Санкт-Петербургский политехнический университет Петра Великого
Институт машиностроения, материалов и транспорта

КУРСОВАЯ РАБОТА

по дисциплине «Объектно-ориентрованное программирование»

Выполнил
студент группы 3331506/80401 _____ Д. В. Белоусов

Руководитель _____ Е. М. Кузнецова

«__»_____2021 г

Санкт-Петербург
2021

Оглавление

1 ВВЕДЕНИЕ

Сортировка вставками – алгоритм сортировки, в котором элементы входной последовательности просматриваются по одному, и каждый новый поступивший элемент размещается в подходящее место среди ранее упорядоченных элементов. это базовый алгоритм сортировки, который применяется в основном в учебном программировании. Он прост в реализации и эффективен при работе с небольшими массивами, а также с частично отсортированными массивами. Этот алгоритм не подходит для повсеместного использования из-за высокой вычислительной сложности, но отлично справляется с небольшими частично отсортированными массивами.

2 ИДЕЯ АЛГОРИТМА

Массив разбивается на две области: упорядоченную и неупорядоченную. Изначально весь массив является неупорядоченной областью. При первом проходе первый элемент из неупорядоченной области изымается и помещается в правильном положении в упорядоченной области. На каждом проходе размер упорядоченной области возрастает на 1, а размер неупорядоченной области сокращается на 1.

Суть алгоритма заключается в следующем: берем нулевой элемент исходного массива как условно отсортированный, затем, сравнивая следующий элемент из неотсортированного массива с элементами отсортированного, переносим его в отсортированную часть. Таким образом, элементы из неотсортированной части один за другим переходят на свое место в отсортированную часть. Рассмотреть детальнее это можно на примере:

Допустим, я хочу отсортировать массив mas 1 = [9 8 3 5 7]

Определим нулевой элемент этого массива, то есть 9, как условно отсортированный элемент и переместим его условно влево – в отсортированную часть.

9___8 7 5 3

Далее, берем первый элемент исходного массива и сравниваем его с отсортированной частью. Так как 8 меньше, чем 9, то в отсортированной части поставим ее слева от 9.

8 9__ 7 5 3

Далее будем сравнивать второй элемент исходного массива с отсортированной частью. 7 меньше чем 9, поэтому поместим ее слева от 9:

8 7 9__5 3

Но 7 также меньше чем 8, поэтому теперь ставим ее слева от 8:

7 8 9__ 5 3.

Действуем по данному алгоритму до полной сортировки массива.

3 СКОРОСТЬ РАБОТЫ

Вход для наилучшего случая - это уже отсортированный массив. В этом случае сортировка вставкой имеет линейное время выполнения (то есть O (n)). Во время каждой итерации первый оставшийся элемент ввода сравнивается только с крайним правым элементом отсортированной части массива.

Простейший вход в худшем случае - это массив, отсортированный в обратном порядке. Набор всех входных параметров наихудшего случая состоит из всех массивов, где каждый элемент является наименьшим или вторым наименьшим из элементов перед ним. В этих случаях каждая итерация внутреннего цикла будет сканировать и сдвигать всю отсортированную часть массива перед вставкой следующего элемента. Это дает сортировке вставки квадратичное время выполнения (то есть O (n^2)).

4 РЕЗУЛЬТАТЫ РАБОТЫ АЛГОРИТМА

В качестве входных данных будем подавать массивы разных размеров. Массивы заполняются случайными числами. Замерим время выполнения для каждого случая и построим график зависимости времени выполнения (в нс) от количества элементов (рисунок 4.1).

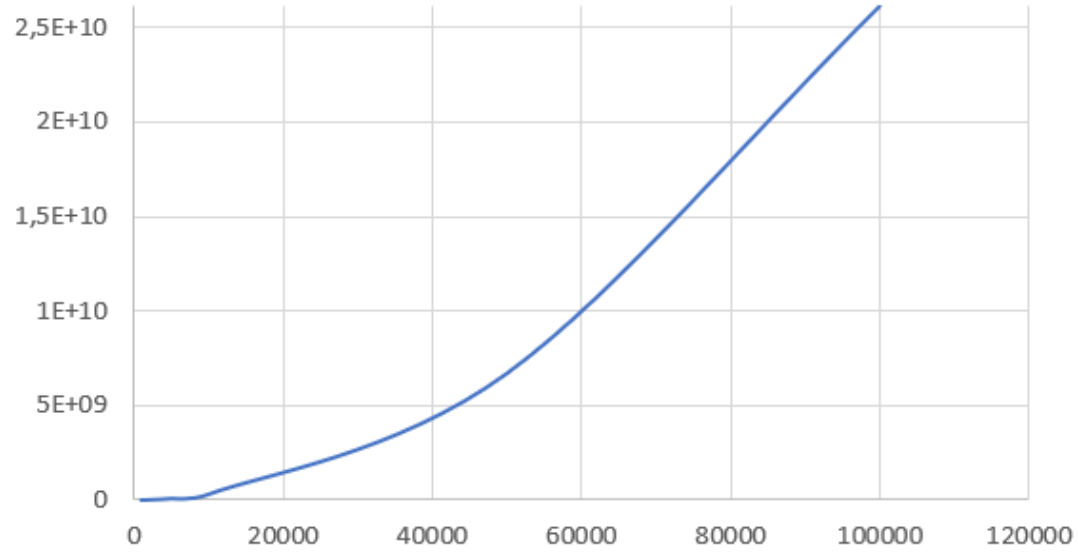

Рисунок 4.1 – График зависимости времени от количества элементов

5 СПИСОК ЛИТЕРАТУРЫ

1. Левитин А. Алгоритмы: введение в разработку и анализ – М.: Вильямс, 2006.
2. 2. Кормен, Т., Лейзерсон, Ч., Ривест, Р., Штайн, К. Алгоритмы: построение и анализ – М.: Вильямс, 2005.

6 ПРИЛОЖЕНИЕ

Приложение 1

```
#include <stdint.h>
#include <algorithm>


using namespace std;

void Sort ( int32_t * array, uint32_t size)
{
    if (size <= 1)
        return;
    for (int sorted_index = 0; sorted_index != size-1; sorted_index ++)
    {
        int k = sorted_index+1;
        while (array[k-1] > array[k] & k > 0)
        {
            swap(array[k], array[k-1]);
            k--;
        }
    }
}

int main()
{
    return 0;
}
```