

# Сортировка расчёской

Лев Захаров

Высшая школа ИТИС КФУ

Казань, 2015

# Содержание

- 1 Вступление
- 2 Алгоритм
- 3 Реализация
- 4 Оценка сложности
- 5 Вывод
- 6 Список литературы

## Вступление

Сортировка пузырьком, пожалуй, самая простая и известная сортировка. Алгоритм является простым для понимания и легко реализуемым. Однако данная сортировка эффективна лишь для небольших массивов. Тем не менее, существуют модификации, позволяющие ускорить работу алгоритма. Одной из таких модификаций является **сортировка расческой** или **comb sort**.

## Во всем виноваты черепашки

В 1980 году Влодзимеж Добосиевич пояснил почему пузырьковая и производные от неё сортировки работают так медленно. *Это всё из-за черепашек.*

## Во всем виноваты черепашки

В 1980 году Влодзимеж Добосиевич пояснил почему пузырьковая и производные от неё сортировки работают так медленно. *Это всё из-за черепашек.*

**Черепаха** — элемент с относительно маленьким значением, находящийся в конце списка.

## Во всем виноваты черепашки

В 1980 году Влодзимеж Добосиевич пояснил почему пузырьковая и производные от неё сортировки работают так медленно. *Это всё из-за черепашек.*

**Черепаха** — элемент с относительно маленьким значением, находящийся в конце списка.

**Кролик** — элемент с относительно большим значением, находящийся в начале списка.

## Во всем виноваты черепашки

В 1980 году Влодзимеж Добосиевич пояснил почему пузырьковая и производные от неё сортировки работают так медленно. *Это всё из-за черепашек.*

**Черепаха** — элемент с относительно маленьким значением, находящийся в конце списка.

**Кролик** — элемент с относительно большим значением, находящийся в начале списка.

В процессе сортировки черепашки сдвигаются только на одну позицию за один проход. С другой стороны *кролики* двигаются достаточно быстро.

## Сортировка расческой

Основная идея сортировки расческой заключается в том, чтобы первоначально выбирать расстояние между сравниваемыми элементами больше единицы.



## Сортировка расческой

Основная идея сортировки расческой заключается в том, чтобы первоначально выбирать расстояние между сравниваемыми элементами больше единицы.

Для первого прохода шаг равен частному от деления размера массива на **фактор уменьшения**.

## Сортировка расческой

Основная идея сортировки расческой заключается в том, чтобы первоначально выбирать расстояние между сравниваемыми элементами больше единицы.

Для первого прохода шаг равен частному от деления размера массива на **фактор уменьшения**.

На каждом последующем шаге расстояние между сравниваемыми элементами делится на **фактор уменьшения**.

## Сортировка расческой

Основная идея сортировки расческой заключается в том, чтобы первоначально выбирать расстояние между сравниваемыми элементами больше единицы.

Для первого прохода шаг равен частному от деления размера массива на **фактор уменьшения**.

На каждом последующем шаге расстояние между сравниваемыми элементами делится на **фактор уменьшения**.

Так продолжается до тех пор, пока разность индексов сравниваемых элементов не достигнет единицы. Далее массив досортировывается пузырьком.

## Фактор уменьшения

Опытным и теоретическим путем было установлено оптимальное значение **фактора уменьшения**:

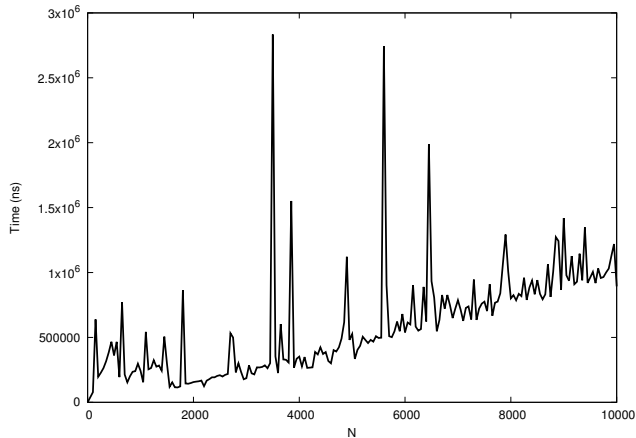
$$\frac{1}{1 - \frac{1}{e^\varphi}} \approx 1.247330950103979$$

, где  $\varphi$  есть золотое сечение, т.е.  $\varphi = \frac{1+\sqrt{5}}{2}$ .

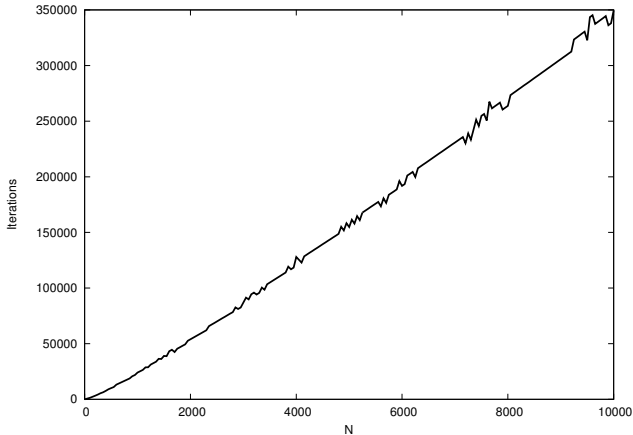
# Реализация

```
1  public static int[] combSort(int[] a) {
2      int gap = a.length;
3      boolean swapped = true;
4
5      while (gap > 1 || swapped) {
6          if (gap > 1.247330950103979) {
7              gap = (int)(gap / 1.247330950103979);
8          }
9
10         int i = 0;
11         swapped = false;
12
13         while (i + gap < a.length) {
14             if (a[i] > a[i + gap]) {
15                 int k = a[i];
16                 a[i] = a[i + gap];
17                 a[i + gap] = k;
18                 swapped = true;
19             }
20
21             i++;
22         }
23     }
24
25     return a;
26 }
```

# Зависимости времени работы от размера массива



# Зависимость итераций от размера массива



# Вычислительная сложность алгоритма

Вычислительная сложность сортировки расческой в среднем равна  $O(n \log n)$ , и  $O(n^2)$  в худшем случае.



# Вычислительная сложность алгоритма

Вычислительная сложность сортировки расческой в среднем равна  $O(n \log n)$ , и  $O(n^2)$  в худшем случае.

Сортировка расческой конкурирует с алгоритмами, подобными быстрой сортировке.

## Вывод

Алгоритм сортировки расческой в среднем работает значительно быстрее сортировки пузырьком и на некоторых входных данных опережает по сложности быстрые сортировки. Однако алгоритм является неустойчивым и на некоторых наборах происходит деградация до квадратичной сложности.

## Вывод

Алгоритм сортировки расческой в среднем работает значительно быстрее сортировки пузырьком и на некоторых входных данных опережает по сложности быстрые сортировки. Однако алгоритм является неустойчивым и на некоторых наборах происходит деградация до квадратичной сложности.

Данный алгоритм выгодно применять в тех случаях, когда требуется один раз отсортировать некоторые данные, так как эта сортировка легка для понимания и быстро реализуема.

## Список литературы



Пузырьковая сортировка и все-все-все

<http://habrahabr.ru/post/204600/>



**A Fast Easy Sort**, Byte Magazine, April 1991