data);

unsigned long long t\_end = clock();

double execution\_time = (double)(t\_end - t\_start) / 1000;

std::cout << len\_data << ' ' << std::setprecision(10) << execution\_time << '\n';

out << std::setprecision(10) << execution\_time << '\n';

}

findata.close();

out.close();

}

int main() {

std::vector<int> data(0); //24 + 4 \* len\_data байт

int len\_data, input\_val; // 8 байт

std::cin >> len\_data;

for (int i=0; i < len\_data ; i++) { //O(len\_data)

//+4 байта в цикле

std::cin >> input\_val;

data.push\_back(input\_val);

}

RadixSort(data);

for (int i=0; i < len\_data ; i++) {

//+4 байта в цикле

std::cout << data[i] << ' ';

}

return 0;

}