Estruturas de Dados

Aula 4.1 – Algoritmos de Ordenação - Insertion Sort e Bubble Sort

Prof. Ana Carolina Sokolonski

Bacharelado em Sistemas de Informação
Instituto Federal da Bahia – Campus Feira de Santana

2026

Agenda

Insertion Sort

Ordenação por Inserção

Insertion Sort — Conceito

Como funciona

Dado $v = \{a_1, a_2, \ldots, a_n\}$, o algoritmo realiza uma **inserção ordenada** dos elementos de forma que o vetor resultante satisfaz $b_1 \leq b_2 \leq \cdots \leq b_n$. [Cormen et al. 2009]

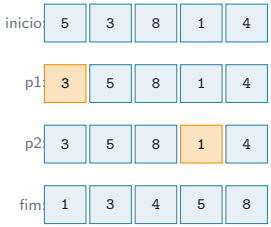
Complexidade

- Pior caso: $O(N^2)$
- Melhor caso: $O(N)$ (vetor já ordenado)

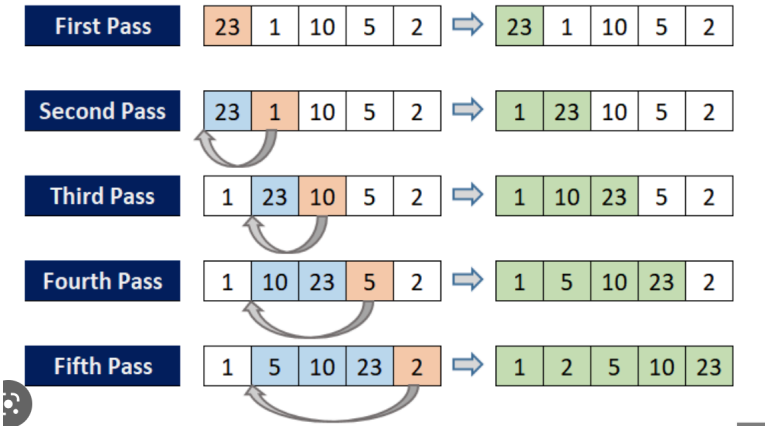
Eficiente para **pequenas quantidades** de elementos.

Vídeo: `https://youtu.be/EdIKIf9mHk0`

Exemplo passo a passo



Insertion Sort — Visualização



Insertion Sort — Pseudocódigo

Algorithm 1 `void insertionSort(int tam, int v[])`

```
1: for i = 1 to N do
2:     j = i
3:     while (j > 0) and (V[j] < V[j − 1]) do
4:         aux = V[j]
5:         V[j] = V[j − 1]
6:         V[j − 1] = aux
7:         j−−
8:     end while
9: end for
```

Intuição

Para cada elemento $V[i]$, percorre o vetor para a **esquerda** trocando-o com elementos maiores, até encontrar sua posição correta — como ordenar cartas na mão.

Invariante

Após a iteração i, o subvetor $V[0\,..\,i]$ está **ordenado**.

Insertion Sort — Código em C

```c
void insertionSort(int tam, int v[]){
    int j,aux;
    for(int i=1; i<tam; i++){
        j = i;
        while((j>0) && (v[j]<v[j-1])){
            aux=v[j];
            v[j]=v[j-1];
            v[j-1]=aux;
            j--;
        }
    }
}
```

Bubble Sort

Ordenação por Comparação

Bubble Sort — Conceito

Como funciona

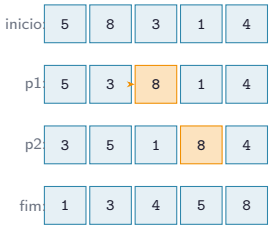
Compara pares adjacentes e, se estiverem fora de ordem, **troca-os**. Repete o processo até percorrer o vetor inteiro sem nenhuma troca. [R. Sedgewick and K. Wayne 2011]

Complexidade

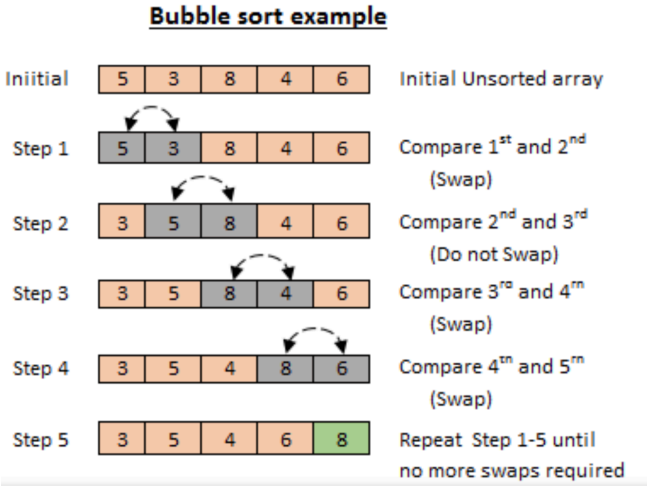
- Pior caso: $O(N^2)$
- Melhor caso: $O(N)$ (nenhuma troca necessária)

Vídeo: `https://youtu.be/Iv3vgjM8Pv4`

"Bolha" sobe para o topo



Bubble Sort — Visualização



Bubble Sort — Pseudocódigo

Algorithm 2 `void bubbleSort(int tam, int v[])`

```
1:  flag = true
2:  while flag do
3:      flag = false
4:      for i = 1 to N do
5:          if V[i − 1] > V[i] then
6:              flag = true
7:              aux = V[i]
8:              V[i] = V[i − 1]
9:              V[i − 1] = aux
10:         end if
11:     end for
12: end while
```

O papel do `flag`

A variável `flag` é o mecanismo de parada antecipada. Se uma passagem inteira não gerar nenhuma troca, `flag` permanece `false` e o loop encerra — o vetor já está ordenado.

Isso garante o melhor caso $O(N)$.

Bubble Sort — Código em C

```c
void bubbleSort(int tam, int v[]){
    int aux,flag = 1;
    while (flag){
        flag = 0;
        for(int i=1; i<tam; i++){
            if (v[i-1] > v[i]){
                flag = 1;
                aux=v[i];
                v[i]=v[i-1];
                v[i-1]=aux;
            }
        }
    }
}
```

Comparação

Insertion Sort vs. Bubble Sort

Insertion Sort vs. Bubble Sort

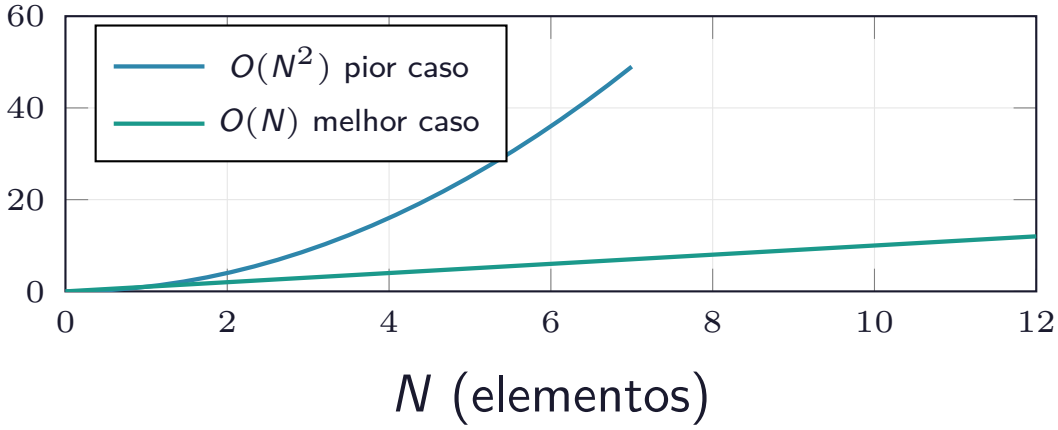
Insertion Sort

- Percorre à **esquerda** trocando com maiores
- Invariante: $V[0..i]$ ordenado após passo i
- **Bom** para vetores quase ordenados
- Pior: $O(N^2)$ Melhor: $O(N)$

Bubble Sort

- Compara **pares adjacentes** e troca se fora de ordem
- Para quando nenhuma troca é necessária (`flag`)
- **Simples** de implementar, pouco usado na prática
- Pior: $O(N^2)$ Melhor: $O(N)$

Insertion Sort vs. Bubble Sort



Referências I

CORMEN, T. H. et al. *Introduction to Algorithms*. 2nd. ed. [S.l.]: The MIT Press, 2009. ISBN 0262032937.

R. Sedgewick and K. Wayne. *Algorithms*. 4th edition. ed. [S.l.]: Addison-Wesley, 2011. v. 4.