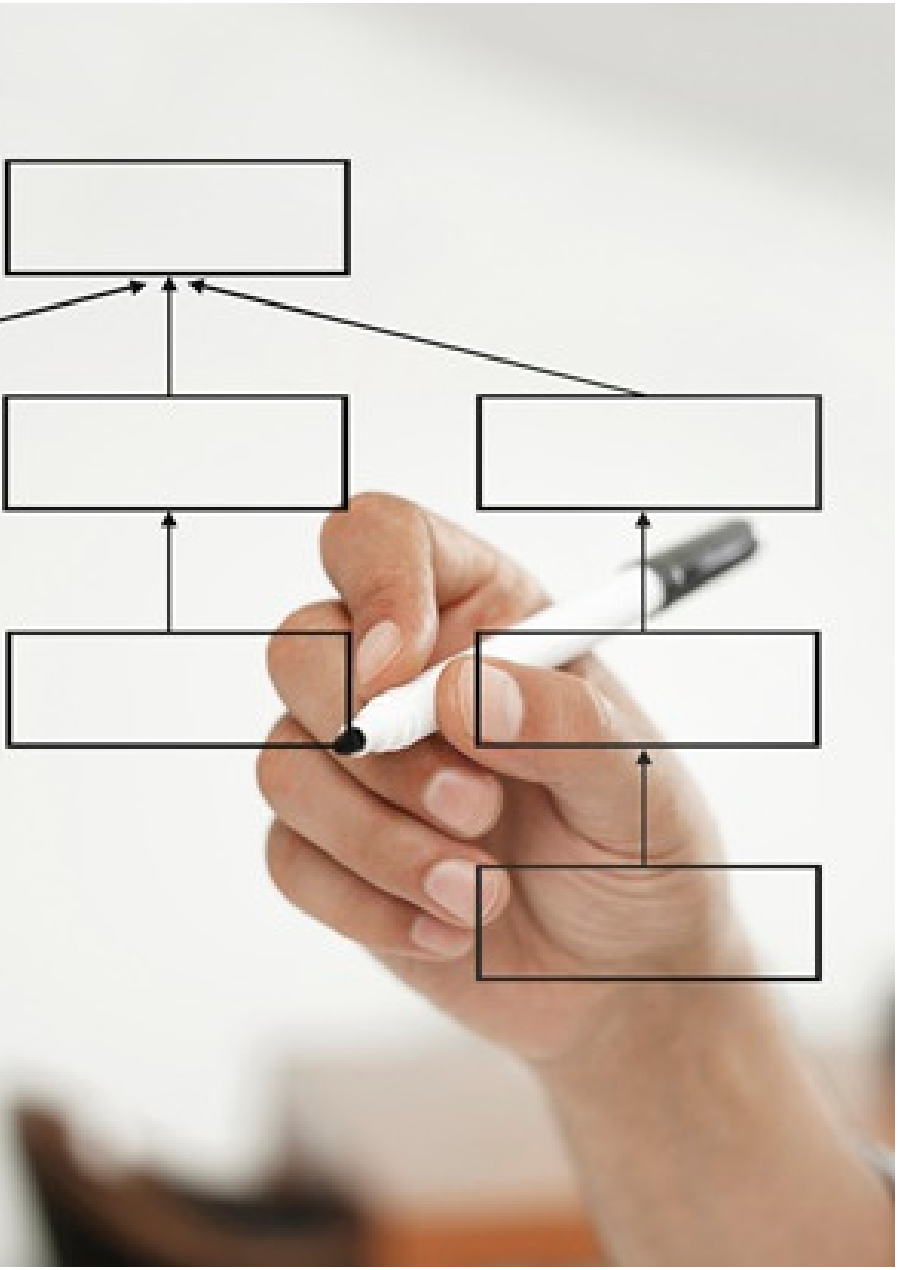
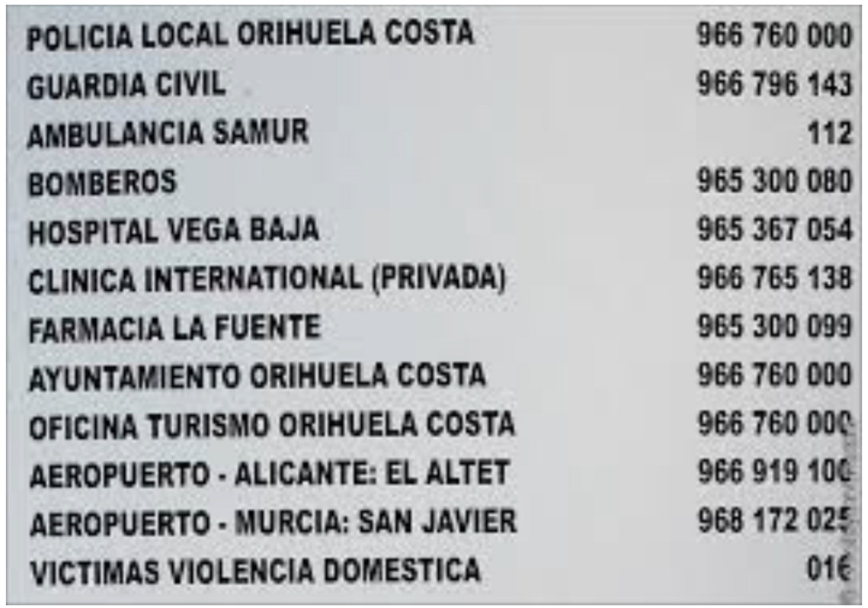


Estrutura de Dados I

CÁSSIO CAPUCHO PEÇANHA – 06

Ordenação

- Ordenar: processo de rearranjar um conjunto de objetos em uma ordem ascendente ou descendente.
- A ordenação visa facilitar a recuperação posterior de itens do conjunto ordenado.
  - Exemplo: Dificuldade de se utilizar um catálogo telefônico se os nomes das pessoas não estivessem listados em ordem alfabética.



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

- Notação utilizada nos algoritmos:
  - Os algoritmos trabalham sobre os registros de um arquivo.
  - Cada registro possui uma chave utilizada para controlar a ordenação.
  - Podem existir outros componentes em um registro.
- Qualquer tipo de chave sobre o qual exista uma regra de ordenação bem-definida pode ser utilizado.

Ordenação

- Do ponto da memória do computador, os algoritmos de ordenação podem ser classificados em:
  - **Ordenação Interna** (quando os dados a serem ordenados estão na memória principal).
  - **Ordenação Externa** (quando os dados a serem ordenados necessitam de armazenamento em memória auxiliar como por exemplo o disco HD).
- Na escolha de um algoritmo de **ordenação interna** deve ser considerado principalmente:
  - O tempo gasto pela ordenação;
  - O uso econômico da memória disponível;

Ordenação

- 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;

Bucket Sort

- O Bucket Sort (ou ordenação por baldes) é um algoritmo de ordenação baseado em distribuição.
- Ele não compara elementos diretamente como o Bubble Sort ou o Selection Sort, mas distribui os elementos em baldes (subconjuntos) e depois ordena cada balde individualmente.

Bucket Sort

- Exemplo de ordenação por distribuição: considere o problema de ordenar um baralho com 52 cartas na ordem:

$$\mathrm{A} < 2 < 3 < \cdots < 10 < \mathrm{J} < \mathrm{Q} < \mathrm{K}$$

e

$$\clubsuit < \diamondsuit < \heartsuit < \spadesuit.$$

- Algoritmo:
  - 1. Distribuir as cartas em treze montes: ases, dois, três, . . ., reis.
  - 2. Colete os montes na ordem especificada.
  - 3. Distribua novamente as cartas em quatro montes: paus, ouros, copas e espadas.
  - 4. Colete os montes na ordem especificada.

Radix Sort

- O Radix Sort também é um algoritmo de ordenação por distribuição (assim como o Bucket Sort).
- A diferença é que ele ordena dígito por dígito, geralmente começando do menos significativo (LSD – Least Significant Digit) até o mais significativo.
  - Isso significa: primeiro organizamos pelos unidades, depois pelas dezenas, e por último pelas centenas.
- E para que funcione corretamente, usamos sempre um algoritmo estável em cada passo (normalmente Counting Sort ou Bucket Sort).

Radix Sort

Vamos ordenar esta lista:

329, 457, 657, 839, 436, 720, 355

- 1º Passo: Ordenar pelas unidades (dígito menos significativo):
  - Unidades [9, 7, 7, 9, 6, 0, 5]
- Ordenando pela unidade:
  - 720, 355, 436, 457, 657, 329, 839
- 2º Passo: Ordenar pelas dezenas:
  - Dezenas [2, 5, 3, 5, 5, 2, 3]
- Ordenando pela dezena:
  - 720, 329, 436, 839, 355, 457, 657
- 3º Passo: Ordenar pelas centenas:
  - Centenas [7, 3, 4, 8, 3, 4, 6]
- Ordenando pela centena:
  - 329, 355, 436, 457, 657, 720, 839
- Resultado Final, a lista fica totalmente ordenada:
  - 329, 355, 436, 457, 657, 720, 839

Radix Sort

- O Radix Sort é como organizar uma lista de números olhando primeiro para o último dígito (unidades), depois o do meio (dezenas) e por fim o primeiro (centenas).
  - Assim, você garante que a ordem final está correta sem precisar comparar números inteiros diretamente.
- O Radix Sort é muito eficiente quando sabemos o número de dígitos máximo (números de 3 dígitos, CPF, matrícula, etc.).
- Ele não faz comparações diretas entre os números, apenas distribui pelos dígitos.
- Ele é estável porque mantém a ordem relativa dos elementos iguais em cada etapa.

Ordenação

- Métodos como o ilustrado são também conhecidos como ordenação digital, radixsort ou bucketsort.
- O método não utiliza comparação entre chaves.
- Uma das dificuldades de implementar este método está relacionada com o problema de lidar com cada monte.
- Se para cada monte nós reservarmos uma área, então a demanda por memória extra pode tornar-se proibitiva.

Ordenação

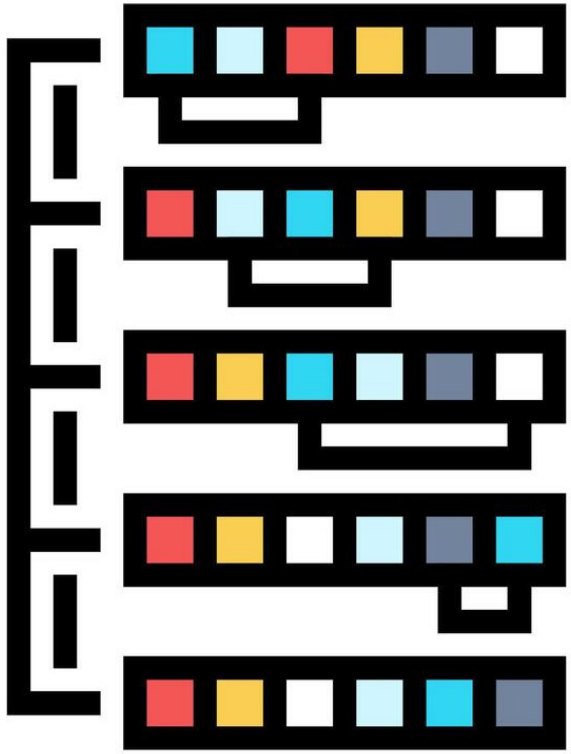
- Algoritmos de ordenação podem ser aplicados a diversos tipos de estrutura, tais como:
  - Vetores;
  - Matrizes;
  - Estruturas dinâmicas.
- Na escolha de um algoritmo de ordenação interna deve ser considerado o tempo gasto pela ordenação.
- Sendo n o número registros no arquivo, as medidas de complexidade relevantes são:
  - Número de comparações C(n) entre chaves.
  - Número de movimentações M(n) de itens do arquivo.
- O uso econômico da memória disponível é um requisito primordial na ordenação interna.
- Métodos que utilizam listas encadeadas não são muito utilizados.
- Métodos que fazem cópias dos itens a serem ordenados possuem menor importância.

Ordenação

Classificação dos métodos de ordenação interna:

- Métodos simples:
  - Adequados para pequenos arquivos.
  - Requerem O(n2) comparações.
  - Produzem programas pequenos.

- Métodos eficientes:
  - Adequados para arquivos maiores.
  - Requerem O(n log n) comparações.
  - Usam menos comparações.
  - As comparações são mais complexas nos detalhes.
  - Métodos simples são mais eficientes para pequenos arquivos.



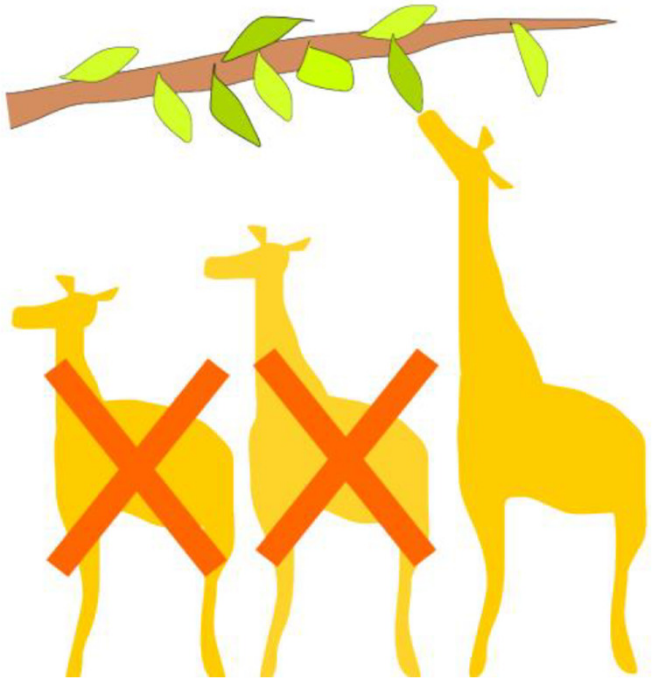
Ordenação por seleção

(SelectionSort)

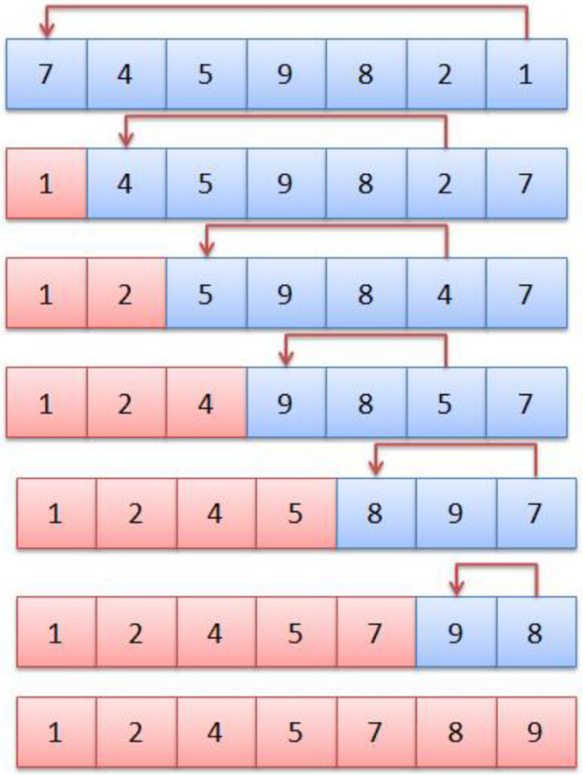
27/08/2025

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.



SelectionSort



SelectionSort - Exercício

1. Refaça o algoritmo, invertendo a logica para ordenação pelo maior elemento (mantendo a ordem crescente no vetor ordenado).
2. Determine o melhor e o pior caso para o SelectionSort?
3. A ordenação por seleção é estável?

4. **Envio de arquivo em .c.txt no blog.

Estabilidade de algoritmo

- Um algoritmo de ordenação é estável quando mantém a ordem relativa dos elementos iguais.
- Ou seja, se dois elementos possuem a mesma chave de comparação, depois da ordenação eles continuam aparecendo na mesma ordem em que estavam originalmente.
- Um algoritmo instável pode inverter a ordem relativa desses elementos iguais.

Estabilidade de algoritmo

- Imagine que temos objetos representando pessoas, ordenados por idade:
  - (Ana, 20)
  - (João, 18)
  - (Maria, 20)
  - Se ordenarmos por idade (chave = 20, 18, 20):
- Ordenação estável: mantém Ana antes de Maria, porque elas tinham a mesma idade (20) e Ana já estava antes:
  - (João, 18) (Ana, 20) (Maria, 20)
- Ordenação instável: pode trocar a ordem relativa:
  - (João, 18) (Maria, 20) (Ana, 20)

Estabilidade de algoritmo

- Bubble Sort → Estável
  - Quando ele encontra dois elementos iguais, não troca a posição deles sem necessidade. Assim, a ordem original é preservada.
- Insertion Sort → Estável
  - Ao inserir o elemento na parte já ordenada, ele não ultrapassa outros elementos iguais que estavam antes.
- Selection Sort → Instável
  - Ele busca o menor elemento e troca diretamente com a posição atual. Essa troca pode mover um elemento igual que estava depois, para antes, mudando a ordem relativa.