Bubble Sort

Apresentação por
Rafael Cunha

Bubble Sort

Bubble Sort é um algoritmo de organização que recebe um Array somente de números e ordena eles em ordem crescente

É importante lembrar que um algoritmo é um conjunto de processos que o computador vai executar para chegar em um determinado resultado

Ele é considerado uma algoritmo lento, já que ele verifica todos os números presentes no Array mesmo sem necessidade

Bubble Sort

Vamos imaginar um Array:

var arr = [1, 5, 3, 2, 4]

O algoritmo vai receber esse array e verificar o primeiro número com o segundo

Se o primeiro for maior que o segundo então eles invertem suas posições

Array atual não foi mudado:

var arr = [1, 5, 3, 2, 4]

Bubble Sort

var arr = [1, 5, 3, 2, 4]

Depois ele vai verificar o segundo número com o terceiro

5 é maior que 3, então vão trocar suas posições

Novo Array gerado:

var arr = [1, 3, 5, 2, 4]

Bubble Sort

A partir daí ele vai começar a verificar n com n+1:

var arr = [1, 3, 5, 2, 4] → var arr = [1, 3, 2, 5, 4]

var arr = [1, 3, 2, 5, 4] → var arr = [1, 3, 2, 4, 5]

Podemos perceber que o 5 chegou no final do vetor, e podemos concluir que o algoritmo encontrou o maior número e levou ele até a última posição (O número 5 flutuou como uma bolha)

Agora o algoritmo vai fazer a mesma coisa para todos os números que restaram dessa série de 5

Podemos dizer que primeiro o algoritmo flutua um número e depois vai em busca de flutuar o próximo

Bubble Sort

var arr = [1, 3, 2, 4, 5] → var arr = [1, 3, 2, 4, 5] → var arr = [1, 2, 3, 4, 5]

Array final e organizado em ordem crescente:

var arr = [1, 2, 3, 4, 5]

Vamos aplicar em JavaScript?

Bubble Sort

Primeiro precisamos fazer uma condicional para ver se o valor de N é maior que N+1

```
if (arr[N] > arr[N+1]) {
        aux = arr[N]
        arr[N] = arr[N+1]
        arr[N+1] = aux
}
```

Precisamos dessa variável “aux” para conseguirmos atribuir um valor depois de ter atribuido o outro, já que o computador esqueceria o valor original do primeiro valor atrelado para atrelar no segundo

Esse N significa a posição que estamos verificando no momento, e como temos 5 posições precisamos utilizar um loop que faz essa verificação com todas as posições do Array

Array original:

var arr = [1, 5, 3, 2, 4]

Resultado (Considerando N = 1):

var arr = [1, 3, 5, 2, 4]

Esse resultado aconteceu já que ele fez somente uma verificação, então o valor de N precisa aumentar para verificar as próximas posições

Bubble Sort

```
for (let N = 0; N < arr.length; N++) {
        if (arr[N] > arr[N+1]) {
                aux = arr[N]
                arr[N] = arr[N+1]
                arr[N+1] = aux
        }
}
```

Ou seja, N vai começar como 0 e enquanto N for menor que o tamanho do Array (para encontrar o tamanho do array utilizamos o método ".length") ele vai adicionar mais um à N para ir avançando os números que estão sendo verificados

Array original:

var arr = [1, 5, 3, 2, 4]

Resultado (N começa como 0, mas vai aumentando de 1 em 1 até chegar a 5*):

var arr = [1, 3, 2, 4, 5]

O Array ainda não está ordenado, já que apenas flutuamos o número 5 e precisamos flutuar todos até sua posição ordenada

* 5 é o valor de "arr.length"

Bubble Sort

```
for (let i = 0; i < arr.length; i++) {
        for (let N = 0; N < arr.length; N++) {
                if (arr[N] > arr[N+1]) {
                        aux = arr[N]
                        arr[N] = arr[N+1]
                        arr[N+1] = aux
                }
        }
}
```

O código anterior foi colocado dentro de um outro loop. Esse novo loop vai fazer com que o algoritmo execute a primeira onda de verificação (no nosso array de exemplo, essa verificação é flutuar o 5 até o final do Array) que em seguida fará a segunda onda de verificação, num total de verificações com o tamanho do Array, simbolizando que todos os números dentro do Array serão utilizados

Perceba que essa variável “i” será utilizada apenas para contagem de ondas de verificação

Array original:

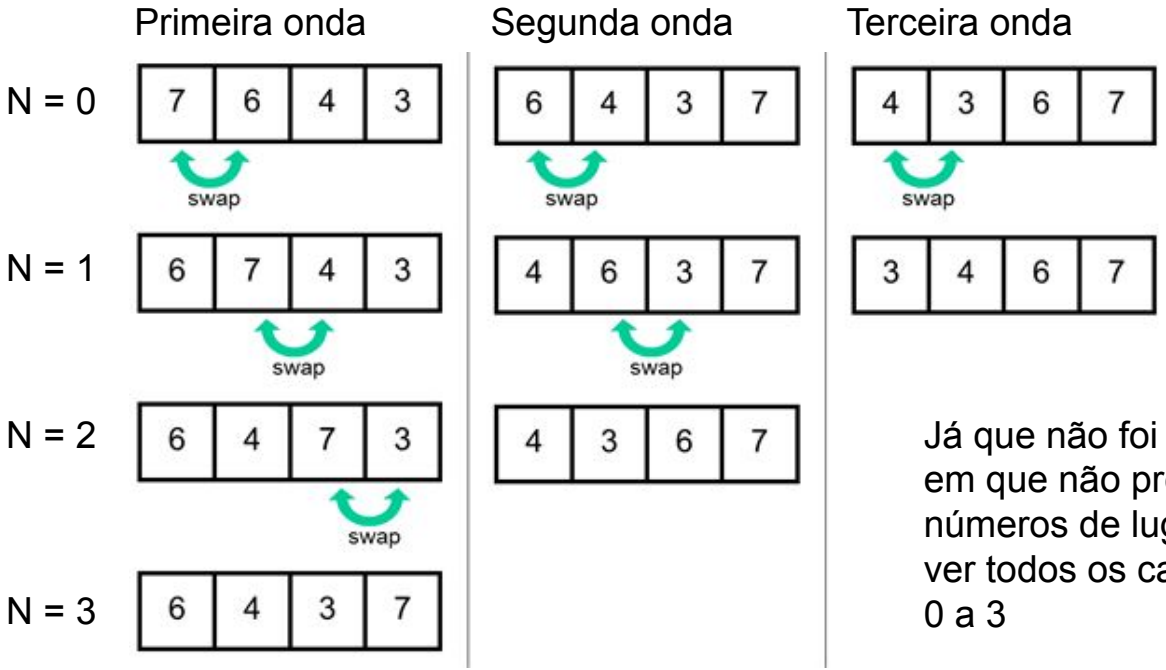
var arr = [1, 5, 3, 2, 4]

Resultado:

var arr = [1, 2, 3, 4, 5]

Bubble Sort

Outro exemplo simples de Bubble Sort:

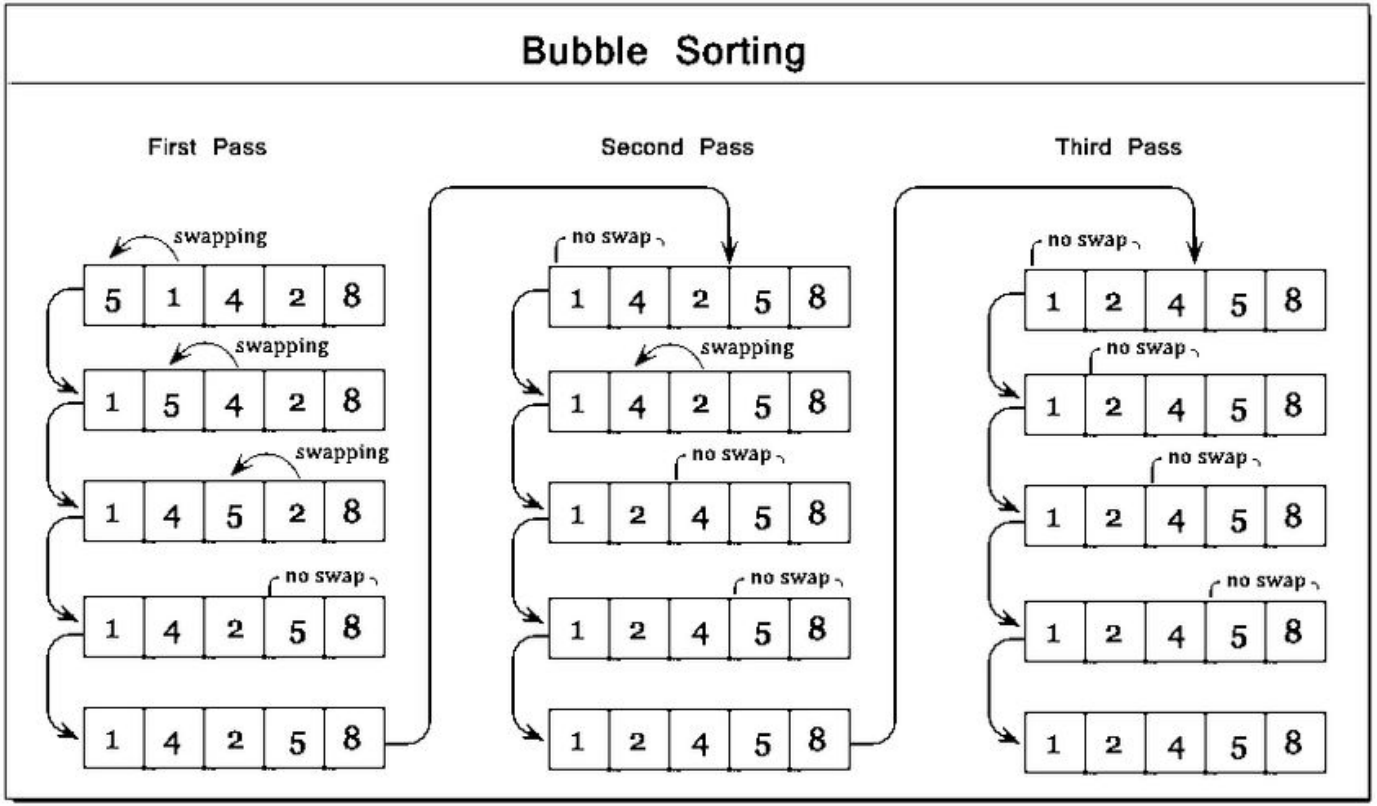


Já que não foi mostrado as vezes em que não precisaria trocar os números de lugar, não foi possível ver todos os casos em que N vai de 0 a 3

Source

Bubble Sort

Nesse exemplo é possível ver todos os momentos de N em todas as ondas (Pass = ondas)



Source

Bubble Sort

Vamos para a prática!