**ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ**

**НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ**

**«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»**

Факультет компьютерных наук

Департамент программной инженерии

**Отчет**

**по учебному проекту**

по теме **Основы С++. Сравнение методов сортировок массивов**

по направлению подготовки бакалавров 09.03.04 «Программная инженерия»

|  |  |
| --- | --- |
|  | Выполнил  студент группы БПИ184  образовательной программы 09.03.04 «Программная инженерия»  А. Д. Курылев  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  И.О. Фамилия  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Подпись, Дата |

Принял

Доцент ДПИ, к.т.н

Р.З. Ахметсафина

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Оценка

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись, Дата

**Москва 2019**

**Постановка задачи**

**Цель работы**

Определение экспериментальным путем эффективности работы различных алгоритмов сортировки при различных входных данный.

**Алгоритмы сортировки**

1. Сортировка методом пузырька
2. Сортировка методом пузырька с условием Айверсона 1
3. Сортировка методом пузырька с условием Айверсона 2
4. Сортировка простыми вставками
5. Сортировка бинарными вставками
6. Сортировка подсчетом
7. Цифровая сортировка

**Входные данные**

Массивы размером от 1000 до 8000 элементов и одним из условий:

1. Элементы принадлежат отрезку [0, 9]
2. Элементы принадлежат отрезку [0, 10000]
3. Элементы стоят в почти (9 пар элементов поменяны местами) сортированном порядке
4. Элементы стоят в обратно сортированном порядке

**Результаты**

**Описание выполненной работы**

В результате проделанной работы были реализованы 5 видом сортировки:

1. Сортировка методом пузырька
2. Сортировка методом пузырька с условием Айверсона 1
3. Сортировка методом пузырька с условием Айверсона 2
4. Сортировка простыми вставками
5. Сортировка бинарными вставками

Также был произведен подсчет количества операций, за которые различные виды сортировки могут отсортировать массивы.

**Используемые алгоритмы**

Ниже представлены алгоритмы сортировки массивов целых чисел. Все алгоритмы написаны на языке программирования C++. Они принимают на вход исходный массив и его размер, сортируют его, и так как массив передается по ссылке, пользователь в основной программе может просматривать сортированный массив.

**Сортировка методом пузырька**

void bubbleSort(int\* array, int N)

{

for (size\_t i = 0; i < N - 1; i++)

{

for (size\_t j = 0; j < N - 1 - i; j++)

{

if (array[j] > array[j + 1])

{

int a = array[j];

array[j] = array[j + 1];

array[j + 1] = a;

}

}

}

}

**Сортировка методом пузырька с условием Айверсона 1**

void bubbleSortIverson1(int\* array, int N)

{

bool flag = false;

for (size\_t i = 0; i < N - 1 && !flag; i++)

{

flag = true;

for (size\_t j = 0; j < N - 1 - i; j++)

{

if (array[j] > array[j + 1])

{

int a = array[j];

array[j] = array[j + 1];

array[j + 1] = a;

flag = false;

}

}

}

}

**Сортировка методом пузырька с условием Айверсона 2**

void bubbleSortIverson2(int\* array, int N)

{

int t = N;

while (t != 0)

{

int bound = t;

t = 0;

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

if (array[i] < array[i - 1]) {

int a = array[i];

array[i] = array[i + 1];

array[i + 1] = a;

t = i;

}

}

}

}

**Сортировка простыми вставками**

void simpleInsertSort(int\* array, int N)

{

for (size\_t i = 1; i < N; i++)

{

int b = array[i];

int j = i - 1;

while (b < array[j] && j >= 0)

{

array[j + 1] = array[j];

j = j - 1;

}

array[j + 1] = b;

}

}

**Сортировка бинарными вставками**

void binaryInsertSort(int\* array, int N)

{

for (size\_t i = 1; i < N; i++)

{

int lf = 0, rg = i - 1;

int c = (lf + rg) / 2;

while (lf != c)

{

if (array[c] > array[i])

{

rg = c - 1;

}

else

{

lf = c;

}

c = (lf + rg) / 2;

}

if (array[lf] < array[i])

{

if (array[i] > array[rg])

{

lf = rg + 1;

}

else

{

lf = rg;

}

}

int k = i;

int p = array[i];

while (k > lf)

{

array[k] = array[k - 1];

k = k - 1;

}

array[lf] = p;

}

}

**Организация эксперимента**

В качестве входных данных для функций сортировки использовались массивы размером от 1000 до 8000 элементов и одним из условий:

1. Элементы принадлежат отрезку [0, 9]
2. Элементы принадлежат отрезку [0, 10000]
3. Элементы стоят в почти (9 пар элементов поменяны местами) сортированном порядке
4. Элементы стоят в обратно сортированном порядке

Всего массивов было по 32 штуки на каждую функцию сортировки, то есть в сумме было отсортировано 160 массивов. Ниже представлен алгоритм подачи входных данных в функции сортировки и получения результатов измерения количества операций в файл output.csv. Алгоритм написан на языке C++.

**Организация эксперимента**

srand(1337);

int\*\* array = new int\* [4];

for (size\_t i = 0; i < 4; i++)

array[i] = new int[8000];

for (size\_t i = 0; i < 8000; i++)

array[0][i] = rand() % 10;

for (size\_t i = 0; i < 8000; i++)

array[1][i] = rand() % 10001;

for (size\_t i = 0; i < 8000; i++)

{

array[2][i] = i;

array[3][i] = i;

}

for (size\_t i = 0; i < 9; i++)

swap(array[2][rand() % 8000], array[2][rand() % 8000]);

reverse(&array[3][0], &array[3][7999]);

void (\*functions[5])(int\* a, int n) = { bubbleSort, bubbleSortIverson1, bubbleSortIverson2, simpleInsertSort, binaryInsertSort };

ofstream fout("output.csv");

fout << "Количество элементов;Пузырек 0-9;Пузырек 0-10к;Пузырек сортированный;Пузырек обратно сортированный;Айверсон 1 0-9;Айверсон 1 0-10к;Айверсон 1 сортированный;Айверсон 1 обратно сортированный;Айверсон 2 0-9;Айверсон 2 0-10к;Айверсон 2 сортированный;Айверсон 2 обратно сортированный;Вставки 0-9;Вставки 0-10к;Вставки сортированный;Вставки обратно сортированный;Бинарные вставки 0-9;Бинарные вставки 0-10к;Бинарные вставки сортированный;Бинарные вставки обратно сортированный;" << "\n";

int\* temp;

for (size\_t i = 1; i <= 8; i++)

{

fout << i << "000 элементов в массиве;";

for (size\_t j = 0; j < 5; j++)

{

for (size\_t k = 0; k < 4; k++)

{

int n = i \* 1000;

temp = new int[n];

for (size\_t l = 0; l < n; l++)

{

temp[l] = array[k][l];

}

functions[j](temp, n);

fout << operationsCount << ";";

}

}

fout << "\n";

}

fout.close();

**Выводы**

По результатам эксперимента были построены графики, представленные в приложении 1.

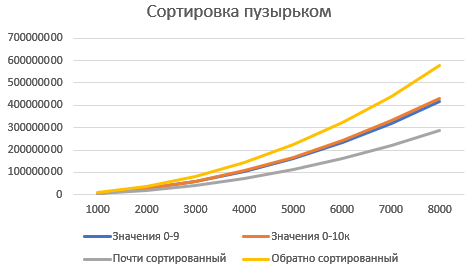
Из рисунков 1,2,4,5 можно понять, что величина чисел, записанных в массив, не сильно влияет на результаты теста. Также видно, что обратно сортированный массив занимает больше всего операций, случайно сгенерированный массив – среднее число, почти сортированный – минимальное.

На рисунке 3 можно увидеть, что при сортировке методом пузырька с условием Айверсона 2, во-первых, сильно влияет величина элементов массива, во-вторых, что сортировка массива с элементами от 0 до 9 занимает в среднем меньше операций, чем все остальные. Также можно увидеть перелом графика почти сортированного массива при количестве элементов массива, равном 5000, после этого значения алгоритм начинает тратить гораздо меньше времени на сортировку некоторого числа элементов, чем раньше.

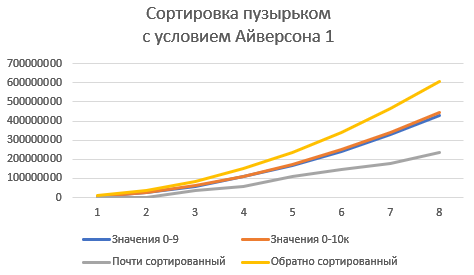
По рисунку 6 можно понять, что сортировка методом пузырька и с условием Айверсона 1 работают примерно одинаково и требуют больше всего операций. Алгоритмы сортировки с условием Айверсона 2 и бинарный поиск также работают практически одинаково и требуют меньше всего операций. На рисунке 7 можем видеть, что в диапазоне значений 0-10000 все алгоритмы сортировки пузырьком работают примерно одинаково по количеству операций, равно как и алгоритмы сортировки вставками. По рисунку 8 можно понять, что для алгоритмов сортировки с условием Айверсона 1 и 2 важен размер сортируемого массива, так как после отметки в 5000 элементов график Айверсона 2 стал расти значительно медленнее, а график алгоритма с условием Айверсона 1, наоборот, после отметки в 4000 элементов стал расти быстрее. Но алгоритмы сортировки методом пузырька в любом случае проигрывают по количеству операций алгоритмам сортировок вставками. На рисунке 9 мы можем видеть, что с обратно сортированными массивами метод сортировки с условием Айверсона 2 справляется лучше остальных методов сортировки пузырьком, а при увеличении числа элементов массива он работает даже лучше метода простых вставок.

В целом, по прикрепленным рисункам можно сделать вывод, что из исследованных мной алгоритмов сортировки, в среднем экономнее в плане количества операций работает метод сортировки бинарными вставками, хотя, например, у метода сортировки простыми вставками заняло меньше операций отсортировать почти сортированный массив.

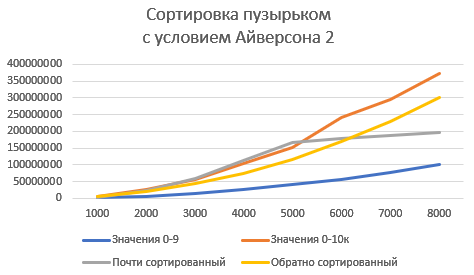
**Приложение 1**



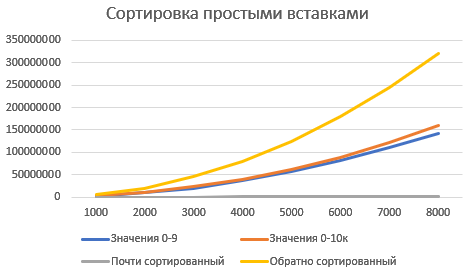
**Рисунок 1 –** «График сортировка методом пузырька»



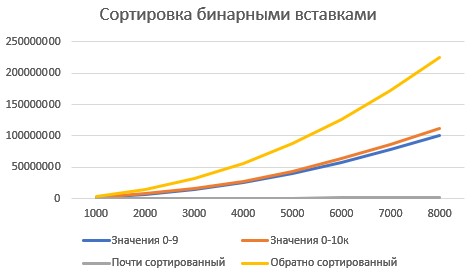
**Рисунок 2** – «График сортировки методом пузырька с условием Айверсона 1»



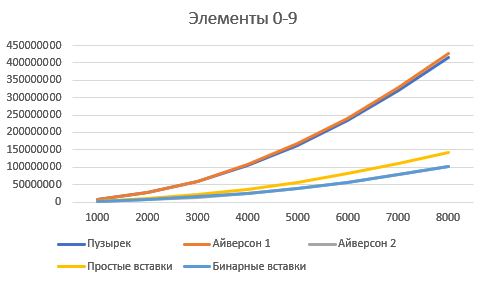
**Рисунок 3** – «График сортировки методом пузырька с условием Айверсона 2»



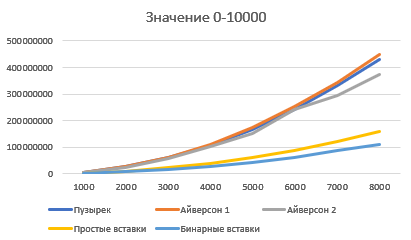
**Рисунок 4** – «График сортировки методом простых вставок»



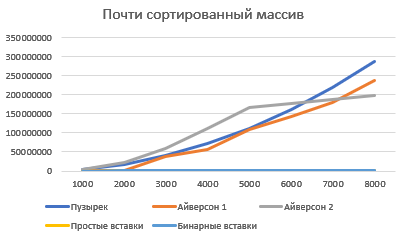
**Рисунок 5** – «График сортировки методом бинарных вставок»



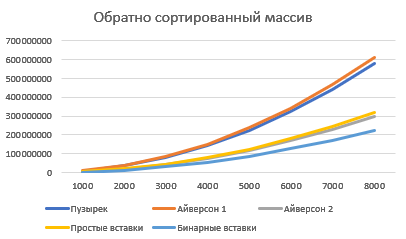
**Рисунок 6** – «График сортировки массива с элементами из [0, 9]»



**Рисунок 7** – «График сортировки массива с элементами из [0, 10000]»



**Рисунок 8 –** «График сортировки почти сортированного массива»



**Рисунок 9 –** «График сортировок обратно сортированного массива»