МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
ИНСТИТУТ МЕТАЛЛУРГИИ, МАШИНОСТРОЕНИЯ И ТРАНСПОРТА
ВЫСШАЯ ШКОЛА АВТОМАТИЗАЦИИ И РОБОТОТЕХНИКИ

К У Р С О В О Й П Р О Е К Т

Сортировка counting sort

по дисциплине «Объектно-ориентиованное программирование»

Выполнил
студент группы 3331506/80401 С. А. Коваленко

Руководитель
доцент, к.н. Е. М. Кузнецова

«___»_____2021 г.

Санкт-Петербург

2021

Оглавление

Введение

Сортировка подсчётом (*counting sort*) — алгоритм сортировки, в котором используется диапазон чисел сортируемого массива (списка) для подсчёта совпадающих элементов. Данная сотрировка не использует операции сравнения, и применяется для массива дискретных данных (например, целых чисел, символов). Применение сортировки подсчётом целесообразно лишь тогда, когда сортируемые числа имеют диапазон возможных значений, который достаточно мал по сравнению с сортируемым множеством, например, миллион натуральных чисел меньших 1000.

Алгоритм сортировки и его идея

Основная идея — мы подсчитываем, сколько чисел содержится в каждом классе, то есть считаем, сколько раз встречается то или иное число в массиве. Зная эти количества, быстро формируем уже упорядоченный массив.

Рассмотрим простейший алгоритм. Пусть входной массив состоит из n целых чисел в диапазоне от 0 до *max_element-1*. Далее создаем массив *counting_array[0..max_element-1]* состоящий из нулей, затем последовательно прочитываем элементы входного массива *array*, для каждого *array[i]* увеличивая *counting_array[array[i]]* на единицу. Теперь достаточно пройти по массиву *counting_array*, для каждого *j {0,…, max_element+1}* в массив *array* последовательно записать число *j counting_array[j]* раз.

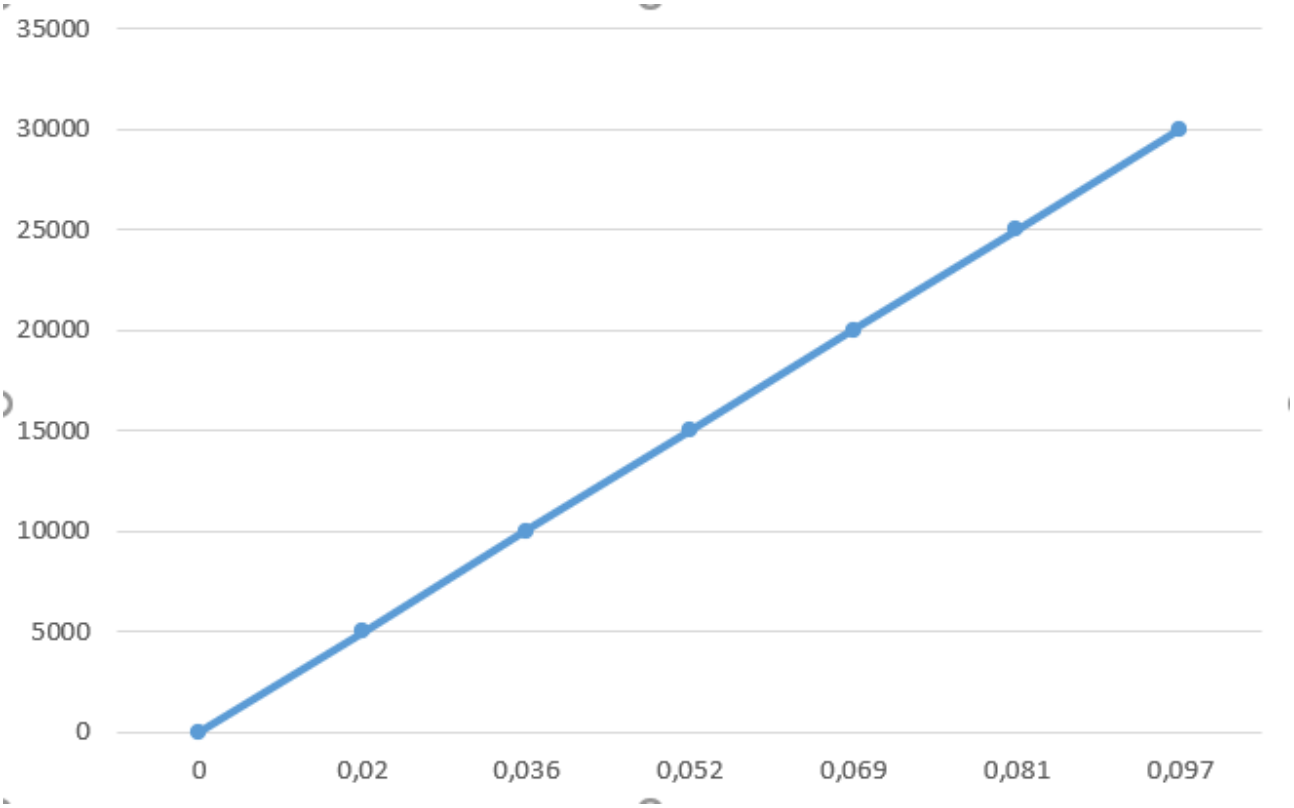
Для данного варианта реализации, необходимо знать диапазон значений. Код простого алгоритма можно посмотреть в приложении 1. Если сделать обобщение на произвольный целочисленный диапазон, то вначале производим поиск максимального и минимального значения. Поскольку индекс массива не может быть отрицательным, приведем минимальное значение к 0. Код обобщенного алгоритма можно посмотреть в приложении 2.

Скорость работы алгоритма

Алгоритм работает за линейное время. Для реализации в приложении 1, первые 2 цикла работают за *O(max_element)* и *O(size)* соответственно, двойной цикл за *O(max_element + size)*. Таким образом алгоритм имеет временную сложность равную *O(max_element + size)*.

Исследование алгоритма

График зависимости времени(ось *x*) сортировки от количества элементов(ось *y*) для алгоритма показанного в приложении 1, представлен на рисунке 1.



Список литературы

1. Левитин А. В. Глава 7. Пространственно-временной компромисс: Сортировка подсчётом // Алгоритмы. Введение в разработку и анализ — М.: Вильямс, 2006. — С. 331—339. — 576 с. — ISBN 978-5-8459-0987-9

2. Кормен, Томас Х., Лейзерсон, Чарльз И., Ривест, Рональд Л., Штайн, Клифорд. Глава 8. Сортировка за линейное время // Алгоритмы: построение и анализ = Introduction to Algorithms. — 2-е издание. — М.: «Вильямс», 2005. — С. 224 - 226. — ISBN 5-8459-0857-4.

Приложения

П1. Реализация алгоритма counting sort на C++

```
void CountingSort(int *array, int size, int max_element)
{
    int *counting_array;
    for(int i = 0; i <= max_element + 1; i++)
    {
        counting_array[i] = 0;
    }
    for(int i = 0; i < size; ++i)
    {
        ++counting_array[array[i]];
    }
    int index = 0;
    for(int i = 0; i < max_element + 1; ++i)
    {
        for (int j = 0; j < counting_array[i]; ++j) {
            array[index] = i;
            index++;
        }
    }
}
```

П2. Реализация улучшенного алгоритма counting sort на C++

```
void CountingSort_Modified(int *array, int size)
{
    int max = INT_MIN, min = INT_MAX;
    for (int i = 0; i < size; i++) {
        if (array[i] > max)
            max = array[i];
        if (array[i] < min)
            min = array[i];
    }
    int *counting_array = new int[max + 1 - min];
    for (int i = 0; i < max + 1 - min; i++) {
        counting_array[i] = 0;
    }
    for (int i = 0; i < size; i++) {
        counting_array[array[i] - min] = counting_array[array[i] - min] + 1;
    }
    int i = 0;
    for (int j = min; j < max + 1; j++) {
        while (counting_array[j-min] != 0) {
            array[i] = j;
            counting_array[j-min]--;
            i++;
        }
    }
}
```