




odd-even sort

Estrutura de dados

Asafe Damasceno

Thiago Gondin Paulo



É um **algoritmo de ordenação**
por comparação baseado no
bubble-sort



Como funciona ?

Este algoritmo é dividido em **duas fases**: fase **ímpar** e **par**.

Na fase ímpar, realizamos um tipo de bolha em elementos indexados ímpares e na fase dupla, realizamos um tipo de bolha em elementos indexados.

Pode ser pensado como a utilização de processadores paralelos, cada qual usando um BubbleSort, mas a partir de diferentes pontos na lista (todos os índices ímpares para a primeira etapa).

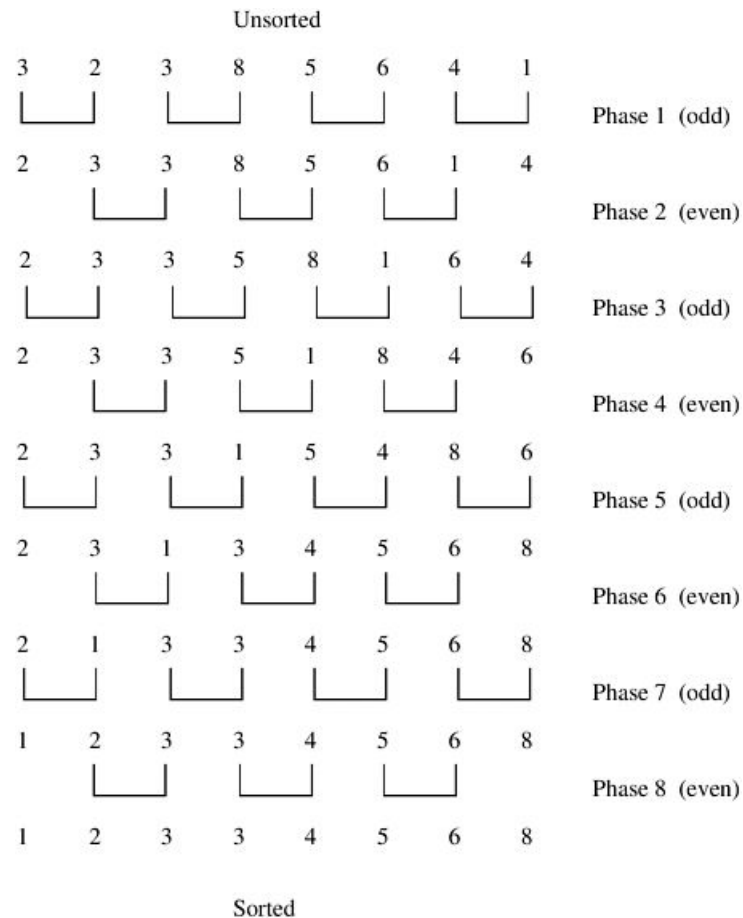
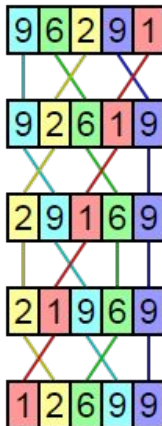
Demonstração

Vídeo

<https://youtu.be/AYehFa-oKCA>

Jogo

<http://www.algostructure.com/sorting/oddevensort.php>



Implementação C

```
// A function to sort the algorithm using Odd Even sort
void oddEvenSort(int arr[], int n)
{
    bool isSorted = false; // Initially array is unsorted

    while (!isSorted)
    {
        isSorted = true;

        // Perform Bubble sort on odd indexed element
        for (int i=1; i<=n-2; i=i+2)
        {
            if (arr[i] > arr[i+1])
            {
                swap(arr[i], arr[i+1]);
                isSorted = false;
            }
        }

        // Perform Bubble sort on even indexed element
        for (int i=0; i<=n-2; i=i+2)
        {
            if (arr[i] > arr[i+1])
            {
                swap(arr[i], arr[i+1]);
                isSorted = false;
            }
        }
    }

    return;
}
```

Implementação Java

```
public class OddEvenSort {
    public static <T extends Comparable<T>> void sort(T[] array) {
        boolean sorted = false;
        while (!sorted) {
            sorted = innerSort(array, 1);
            sorted = innerSort(array, 0) && sorted;
        }
    }

    private static <T extends Comparable<T>> boolean innerSort(T[] array, Integer i) {
        boolean sorted = true;
        for (; i < array.length - 1; i += 2)
        {
            if (array[i].compareTo(array[i + 1]) > 0)
            {
                swap(array, i, i + 1);
                sorted = false;
            }
        }
        return sorted;
    }

    private static <T extends Comparable<T>> void swap(
        T[] array, int a, int b) {
        T temp = array[a];
        array[a] = array[b];
        array[b] = temp;
    }
}
```

Implementação Python

```
def default_compare(a, b):
    if a < b:
        return -1
    elif a > b:
        return 1
    return 0

def sort(array, compare=default_compare):
    sorted = False
    while not sorted:
        sorted = inner_sort(array, 1, compare)
        sorted = inner_sort(array, 0, compare) and sorted
    return array

def inner_sort(array, start_i, compare):
    sorted = True
    for i in range(start_i, len(array) - 1, 2):
        if compare(array[i], array[i + 1]) > 0:
            array[i], array[i + 1] = array[i + 1], array[i]
            sorted = False
    return sorted
```



Complexidade

Tempo			Espaço
Pior caso	Melhor caso	Caso médio	Pior caso
$O(n^2)$	$\Theta(n)$	$O(n^2)$	$O(1)$ auxiliar



Quando é rápido ?

O tipo odd-even é rápido quando todos os elementos da matriz de entrada estão próximos dos índices classificados





Quando é lento ?

Se o maior item na matriz estava no início, o tipo impar-par funcionaria no seu pior caso em termos de iterações e comparações.



Referências

<http://www.growingwiththeweb.com/2016/10/odd-even-sort.html>

https://pt.wikipedia.org/wiki/Odd-even_sort

<http://www.geeksforgeeks.org/odd-even-sort-brick-sort/>




shaker sort ou bubble sort bidirecional

Estrutura de dados

Asafe Damasceno

Thiago Gondin Paulo



É um **algoritmo de ordenação** por
comparação baseado no **bubble-sort**,
contudo, bidirecional.



Como funciona ?

- Corre o vetor de forma Bilateral
- Deslocando elemento tanto para parte superior como inferior do vetor
- Resolve o problema do bubble-sort "rabbits and turtles"



Demonstração

<http://www.algostructure.com/sorting/cocktailsort.php>



Implementação C++

```
01. void shakerSort(int array[], int size) {
02.     for (int i = 0; i < size/2; i++) {
03.         bool swapped = false;
04.         for (int j = i; j < size - i - 1; j++) { //one way
05.             if (array[j] < array[j+1]) {
06.                 int tmp = array[j];
07.                 array[j] = array[j+1];
08.                 array[j+1] = tmp;
09.                 swapped = true;
10.             }
11.         }
12.         for (int j = size - 2 - i; j > i; j--) { //and back
13.             if (array[j] > array[j-1]) {
14.                 int tmp = array[j];
15.                 array[j] = array[j-1];
16.                 array[j-1] = tmp;
17.                 swapped = true;
18.             }
19.         }
20.         if(!swapped) break; //block (break if no element was swapped - the array is sorted)
21.     }
21. }
```




Complexidade

Time			Space
Worst case	Best case	Average case	Worst case
$O(n^2)$	$O(n)$	$O(n^2)$	$O(1)$ auxiliary



Referências

<http://www.programming-algorithms.net/article/40270/Shaker-sort>

https://pt.wikipedia.org/wiki/Cocktail_sort

<http://www.growingwiththeweb.com/2016/04/cocktail-sort.html>