### **Métodos de Classificação (Ordenação) de Dados**

#### 1) Método da Seleção Direta (Selection Sort) Demonstração: https://youtu.be/pDY2rGdsYAE?si=u5MfneLnyOO3kAoW

* **Conceito:** A cada passo, busca o menor elemento dentre os ainda não ordenados e o posiciona no início da sub-lista desordenada.
* **Como funciona:**
  1. Divide a lista em duas partes: uma ordenada (à esquerda, inicialmente vazia) e outra desordenada (à direita).
  2. Encontra o menor valor na parte desordenada.
  3. Troca esse valor com o primeiro elemento da parte desordenada.
  4. Aumenta a parte ordenada em um elemento e repete o processo até que a parte desordenada não exista mais.
* **Característica Principal:** É um algoritmo simples de entender, porém ineficiente para grandes volumes de dados, pois o número de comparações é sempre o mesmo, independentemente do quão ordenada a lista já esteja. Sua complexidade de tempo é quadrática: O(n2).

#### 2) Método da Inserção Direta (Insertion Sort) Demonstração: https://youtu.be/7GUwzd\_h3pI?si=F1AdameVzGCQWNzu

* **Conceito:** Percorre a lista e, a cada novo elemento, o insere em sua posição correta dentro da parte já ordenada à sua esquerda.
* **Como funciona:**
  1. Começa a partir do segundo elemento da lista.
  2. Pega o elemento atual e o compara com os elementos à sua esquerda.
  3. Se o elemento atual for menor, "empurra" os elementos maiores para a direita para abrir espaço e o insere na posição correta.
  4. Repete o processo até o final da lista.
* **Característica Principal:** Muito eficiente para listas pequenas ou que já estão parcialmente ordenadas. Sua analogia comum é a ordenação de cartas de baralho na mão. A complexidade de tempo no pior caso também é O(n2).

#### 3) Método dos Incrementos Decrescentes (Shellsort) Demonstração: https://youtu.be/1tGpLS5JH2Y?si=hR34LKiectW-F5Vk

* **Conceito:** É uma otimização do método da Inserção Direta. Ele compara e troca elementos que estão a uma determinada distância (incremento ou *gap*) e vai reduzindo essa distância a cada passo.
* **Como funciona:**
  1. Define uma sequência de incrementos (gaps), começando com um valor grande (ex: metade do tamanho da lista).
  2. Aplica uma espécie de "ordenação por inserção" para os elementos que estão a essa distância um do outro. Isso move rapidamente os elementos menores para perto do início da lista.
  3. Reduz o incremento (ex: divide por 2) e repete o processo.
  4. O último passo é com um incremento de 1, que funciona como uma Inserção Direta, mas em uma lista agora "quase ordenada", tornando-a muito mais rápida.
* **Característica Principal:** Significativamente mais rápido que a Inserção e Seleção Diretas. Sua complexidade de tempo é melhor que O(n2), embora sua análise exata seja complexa e dependa da sequência de incrementos usada.

#### 4) Método da Bolha (Bubblesort) Demonstração: https://youtu.be/i2IKFmI\_gyk?si=E7\_I1xnd4uGuyaiy

* **Conceito:** Compara repetidamente pares de elementos adjacentes e os troca se estiverem na ordem errada, fazendo com que os maiores elementos "flutuem" (ou "borbulhem") para o final da lista a cada passagem.
* **Como funciona:**
  1. Percorre a lista comparando o primeiro com o segundo elemento, o segundo com o terceiro, e assim por diante.
  2. Se um par estiver fora de ordem, seus elementos são trocados.
  3. Ao final da primeira passagem, o maior elemento da lista estará na última posição.
  4. Repete o processo para a sub-lista restante, até que nenhuma troca seja necessária em uma passagem completa.
* **Característica Principal:** É um dos algoritmos mais simples de implementar, mas também um dos mais lentos, com complexidade de tempo O(n2). É mais utilizado para fins didáticos.

#### 5) Método da Troca e Partição (Quicksort) Demonstração: https://youtu.be/WP7KDljG6IM?si=SwXlUYougEjEypjW

* **Conceito:** Utiliza a estratégia de "dividir para conquistar". Ele seleciona um elemento como **pivô** e reorganiza a lista de modo que os elementos menores que o pivô fiquem antes dele e os maiores, depois.
* **Como funciona:**
  1. Escolhe um elemento da lista para ser o pivô (a escolha do pivô impacta a performance).
  2. **Particiona** a lista: reorganiza os elementos para que todos os menores que o pivô estejam à sua esquerda e os maiores, à sua direita. O pivô fica em sua posição final correta.
  3. Aplica o mesmo processo recursivamente para as duas sub-listas (a da esquerda e a da direita do pivô).
* **Característica Principal:** É um dos algoritmos mais rápidos na prática para grandes volumes de dados. Sua complexidade de tempo média é O(nlogn), mas no pior caso (com pivôs mal escolhidos) pode decair para O(n2).

#### 6) Método da Intercalação (Mergesort) Demonstração: https://youtu.be/BnsYGiYYdnQ?si=z0ULJJOO7fzEudWX

* **Conceito:** Também baseado em "dividir para conquistar", ele divide a lista em metades sucessivamente até que restem apenas listas de um elemento. Em seguida, combina (intercala ou *merge*) essas listas de forma ordenada.
* **Como funciona:**
  1. **Divide** a lista ao meio repetidamente até que cada sub-lista tenha apenas um elemento (uma lista de um elemento é, por definição, ordenada).
  2. **Conquista (Intercala):** Começa a juntar duas sub-listas adjacentes, comparando seus elementos e criando uma nova lista ordenada a partir delas.
  3. Repete o processo de intercalação subindo na recursão até que todas as partes sejam unidas em uma única lista final ordenada.
* **Característica Principal:** Sua performance é estável e garantida. A complexidade de tempo é sempre O(nlogn), tanto no caso médio quanto no pior. Sua principal desvantagem é a necessidade de usar memória auxiliar para realizar a intercalação.