

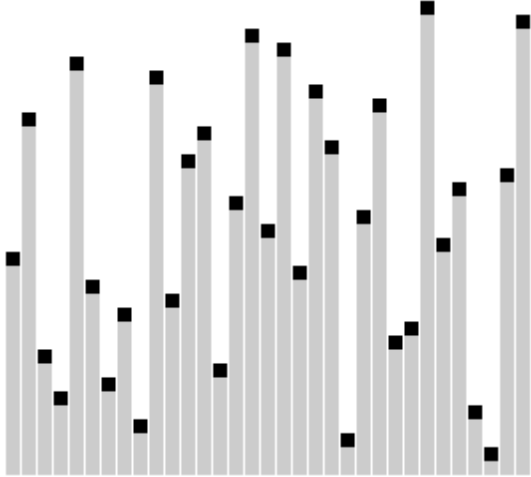
# Сортировка расчёской

Материал из Википедии — свободной энциклопедии

**Сортировка расчёской** (англ. *comb sort*) — это довольно упрощённый алгоритм сортировки, изначально спроектированный Влодзимежом Добосевичем в 1980 г. Позднее он был переоткрыт и популяризован в статье Стивена Лэйси и Ричарда Бокса в журнале *Byte Magazine* в апреле 1991 г<sup>[1]</sup>. Сортировка расчёской улучшает сортировку пузырьком, и конкурирует с алгоритмами, подобными быстрой сортировке. Основная идея — устранить *черепаш*, или маленькие значения в конце списка, которые крайне замедляют сортировку пузырьком (*кролики*, большие значения в начале списка, не представляют проблемы для сортировки пузырьком).

В сортировке пузырьком, когда сравниваются два элемента, промежуток (расстояние друг от друга) равен 1. Основная идея сортировки расчёской в том, что этот промежуток может быть гораздо больше, чем единица (сортировка Шелла также основана на этой идее, но она является модификацией сортировки вставками, а не сортировки пузырьком).

### Сортировка расчёской



Визуализация сортировки расчёской

<b>Предназначение</b>	<span></span> <span>Алгоритм сортировки</span>
<b>Худшее время</b>	$O(n^2)$
<b>Лучшее время</b>	$O(n \log n)$
<b>Среднее время</b>	$\Omega(n^2 / 2^p)$
<b>Затраты памяти</b>	$O(1)$

## Содержание

- Алгоритм
- Реализация на Паскаль
- Реализация на C++
- Реализация на Java
- Реализация на PHP
- Реализация на Python
- Реализация на JavaScript
- Примечания

## Алгоритм

В «пузырьке», «шейкере» и «чёт-нечете» при переборе массива сравниваются соседние элементы. Основная идея «расчёски» в том, чтобы первоначально брать достаточно большое расстояние между сравниваемыми элементами и по мере упорядочивания массива сужать это расстояние вплоть до минимального. Таким образом, мы как бы причёсываем массив, постепенно разглаживая на всё более аккуратные пряди. Первоначальный разрыв между сравниваемыми элементами лучше брать с учётом специальной величины,

называемой фактором уменьшения, оптимальное значение которой равно примерно 1,247. Сначала расстояние между элементами равно размеру массива, разделённого на фактор уменьшения (результат округляется до ближайшего целого). Затем, пройдя массив с этим шагом, необходимо поделить шаг на фактор уменьшения и пройти по списку вновь. Так продолжается до тех пор, пока разность индексов не достигнет единицы. В этом случае массив досортировывается обычным пузырьком.

Оптимальное значение фактора уменьшения  $1,247\dots = \frac{1}{1-e^{-\phi}}$ , где  $e$  — основание натурального логарифма, а  $\phi$  — золотое сечение.

## Реализация на Паскаль

1. Заполняю массив случайными числами.
2. Завожу цикл с условием « $i < i + j$ », которое буквально означает « $i$  отличается от  $i + j$ ».
  1. Обнуляю  $i$  для того, чтобы при новом пробеге по массиву индекс не вышел за его границы.
  2. Завожу внутренний цикл с условием « $i + j \leq n$ », которое буквально значит «сумма индекса  $i$  и расстояния  $j$  между  $a[i]$  и другим сравниваемым элементом не больше, чем самый большой индекс массива».
    1. Если  $a[i] > a[i + j]$ , то меняю их местами.
    2. Увеличиваю  $i$ .
  3. Уменьшаю  $j$ .

```
const
  n = 5;

var
  a: array [0..n] of integer;
  i, jr: integer;
  j: real;

begin
  for i := 0 to n do a[i] := Random(12);
  j := n;
  jr := Round(j);
  while i < i + jr do
  begin
    i := 0;
    jr := Round(j);
    while i + j <= n do
    begin
      if a[i] > a[i + Round(j)] then
      begin
        a[i] := a[i] + a[i + jr];
        a[i + jr] := a[i] - a[i + jr];
        a[i] := a[i] - a[i + jr];
      end;
      Inc(i);
    end;
    j := j / 1.247;
  end;

  for i := 0 to n do
  begin
    for jr := 0 to i - 1 do
    begin
      if a[jr] > a[jr + 1] then
      begin
        a[jr] := a[jr] + a[jr + 1];
        a[jr + 1] := a[jr] - a[jr + 1];
        a[jr] := a[jr] - a[jr + 1];
      end;
    end;
  end;
  WriteLn(a);
end.
```

Цикл прекратится лишь тогда, когда  $j$  станет равной 0, иначе говоря, когда  $i$  станет равным  $i + j$ .

## Реализация на C++

```

int comb(vector<double> sort)// sort-название массива, size-размер массива,если нужно ввести массив с клавиатуры,
то нужно создать динамический массив
{
    int n = 0; // количество перестановок
    double fakt = 1.2473309; // фактор уменьшения
    double step = sort.size() - 1;

    while (step >= 1)
    {
        for (int i = 0; i + step < sort.size(); ++i)
        {
            if (sort[i] > sort[i + step])
            {
                swap(sort[i], sort[i + step]);
                n++;
            }
        }
        step /= fakt;
    }
    // сортировка пузырьком
    for (int i = 0; i < sort.size() - 1; i++)
    {
        bool swapped = false;
        for (int j = 0; j < sort.size() - i - 1; j++)
        {
            if (sort[j] > sort[j + 1]) {
                swap(sort[j], sort[j + 1]);
                swapped = true;
                ++n;
            }
        }

        if (!swapped)
            break;
    }
    return n;
}

```

## Реализация на Java

```

public static <E extends Comparable<? super E>> void sort(E[] input) {
    int gap = input.length;
    boolean swapped = true;
    while (gap > 1 || swapped) {
        if (gap > 1)
            gap = (int) (gap / 1.247330950103979);

        int i = 0;
        swapped = false;
        while (i + gap < input.length) {
            if (input[i].compareTo(input[i + gap]) > 0) {
                E t = input[i];
                input[i] = input[i + gap];
                input[i + gap] = t;
                swapped = true;
            }
            i++;
        }
    }
}

```

## Реализация на PHP

```

function combsort($array)
{
    $sizeArray = count($array);

    // Проходимся по всем элементам массива

```

```

for ($i = 0; $i < $sizeArray; $i++) {
    // Сравниваем попарно.
    // Начинаем с первого и последнего элемента, затем постепенно уменьшаем
    // диапазон сравниваемых значений.
    for ($j = 0; $j < $i + 1; $j++) {

        // Индекс правого элемента в текущей итерации сравнения
        $elementRight = ($sizeArray - 1) - ($i - $j);

        if ($array[$j] > $array[$elementRight]) {

            $buff                = $array[$j];
            $array[$j]           = $array[$elementRight];
            $array[$elementRight] = $buff;
            unset($buff);

        }
    }
}

return $array;
}

```

## Реализация на Python

```

def combsort(alist):
    alen = len(alist)
    gap = (alen * 10 // 13) if alen > 1 else 0
    while gap:
        if 8 < gap < 11:    ## variant "comb-11"
            gap = 11
        swapped = False
        for i in range(alen - gap):
            if alist[i + gap] < alist[i]:
                alist[i], alist[i + gap] = alist[i + gap], alist[i]
                swapped = True
        gap = (gap * 10 // 13) or swapped

```

## Реализация на JavaScript

```

function combSorting(array) {
    const factor = 1.247;
    let gapFactor = array.length / factor;

    while (gapFactor > 1) {
        const gap = Math.round(gapFactor);
        for (let i = 0, j = gap; j < array.length; i++, j++) {
            if (array[i] >= array[j]) {
                [ array[i], array[j] ] = [ array[j], array[i] ];
            }
        }
        gapFactor = gapFactor / factor;
    }

    return array;
}

```

## Примечания

- ↑ 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 (https://www.worldcat.org/search?fq=x0:jrnl&q=n2:0360-528).

---

**Эта страница в последний раз была отредактирована 13 января 2019 в 18:38.**

Текст доступен по лицензии [Creative Commons Attribution-ShareAlike](#); в отдельных случаях могут действовать дополнительные условия.

Wikipedia® — зарегистрированный товарный знак некоммерческой организации [Wikimedia Foundation, Inc.](#)