Métodos de Seleção: Selection, Insertion, Bubble Sort

1

Ordenação

É a operação de rearranjar os dados em uma determinada ordem.

Problema da ordenação - Definição formal [1]

Entrada: Uma sequência de n números $(a_1, a_2, \ldots, a_n)$.

Saída: Uma permutação (reordenada) $(a'_1, a'_2, \ldots, a'_n)$ da sequência de entrada tal que $a'_1 \leq a'_2 \leq \ldots \leq a'_n$

2

Ordenação

- A eficiência no manuseio de dados muitas vezes pode ser aumentada se os dados estiverem dispostos de acordo com algum critério de ordem
  - Exemplo: Uma lista telefônica sem ordem.

POLICIA LOCAL ORIHUELA COSTA	966 760 000
GUARDIA CIVIL	966 796 143
AMBULANCIA SAMUR	112
BOMBEROS	965 300 080
HOSPITAL VEGA BAJA	965 367 054
CLINICA INTERNATIONAL (PRIVADA)	966 765 138
FARMACIA LA FUENTE	965 300 099
AYUNTAMIENTO ORIHUELA COSTA	966 760 000
OFICINA TURISMO ORIHUELA COSTA	966 760 000
AEROPUERTO - ALICANTE: EL ALTET	966 919 100
AEROPUERTO - MURCIA: SAN JAVIER	968 172 025
VICTIMAS VIOLENCIA DOMESTICA	016

Ordenação

- Do ponto da memória do computador, os algoritmos de ordenação podem ser classificados em:
  1. **Ordenação Interna** (quando os dados a serem ordenados estão na memória principal).

Ordenação

- Do ponto da memória do computador, os algoritmos de ordenação podem ser classificados em:
  1. **Ordenação Interna** (quando os dados a serem ordenados estão na memória principal).
  2. **Ordenação Externa** (quando os dados a serem ordenados necessitam de armazenamento em memória auxiliar como por exemplo o disco HD).

Nesta disciplina estamos interessados em métodos de ordenação interna.

5

Ordenação Interna

Na escolha de um algoritmo de ordenação interna deve ser considerado principalmente:

- O tempo gasto pela ordenação;
- O uso econômico da memória disponível;

6

Ordenação Interna

- A maioria dos métodos de ordenação é baseado em **comparação de chaves**;
  - Exemplos: insertionsort, selectionsort, etc;
- Existem métodos de ordenação que utilizam o princípio da **distribuição**;
  - Exemplos: radixsort, bucketsort, etc;

Ordenação Interna

Algoritmos de Ordenação

Algoritmos de ordenação podem ser aplicados a diversos tipos de estrutura, tais como:

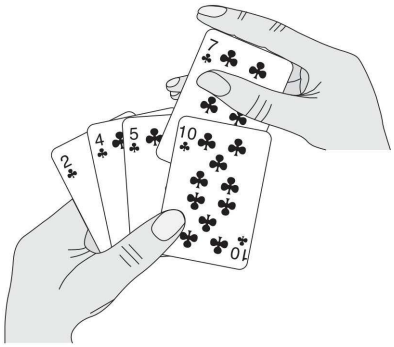
- Vetores;
- Matrizes;
- Estruturas dinâmicas.

Classificação

Um algoritmo de ordenação é **estável** se preserva ordem relativa de itens com valores idênticos.

Ordenação por inserção (InsertionSort)

A idéia da ordenação por inserção é dividir os elementos em duas subestruturas, uma com os elementos já ordenados e outra com elementos ainda por ordenar.



9

Ordenação por inserção (InsertionSort) - Exemplo

O	R	D	E	N	A

10

Ordenação por inserção (InsertionSort) - Exemplo

O	R	D	E	N	A
O	R	D	E	N	A

11

Ordenação por inserção (InsertionSort) - Exemplo

O	R	D	E	N	A
O	R	D	E	N	A
O	R	D	E	N	A

12

Ordenação por inserção (InsertionSort) - Exemplo

O	R	D	E	N	A
O	R	D	E	N	A
O	R	D	E	N	A
D	O	R	E	N	A

13

Ordenação por inserção (InsertionSort) - Exemplo

O	R	D	E	N	A
O	R	D	E	N	A
O	R	D	E	N	A
D	O	R	E	N	A
D	E	O	R	N	A

14

Ordenação por inserção (InsertionSort) - Exemplo

O	R	D	E	N	A
O	R	D	E	N	A
O	R	D	E	N	A
D	O	R	E	N	A
D	E	O	R	N	A
D	E	N	O	R	A

15

Ordenação por inserção (InsertionSort) - Exemplo

O	R	D	E	N	A
O	R	D	E	N	A
O	R	D	E	N	A
D	O	R	E	N	A
D	E	O	R	N	A
D	E	N	O	R	A
A	D	E	N	O	R

Intercultural Computer Science Education
https://www.youtube.com/watch?v=ROaIU379l3U

16

Ordenação por inserção (InsertionSort) - código em Python

```
def insertion_sort(lista):
    for i in range(1, len(lista)):
        chave = lista[i]
        j = i - 1
        while j >= 0 and lista[j] > chave:
            lista[j + 1] = lista[j]
            j -= 1
        lista[j + 1] = chave

# Exemplo de uso
lista = [5, 2, 8, 12, 1, 7]
insertion_sort(lista)
print(lista)  # Output: [1, 2, 5, 7, 8, 12]
```


Passo a passo

Passo 1: Inicialize a lista a ser ordenada.
Passo 2: Defina um loop for para percorrer a lista a partir do segundo elemento.
Passo 3: Armazene o elemento atual em uma variável chamada “chave”.
Passo 4: Defina um loop while para comparar o elemento atual com os elementos anteriores.
Passo 5: Se o elemento anterior for maior que a chave, mova-o uma posição à frente.
Passo 6: Repita o Passo 5 até que o elemento anterior seja menor ou igual à chave, ou até chegar ao início da lista.
Passo 7: Insira a chave na posição correta encontrada.
Passo 8: Repita os Passos 3 a 7 para todos os elementos da lista.
Passo 9: A lista estará ordenada.

Passo a passo

Passo 1: A lista inicial é [5, 2, 8, 12, 1, 7].

Passo 2: O loop for começa a partir do segundo elemento (índice 1).

Passo 3: A chave é o elemento atual, que começa com o valor 2.

Passo 4: O loop while compara o elemento atual (2) com o elemento anterior (5).

Passo 5: Como o elemento anterior (5) é maior que a chave (2), ele é movido uma posição à frente.

Passo 6: O loop while continua comparando a chave com os elementos anteriores, até chegar ao início da lista.

Passo 7: A chave (2) é inserida na posição correta encontrada, resultando em [2, 5, 8, 12, 1, 7].

Passo 8: O passo a passo é repetido para os elementos restantes da lista.

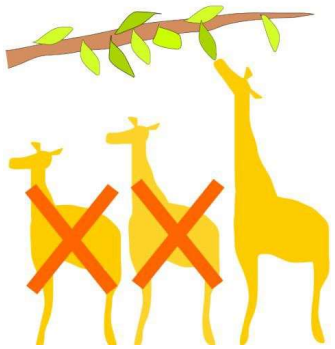
Passo 9: Ao final do processo, a lista estará ordenada: [1, 2, 5, 7, 8, 12].

Exemplo

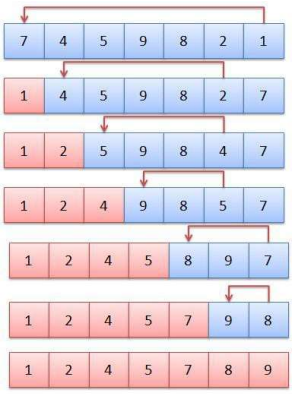
https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/AOrdenacaoPorInsercao.html

Ordenação por seleção (SelectionSort)

- A idéia da ordenação por seleção é procurar o menor elemento do vetor (ou maior) e movimentá-lo para a primeira (última) posição do vetor.
- Repetir para os *n* elementos do vetor.



Ordenação por seleção (SelectionSort) - Exemplo



Intercultural Computer Science Education
https://www.youtube.com/watch?v=Ns4TPTC8whw

SelectionSort - código em Pyhton

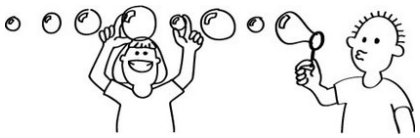
```
def selection_sort(lista):
    n = len(lista)
    for i in range(n):
        min_index = i
        for j in range(i + 1, n):
            if lista[j] < lista[min_index]:
                min_index = j
        lista[i], lista[min_index] =
lista[min_index], lista[i]
    return lista

# Exemplo de uso:
lista = [64, 25, 12, 22, 11]
lista_ordenada = selection_sort(lista)
print("Lista ordenada:", lista_ordenada)
```


SelectionSort - código em Pyhton

https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/AOrdenacaoPorSelecao.html

Ordenação por bolhas (BubbleSort)

- A idéia da ordenação por bolhas é flutuar o maior elemento para o fim.
- Repita *n* vezes a flutuação.



25

Ordenação por bolhas (BubbleSort)

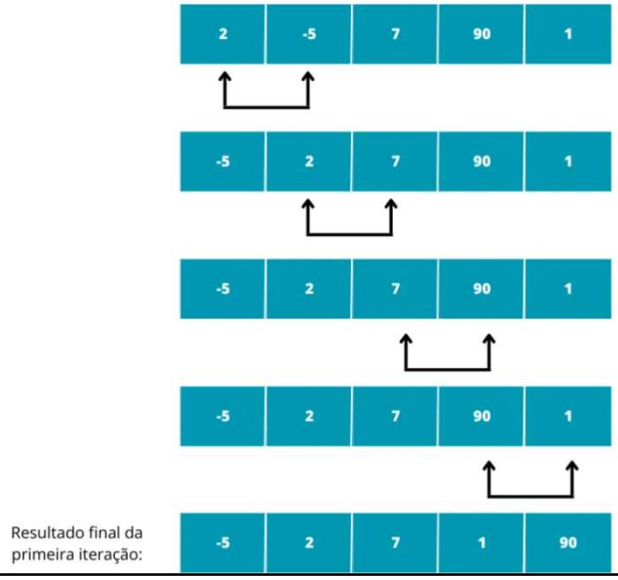
organizar os seguintes elementos em ordem crescente: 2, -5, 7, 90, 1

Passos realizados pelo algoritmo:

O algoritmo irá pegar o elemento com índice 0 e comparar com o elemento de índice 1.
Se o elemento de índice 0 for maior que o elemento de índice 1, eles serão trocados.
Agora, a comparação será realizada entre o elemento de índice 1 com o elemento de índice 2.
Esse processo irá continuar até o último elemento.
O mesmo processo irá acontecer nas iteração futuras, até que os elementos estejam ordenados em ordem crescente.

26

Ordenação por bolhas (BubbleSort)



Ordenação por bolhas (BubbleSort) - Exemplo

Usando "visualgo" para rodar exemplos

Vamos rodar um exemplo com:
https://visualgo.net/en/sorting

Intercultural Computer Science Education
https://www.youtube.com/watch?v=lyZQPjUT5B4

BubbleSort - código em Python

```
def bubble_sort(elemento):
    n = len(elemento)
    for i in range(n):
        print(f'Iteração {i+1}: {elemento}')
        for j in range(0, n-i-1):
            if elemento[j] > elemento[j+1]:
                elemento[j], elemento[j+1] =
elemento[j+1], elemento[j]
    print(f'Resultado final: {elemento}')
    return elemento

lista = [2, -5, 7, 90, 1]
bubble_sort(lista)
```


BubbleSort - código em Python

https://panda.ime.usp.br/panda/static/pythonds_pt/05-OrdenacaoBusca/OBubbleSort.html

Ordenação por inserção (InsertionSort) - Exercícios

1. Executar a função insertionSort para um vetor de tamanho 5 em ordem crescente e contar o número de vezes que a linha 12 é executada;
2. Executar a função insertionSort para um vetor de tamanho 5 em ordem decrescente e contar o número de vezes que a linha 12 é executada;
3. Qual é o pior caso e o melhor caso para este algoritmo?
4. A ordenação por inserção é estável?

Ordenação por inserção (InsertionSort)

- O número mínimo de comparações e movimentos ocorre quando os itens estão originalmente em ordem.
- O número máximo de comparações e movimentos ocorre quando os itens estão originalmente em ordem reversa.
- Bom método a ser usado quando a sequência esta quase ordenada, ou quando se deseja adicionar poucos itens a uma sequência já ordenada.
- O algoritmo de ordenação por inserção é estável.

SelectionSort - Exercícios

1. Determine o melhor e o pior caso para o SelectionSort?
2. A ordenação por seleção é estável?

SelectionSort - Vantagens e Desvantagens

1. Uma vantagem do Selection Sort é que entre os algoritmos de ordenação ele apresenta uma das menores quantidades de movimentos entre os elementos, assim pode haver algum ganho quando se necessita ordenar estruturas complexas.
2. Uma desvantagem é que o número de comparações é igual para o melhor caso, caso médio e o pior caso. Assim, mesmo que o vetor esteja ordenado o custo continua quadrático (n^2).
3. Não é estável (depende da implementação).

Exercícios

1. Determine o melhor e o pior caso para o BubbleSort?
2. Como o algoritmo BubbleSort apresentado pode ser melhorado?
3. O BubbleSort é estável?

BubbleSort - Vantagens e Desvantagens

1. Simples de entender e implementar.
2. Uma desvantagem é que na prática ele tem execução lenta mesmo quando comparado a outros algoritmos quadráticos (n^2).
3. Tem um número muito grande de movimentação de elementos, assim não deve ser usado se a estrutura a ser ordenada for complexa.