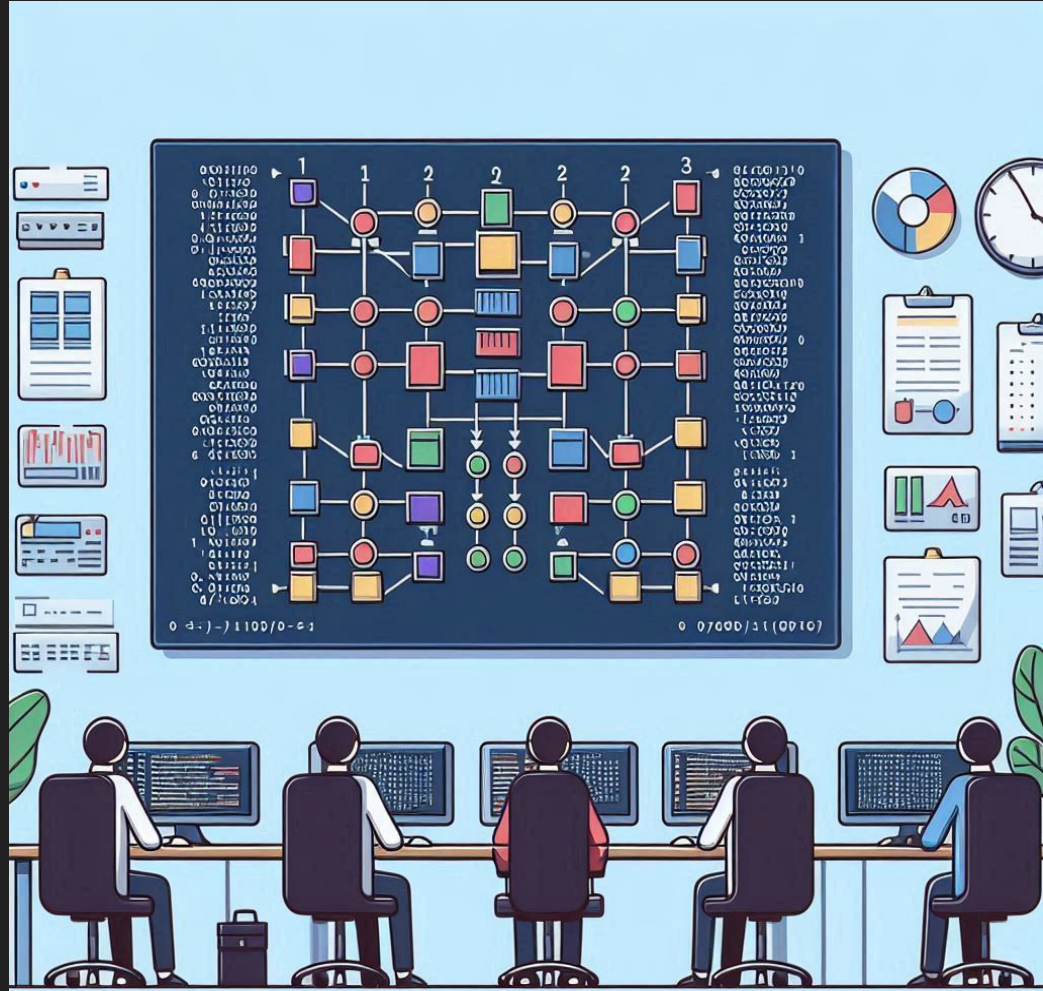
Algoritmos de Ordenação

Conceitos e Implementação

Prof.: Juliano Ramos Matos



O Que São Algoritmos de Ordenação

- Métodos ou procedimentos usados para reorganizar os elementos de uma lista ou array em uma ordem específica, geralmente crescente ou decrescente.
- Ordenação é uma operação básica em muitas tarefas, como busca, análise de dados e otimização.
- Quando os dados estão ordenados, encontrar um elemento específico se torna muito mais rápido. É como procurar uma palavra em um dicionário.
- Muitos algoritmos de análise de dados dependem de dados ordenados para funcionar corretamente.

Principais Algoritmos de Ordenação

- Métodos simples:
  - Bubble Sort (ordenação por flutuação)
  - Selection Sort (ordenação por seleção)
  - Insertion Sort (ordenação por inserção)
- Métodos sofisticados:
  - Merge Sort (ordenação por intercalação)
  - Quick Sort (ordenação rápida)

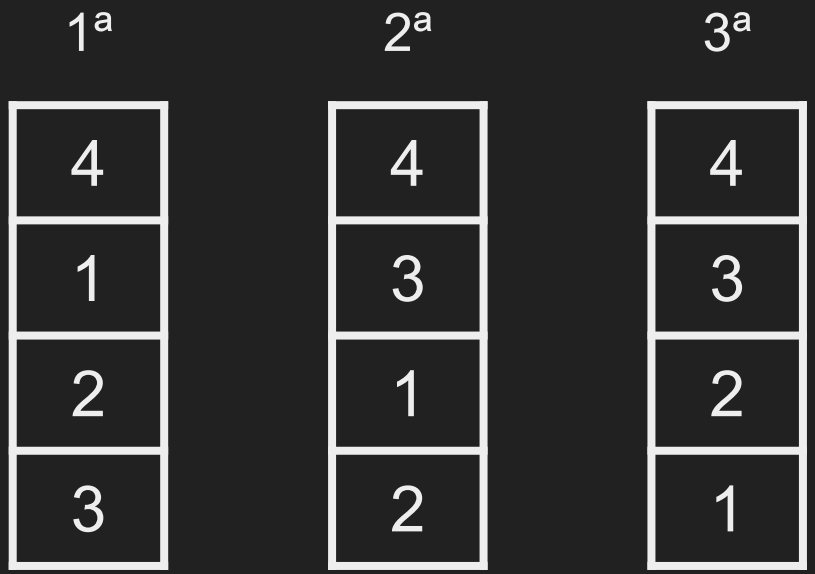
Bubble Sort

- Repetidamente percorre a lista, comparando elementos adjacentes e trocando-os se estiverem na ordem errada.
- Como Funciona:
  - Inicia no início do array.
  - Compara o primeiro elemento com o segundo; se o primeiro é maior, troca-os.
  - Move para o próximo par de elementos adjacentes e repete o processo até o final da lista.
  - O maior elemento "borbulha" para o final da lista.
  - Repete o processo para os elementos restantes até que a lista esteja ordenada.
- Simples de implementar, mas ineficiente para grandes listas.

Bubble Sort - Exemplo

Vetor de entrada

Saída a cada iteração



Bubble Sort - Implementação

```
int n = arr.length;
for (int i = 0; i < n - 1; i++) {
      for (int j = 0; j < n - i - 1; j++) {
          if (arr[j] > arr[j + 1]) {
              int temp = arr[j];
              arr[j] = arr[j + 1];
              arr[j + 1] = temp;
          }
      }
  }
```

Selection Sort

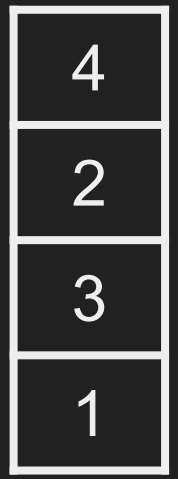
- Repetidamente encontra o menor (ou maior) elemento da lista desordenada e o coloca na posição correta na lista ordenada.
- Como Funciona:
  - Cada número do vetor a partir do primeiro será eleito e comparado.
  - Itera sobre a lista desordenada para encontrar o menor elemento.
  - Troca o menor elemento encontrado com o elemento eleito.
  - Passa para o próximo elemento e repete o processo até que toda a lista esteja ordenada.
- Melhor que o Bubble Sort em termos de número de trocas, mas ainda não é eficiente para grandes listas.

Selection Sort - Exemplo

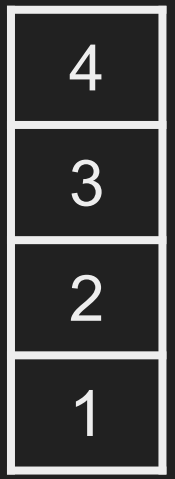
Vetor de entrada

Saída a cada iteração

1ª



2ª



Selection Sort - Implementação

```
int n = arr.length;
for (int i = 0; i < n-1; i++) {
    int min_idx = i;
    for (int j = i+1; j < n; j++){
        if (arr[j] < arr[min_idx])
            min_idx = j;
    int temp = arr[min_idx];
    arr[min_idx] = arr[i];
    arr[i] = temp;
    }
}
```

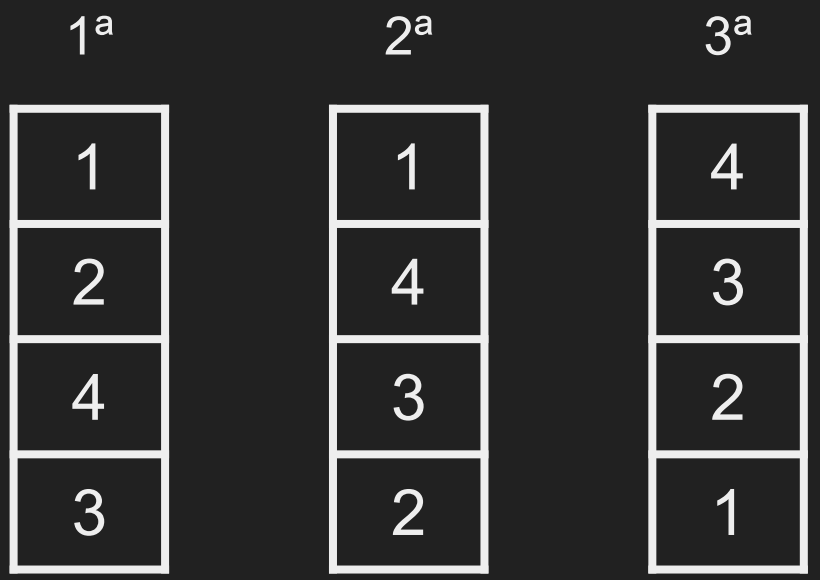
Insertion Sort

- Constrói a lista ordenada um elemento de cada vez, tomando um elemento da lista desordenada e inserindo-o na posição correta na lista ordenada.
- Como Funciona:
  - Começa no segundo elemento do array.
  - Compara o elemento atual com os elementos na lista ordenada e encontra a posição correta.
  - Move todos os elementos maiores que o elemento atual uma posição para a direita.
  - Insere o elemento atual na posição correta.
  - Repete o processo para todos os elementos até que a lista esteja completamente ordenada.
- Eficiente para pequenas listas ou listas que já estão quase ordenadas.

Insertion Sort

Vetor de entrada

Saída a cada iteração



Insertion Sort - Implementação

```
int n = arr.length;
for (int i = 1; i < n; i++) {
    int key = arr[i]; // Elemento a ser inserido na parte ordenada
    int j = i - 1;
    while (j >= 0 && arr[j] > key) {
        arr[j + 1] = arr[j];
        j = j - 1;
    }
    arr[j + 1] = key; // Insere o elemento na posição correta
}
```

Comparação

Iteração	Bubble	Selection	Insertion
Inicial	[5, 2, 9, 1, 6]	[5, 2, 9, 1, 6]	[5, 2, 9, 1, 6]
1	[2, 5, 1, 6, 9]	[1, 2, 9, 5, 6]	[2, 5, 9, 1, 6]
2	[2, 1, 5, 6, 9]	[1, 2, 9, 5, 6]	[2, 5, 9, 1, 6]
3	[1, 2, 5, 6, 9]	[1, 2, 5, 9, 6]	[1, 2, 5, 9, 6]
4	[1, 2, 5, 6, 9]	[1, 2, 5, 6, 9]	[1, 2, 5, 6, 9]

Vamos a Prática!

Referências

ASCENCIO, Ana Fernanda Gomes; ARAÚJO, Graziela Santos de. Estruturas de dados: algoritmos, análise da complexidade e implementações em Java e C/C++. 1. ed. São Paulo: Pearson, 2010.

https://www.freecodecamp.org/portuguese/news/algoritmos-de-ordenacao-explicados-com-exemplos-em-python-java-e-c/

https://www.devmedia.com.br/algoritmos-de-ordenacao-analise-e-comparacao/28261