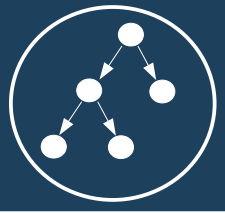


Data Structures and Algorithms

Алгоритмы. Сортировка расческой

Сведение о алгоритме

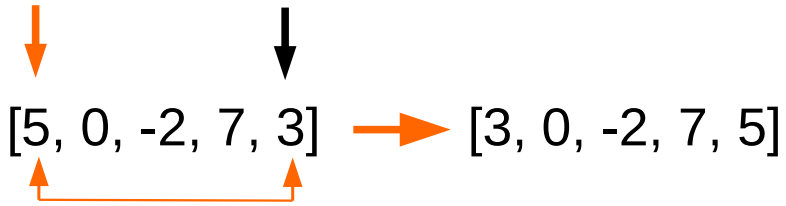
Алгоритм сортировки расческой.

Сложность по времени в наихудшем случае	$O(n^2)$
Затраты памяти	$O(n)$

Принцип работы алгоритма

1) Выбирается начальное значение шага сортировки. Рекомендуемым является шаг равный целой части деления размера последовательности на 1.247. Также добавляем переменную для хранения числа обменов за один проход.
2) Устанавливаем переменную для хранения числа обменов равной нулю. Начиная с первого элемента последовательности сравниваем текущий элемент с элементом, индекс которого равен индексу текущего элемента плюс шаг. Если текущий элемент больше элемента с которым сравниваем производим их обмен и увеличиваем переменную для хранения числа обменов на единицу.
3) В случае если на предыдущем шаге количество обменов равно 0 и шаг равен 1, то заканчиваем алгоритм. В противном случае устанавливаем значение шага равным целой части от деления предыдущего шага на 1.247 (в случае если получается нулевой шаг то устанавливаем его в 1) и возвращаемся у пункту 2.

Графическая иллюстрация работы алгоритма



1)

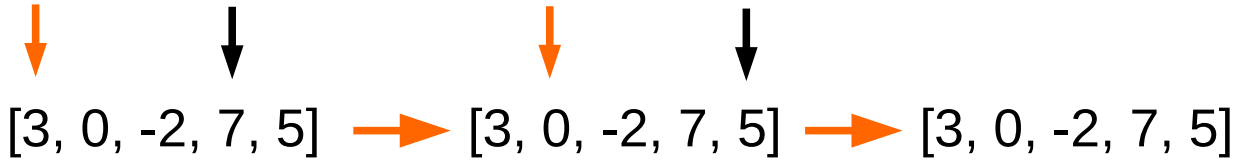
Вычисляем шаг, как целую часть от деления размера последовательности на 1.247

$$step = \left\lfloor \frac{5}{1.247} \right\rfloor = 4$$

Так как обмен был и значение шага больше одного, то вычисляем новое значение шага.

$$step = \left\lfloor \frac{4}{1.247} \right\rfloor = 3$$

Графическая иллюстрация работы алгоритма

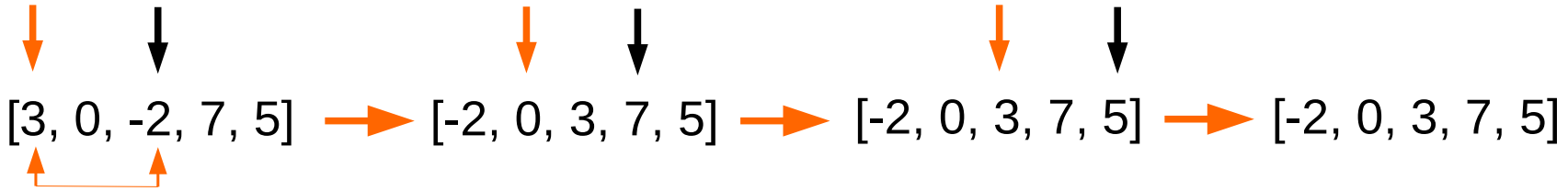


Шаг вычисленный на прошлом шаге равен 3.

Обменов не было, но значение шага больше одного следовательно вычисляем новое значение шага.

$$step = \left\lfloor \frac{3}{1.247} \right\rfloor = 2$$

Графическая иллюстрация работы алгоритма

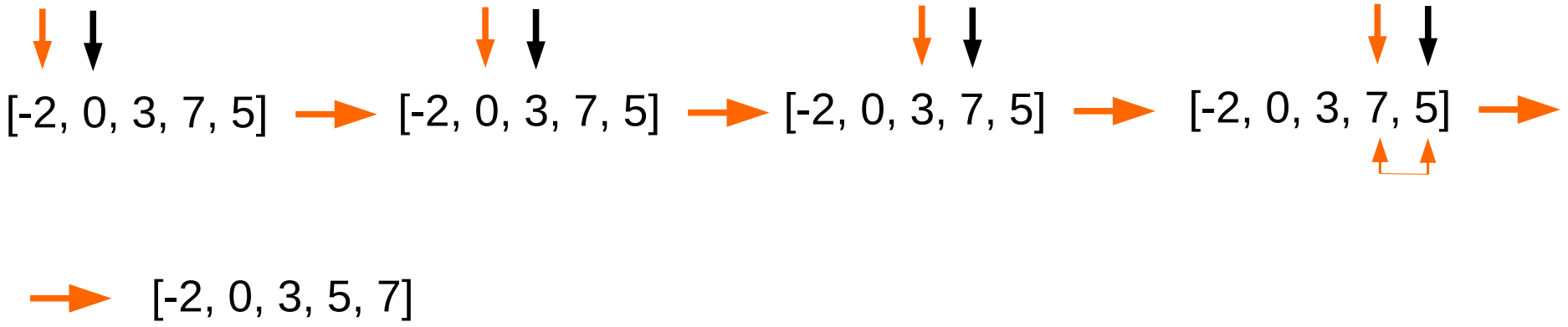


Шаг вычисленный на прошлом шаге равен 2.

Обмены были и значение шага больше одного следовательно вычисляем новое значение шага.

$$step = \left\lfloor \frac{2}{1.247} \right\rfloor = 1$$

Графическая иллюстрация работы алгоритма



Шаг вычисленный на прошлом шаге равен 1.

Обмены были, значение шага равно единице следовательно указываем значение шага равное 1.

Графическая иллюстрация работы алгоритма

Шаг вычисленный на прошлом шаге равен 1.

Обменов не было, значение шага равно единице следовательно заканчиваем алгоритм сортировки.

Python

Реализация алгоритма на Python

Реализация алгоритма на Python

```
def comb_sort(sequince):
    step = int(len(sequince)/1.247)
    swap = 1
    while True:
        swap = 0
        i = 0
        while i + step < len(sequince):
            if sequince[i] > sequince[i+step]:
                sequince[i], sequince[i+step] = sequince[i+step], sequince[i]
                swap += 1
            i = i + 1
        if (step == 1 and swap == 0):
            break
        step = int(step/1.247)
        if step < 1:
            step = 1
```

Java

Реализация алгоритма на Java

Реализация алгоритма на Java

```
public static void combSort(int[] array) {
     int step = (int) (array.length / 1.247);
     int swapCount = 1;
     for (;;) {
          swapCount = 0;
          for (int i = 0; i + step < array.length; i++) {
               if (array[i] > array[i + step]) {
                    int temp = array[i];
                    array[i] = array[i + step];
                    array[i + step] = temp;
                    swapCount++;
               }
          }
          if (step == 1 && swapCount == 0) {
               break;
          }
          step = (int) (step / 1.247);
          if (step < 1) {
               step = 1;
          }
     }
}
```

Список литературы

1) Lacey S., Box R. A Fast, Easy Sort: A novel enhancement makes a bubble sort into one of the fastest sorting routines (англ.) // Byte. — Апрель 1991. — Vol. 16, no. 4. — P. 315—320. — ISSN 0360-528