





UNIVERSIDADE FEDERAL DE GOIÁS INSTITUTO DE INFORMÁTICA BACHARELADO EM ENGENHARIA DE SOFTWARE

ORDENAÇÃO POR INSERÇÃO

Goiânia

1. INTRODUÇÃO

Antes de mais nada, a ordenação de elementos consiste basicamente em colocá-los em ordem crescente ou decrescente, o algoritmo de ordenação por inserção, é um algoritmo estável que constrói aos poucos a lista ordenada. Ele inicia tomando o elemento da primeira posição da lista como ordenado e, em seguida, compara cada um dos elementos subsequentes com os elementos já ordenados, colocando-os em sua posição correta entre os ordenados.

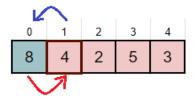
Em suma, assim como o algoritmo de ordenação por seleção, o *insertion sort* divide a lista em duas partes, uma ordenada e a outra desordenada. Assim, ele vai pegando sempre o primeiro elemento da lista desordenada e colocando em sua posição correta na lista ordenada. Esse processo ocorre até que a lista esteja completamente ordenada. A baixo temos algumas etapas do processo de ordenação:

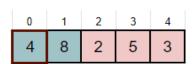
- 1. Primeiramente, é definida a primeira posição da lista como ordenada.
- 2. Em seguida, o primeiro elemento da lista não ordenada é colocado na posição correta na lista ordenada.
- 3. Por fim, o passo 2 é repetido até que todos os elementos da lista não ordenada sejam inseridos na lista ordenada.

Primeira iteração

No início, são consideradas duas sublistas:

- lista ordenada: [8];
- lista não ordenada: [4, 2, 5, 3];





atual = 4;

Insira o elemento atual na posição correta entre os elementos que estão à suas esquerta (ordenados).

Comparações e movimentações:

```
8 > 4 ? SIM, então vetor[1] = 8;
vetor[0] = atual = 4;
```

- O elemento inserido fará parte da lista ordenada:
 - lista ordenada: [4, 8]
 - lista não ordenada: [2, 5, 3];

ALGORITMO

```
1 Receba um vetor com n elementos.
2 Para i = 1 até n-1 faça:
3
      atual = vetor[i]
4
      j = i - 1
5
      Enquanto j > 0 e vetor[j] > atual faça:
6
           vetor[j + 1] = vetor[j]
7
           j = j - 1
8
        fim-enquanto
9
        A[j + 1] = atual
10 fim-para
11 Retorne o vetor ordenado.
```

EXPLICAÇÃO DO ALGORITMO

- Linha 1: o algoritmo inicia recebendo o vetor n\u00e3o ordenado com n elementos, considerando o primeiro elemento na posi\u00e7\u00e3o 0 (zero).
- Linha 2: o algoritmo realiza um loop que percorrerá a lista de elementos. A variável 'i' é utilizada como contador e começará com o valor '1', indicando que o primeiro elemento, o da posição zero, já está ordenado. O loop seguirá até que a variável 'i' atinja o valor 'n 1', indicando que todos os elementos da lista foram comparados.
- Linha 3: esta linha atribui o valor do elemento atual (vetor[i]) à variável 'atual'.
- Linha 4: esta linha inicializa a variável 'j' com o valor 'i 1', o que indica que o último elemento à esquerda já está ordenado.
- Linha 5: aqui é definido o início de um loop que percorrerá os elementos já ordenados
 à esquerda do elemento atual. O loop continuará enquanto 'j >= 0' (ou seja, enquanto
 ainda houver elementos à esquerda para comparar) e 'vetor[j] > atual' (ou seja,
 enquanto o elemento atual for menor que o elemento comparado).
- Linha 6: desloca o elemento comparado ('vetor[j]') uma posição à direita ('vetor[j + 1]').
- Linha 7: decrementa o valor de 'j' em '1', o que indica que o próximo elemento à
 esquerda será comparado na próxima iteração deste loop.
- Linha 8: finaliza o loop que compara os elementos já ordenados à esquerda do elemento atual.

- Linha 9: insere o elemento 'atual' na posição correta, deslocando os elementos maiores à direita.
- Linhas 10 e 11: depois que todos os elementos são percorridos e ordenados, o vetor estará ordenado.

CÓDIGO EM GO

```
package main
 2 import "fmt"
 4 func main() {
        exemplo := []int{12, 11, 13, 5, 6}
        fmt.Println("Antes da ordenação por inserção:", exemplo)
insercao(exemplo)
        fmt.Println("Após a ordenação por inserção:", exemplo)
 9 }
11 func insercao(exemplo []int) {
        n := len(exemplo)
        for i := 1; i < n; i++ {
13 -
            valor := exemplo[i]
            j := i - 1
17 -
            for j \ge 0 \&\& exemplo[j] > valor {
                 exemplo[j+1] = exemplo[j]
                j--
21
            exemplo[j+1] = valor
        }
23 }
```