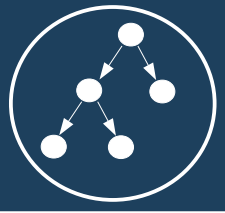


Data Structures and Algorithms

Алгоритмы.
Сортировка подсчетом

Описание сути алгоритма

Сортировка подсчетом используется для сортировки массивов целых чисел, значения которых лежат в относительно узком диапазоне. Например если существует массив целых чисел размером 1000000, значения которых лежат в диапазоне [0..1000], то этот алгоритм покажет очень хорошее быстродействие. Отдельно стоит отметить такую особенность этого алгоритма сортировки, как отсутствие операции сравнения ключей.

Сведение о алгоритме

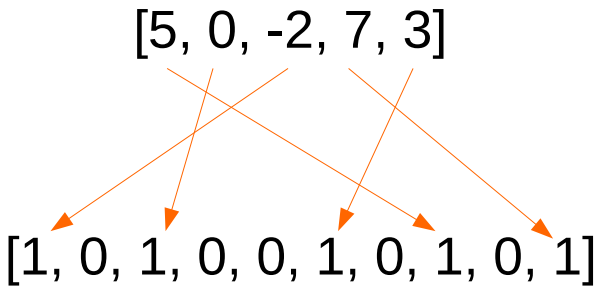
Сложность по времени в наихудшем случае O(n)

Требует дополнительно памяти в размере диапазона чисел

Описание алгоритма

1) Определяем минимальное и максимальное значение в сортируемой последовательности(в дальнейшем sort) (обозначим их как min и max соответственно). Объявляем вспомогательную последовательность (дальнейшем support) длинна которой вычисляется как max-min+1. Заполняем ее нулями.
2) Выполняем проход по sort, добавляем единицу к значению support[element-min] где element это текущий элемент в sort
3) Выполняем проход по индексам (дальше i) последовательности support добавляя в sort значения i+min в количестве support[i]

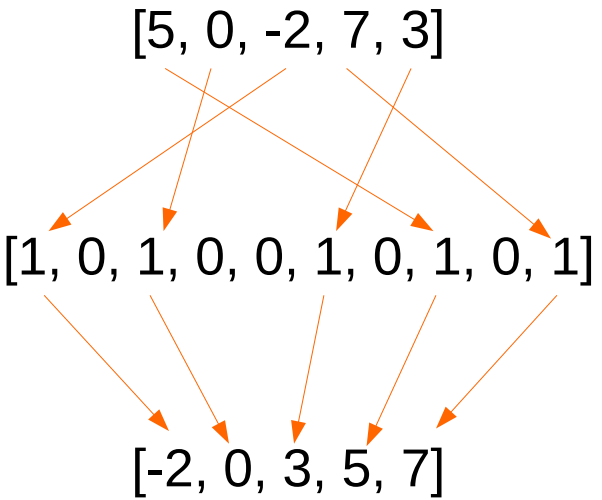
Графическая иллюстрация работы алгоритма



min = -2
max = 7
Длинна вспомогательной последовательности max-min+1 = 7-(-2)+1 = 10

Для элемента основной последовательности (например для 5) вычисляется соответствующий ему индекс во вспомогательной последовательности. Для вычисления используется зависимость вида index = элемент-min. Так, например для 5 получим 5-(-2)==7. После чего увеличиваем значение элемента во вспомогательной последовательности на этом индексе на единицу.

Графическая иллюстрация работы алгоритма



При обратном заполнении выполняется проход по индексам вспомогательной последовательности. Элемент который должен быть добавлен в основную последовательность определяется как индекс+min, а значение которое стоит в вспомогательной последовательности на этом индексе определяет сколько таких элементов должно быть добавлено.

Python

Реализация алгоритма на Python

Python

Реализация алгоритма на Python

```
def counting_sort(sequince):
    min_value = min(sequince)
    max_value = max(sequince)
    support = [0 for i in range(max_value-min_value+1)]
    for element in sequince:
        support[element-min_value] += 1
    index = 0
    for i in range(len(support)):
        for element in range(support[i]):
            sequince[index] = i+min_value
            index += 1
    return None
```

Java

Реализация алгоритма на Java

Реализация алгоритма на Java

Вспомогательный метод для поиска минимума и максимума

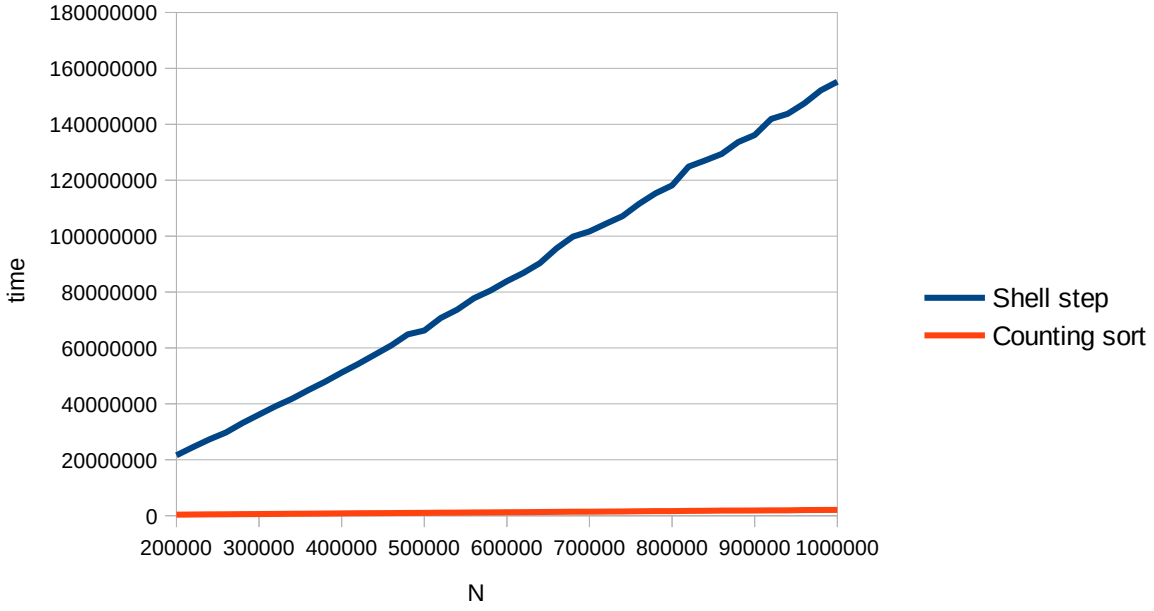
```
public static int[] findMinMax(int[] array) {
    int min = array[0];
    int max = array[0];
    for (int element : array) {
        if (min > element) {
            min = element;
        }
        if (max < element) {
            max = element;
        }
    }
    return new int[] { min, max };
}
```

Реализация алгоритма на Java

```
public static void countingSort(int[] sort) {
    int[] minMax = findMinMax(sort);
    int minValue = minMax[0];
    int maxValue = minMax[1];
    int[] support = new int[maxValue - minValue + 1];
    for (int element : sort) {
        support[element - minValue] += 1;
    }
    int index = 0;
    for (int i = 0; i < support.length; i++) {
        for (int j = 0; j < support[i]; j++) {
            sort[index] = i + minValue;
            index += 1;
        }
    }
}
```

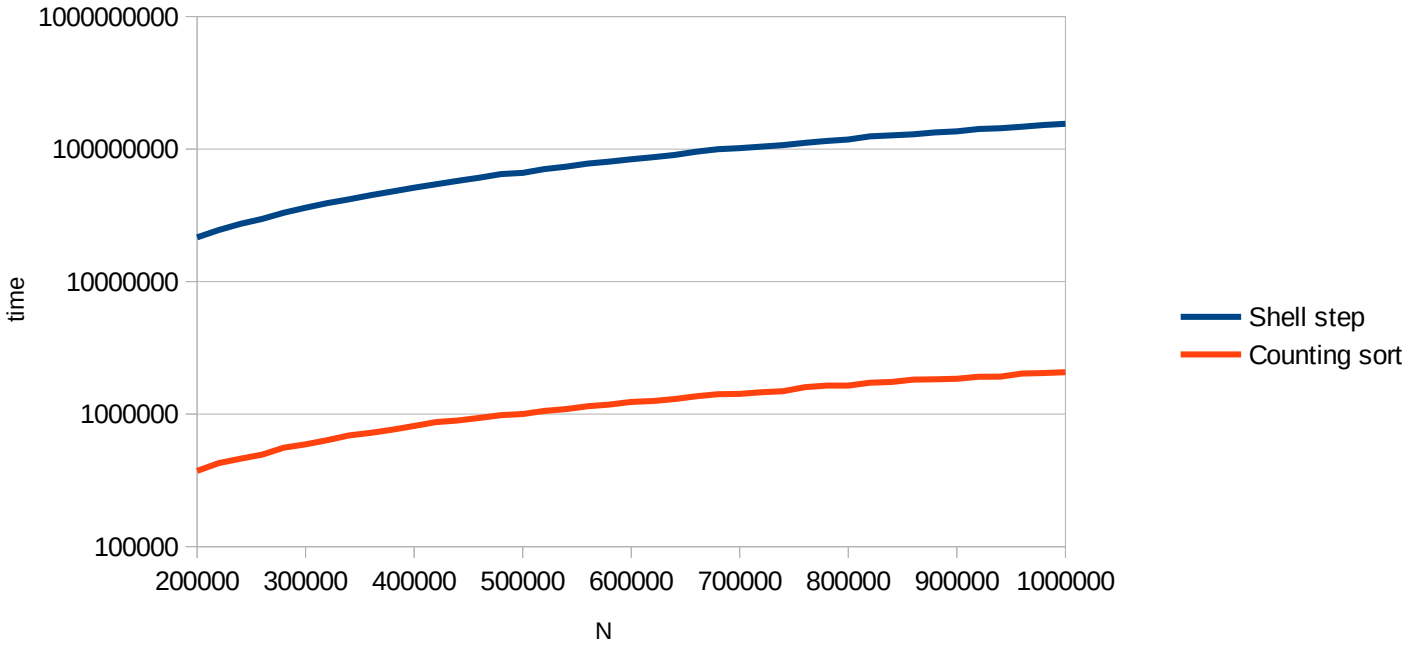
Вычислительный эксперимент

Для проверки увеличения эффективности данного алгоритма по сравнению с алгоритмом сортировки Шеллом был проведен вычислительный эксперимент. Для массивов разных размеров (заполненных случайными числами распределенными в диапазоне 0.01 от размера массива) было замерено среднее время сортировки с помощью того и другого алгоритма. На графике приведена зависимость среднего времени сортировки от размера массива.



Вычислительный эксперимент

Для возможности приблизительной оценки разности скорости работы график был перестроен в логарифмическом масштабе. Как можно видеть в таком случае скорость возрастает в сотни раз.



Список литературы

1) Д. Кнут. Искусство программирования. Том 3. «Сортировка и поиск», 2-е изд. ISBN 5-8459-0082-4