## Федеральное государственное автономное образовательное учреждение высшего образования

## **Санкт-Петербургский национальный исследовательский университет**

# информационных технологий, механики и оптики

# Кафедра ИПМ

# Лабораторная работа №2

1 курс

**Сортировки за O(n\*logn)**

# Выполнил

Студент 2 курса

# Группы P3218

# Гхази Даниэль

# Принял:

К.т.н.

Старший преподаватель

Симоненко Зинаида

Григорьевна

Санкт-Петербург

2016

**Цель работы:**

Для выполнения лабораторной работы необходимо сгенерировать тестовые файлы (используя генераторы случайных чисел), содержащие целые числа, в количестве от 26 до 220 , реализовать алгоритмы используя один из следующих языков программирования: C++, C#, C, Python, для каждого тестового файла из набора выполнить сортировку данных, вычислить среднее время сортировки по одному файлу, построить график зависимости времени сортировки от количества элементов в файле, выполнить сравнение алгоритмов.

**Генератор исходных данных:**

Int main()

{

srand(time(NULL));

FILE \*file;

if ((file = fopen("mas.txt", "wt")) == NULL)

return 1;

for (inti = 0; i<ARR\_SIZE; i++)

fprintf(file, "%i", rand() % ARR\_SIZE);

fclose(file);

return 0;

}

**Сортировки:**

**Сортировка бинарными вставками**

#define ARR\_SIZE 100000

int start, end, curPos, newElem;

for (int i = 0; i < ARR\_SIZE - 1; i++)

{

start = 0;

end = i;

newElem = mas[i + 1];

while (start != end)

{

curPos = (start + end) / 2;

if (newElem > mas[curPos])

start = curPos + 1;

else

end = curPos - 1;

if ((start > i) || (start > end))

start = end;

else if (end < 0)

end = start;

}

curPos = end;

if (newElem > mas[curPos])

curPos += 1;

for (int j = i + 1; j > curPos; j--)

mas[j] = mas[j - 1];

mas[curPos] = newElem;

}

**Быстрая сортировка**

#define ARR\_SIZE 100000

int l = 0;

int r = ARR\_SIZE - 1;

quickSort(mas, l, r);

void quickSort(int \*p\_mas, int s, int e)

{

int l = s, r = e;

int piv = p\_mas[(l + r) / 2];

while (l <= r)

{

while (p\_mas[l] < piv)

l++;

while (p\_mas[r] > piv)

r--;

if (l <= r)

{

int temp = p\_mas[l];

p\_mas[l] = p\_mas[r];

p\_mas[r] = temp;

l++;

r--;

}

}

if (s < r)

quickSort(p\_mas, s, r);

if (e > l)

quickSort(p\_mas, l, e);

}

**Результаты:**

|  |  |  |
| --- | --- | --- |
| Количество элементов | Бинарными вставками, мс | Быстрая сортировка, мс |
| 32000 | 1210 | 16 |
| 64000 | 3190 | 15 |
| 128000 | 5539 | 45 |
| 256000 | 49439 | 60 |
| 512000 | 264231 | 119 |

**Выводы:**

В результате сравнения алгоритмов на большом количестве значений удалось выяснить, что, начиная со 128000 рубежа, алгоритм быстрой сортировки работает в сотни раз быстрее. Следовательно, он является наиболее предпочтительным для решения задач с большим объемом сортируемых данных.