Métodos de Ordenação

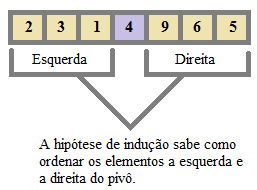
Método Quick sort - O algoritmo Quicksort é um método de ordenação muito rápido e eficiente, tendo como objetivo reduzir o problema original em subproblemas que possam ser resolvidos mais fácil e rápido. É um algoritmo de ordenação por comparação não-estável.

O Quicksort adota a estratégia de divisão e conquista. A estratégia consiste em rearranjar as chaves de modo que as chaves "menores" precedam as chaves "maiores". Em seguida o Quicksort ordena as duas sublistas de chaves menores e maiores recursivamente até que a lista completa se encontre ordenada. Os passos são:

1. É escolhido um elemento da lista, denominado pivô;
2. Rearranjando a lista de forma que todos os elementos anteriores ao pivô sejam menores que ele, e todos os elementos posteriores ao pivô sejam maiores que ele. Ao fim do processo o pivô estará em sua posição final e haverá duas partes não ordenadas. Essa operação é denominada partição;
3. Utilizando Recursividade o algoritmo fica ordenado com a parte dos elementos menores e a parte dos elementos maiores;

A base da recursão são as listas de tamanho zero ou um, que estão sempre ordenadas. O processo é finito, pois a cada iteração pelo menos um elemento é posto em sua posição final e não será mais manipulado na iteração seguinte.

Exemplo:

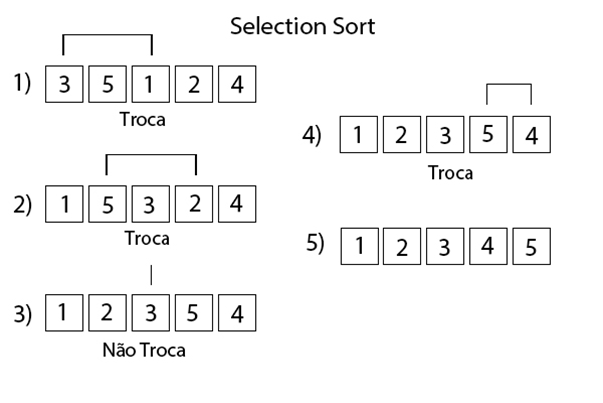


Método Selection sort - A ordenação por seleção consiste, em cada etapa, em selecionar o maior (ou o menor) elemento e colocá-lo em sua posição correta dentro da futura lista ordenada.

Durante a aplicação do método de seleção a lista com n registros fica decomposta em duas sub listas, uma contendo os itens já ordenados e a outra com os restantes ainda não ordenados. No início a sub lista ordenada é vazia e a outra contém todos os demais.

No final do processo a sub lista ordenada apresentará (n-1) itens e a outra apenas 1.

Para uma melhor compreensão trabalharemos com um exemplo, visando demonstrar o resultado das etapas da ordenação de um vetor de inteiros.

Exemplo: 

Método Insertion Sort - O método de ordenação por Inserção Direta é o mais rápido entre os outros métodos considerados básicos – Bubblesort e Seleção Direta. A principal característica deste método consiste em ordenarmos o arranjo utilizando um sub-arranjo ordenado localizado em seu inicio, e a cada novo passo, acrescentamos a este sub-arranjo mais um elemento, até que atingimos o último elemento do arranjo fazendo assim com que ele se torne ordenado.

Exemplo:  
Sequência:  
**5 - 3 - 1 - 4 - 2**  
 Inicialmente considera-se o primeiro elemento do arranjo como se ele estivesse ordenado; ele será considerado o sub-arranjo ordenado inicial.  
Agora o elemento imediatamente superior ao o sub-arranjo ordenado, no o exemplo o número 3, deve se copiado para uma variável auxiliar qualquer. Após copiá-lo, devemos percorrer o sub-arranjo a partir do último elemento para o primeiro. Assim poderemos encontrar a posição correta da nossa variável auxiliar dentro do sub-arranjo.  
No caso verificamos que a variável auxiliar é menor que o último elemento do o sub-arranjo ordenado ( o o subarranjo só possui por enquanto um elemento, o número 5 ). O número 5 deve então ser copiado uma posição para a direita para que a variável auxiliar com o número 3, seja colocada em sua posição correta.   
**3 - 5 - 1 - 4 - 2**  
 Verifique que o sub-arranjo ordenado possui agora dois elementos. Vamos repetir o processo anterior para que se continue a ordenação. Copiamos então mais uma vez o elemento imediatamente superior ao o sub-arranjo ordenado para uma variável auxiliar. Logo em seguida vamos comparando nossa variável auxiliar com os elementos do sub-arranjo, sempre a partir do último elemento para o primeiro   
Neste caso verificamos que a nossa variável auxiliar é menor que o último elemento do sub-arranjo. Assim, copiamos este elemento para a direita e continuamos com nossas comparações ( 5 permanece como cópia no lugar do 1).  
Aqui, mais uma vez a nossa variável auxiliar é menor que o elemento do sub-arranjo que estamos comparando. Por isso ele deve ser copiado para a direita, abrindo espaço para que a variável auxiliar seja colocada em sua posição correta.  
  
**1 - 3 - 5 - 4 - 2**  
  
 Aplicando o algoritmo até que se chegue ao fim da sequência, poderá resultar a seguinte sequência:  
  
**1 - 3 - 4 - 5 - 2**  
**1 - 2 - 3 - 4 – 5**

Método Merge Sort - O merge sort, ou ordenação por mistura, é um exemplo de algoritmo de ordenação do tipo dividir para conquistar.

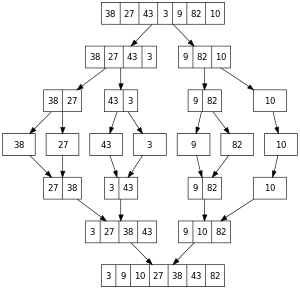
Sua ideia básica consiste em dividir (o problema em vários subproblemas e resolver esses subproblemas através da recursividade) e Conquistar (após todos os subproblemas terem sido resolvidos ocorre a conquista que é a união das resoluções dos subproblemas). Como o algoritmo Merge Sort usa a recursividade, há um alto consumo de memória e tempo de execução, tornando esta técnica não muito eficiente em alguns problemas .

Os três passos úteis dos algoritmos dividir para conquistar, que se aplicam ao merge sort são:

\*Dividir: Dividir os dados em subsequências pequenas;

\*Conquistar: Classificar as metades recursivamente aplicando o merge sort;

\*Combinar: Juntar as metades em um único conjunto já classificado.



Método Shell Sort - Shell Sort é um algoritmo de ordenação sofisticado ou eficiente. Um algoritmo simples de entender, relativamente rápido de processar e fácil de implementar.

Seu nome vem de Donald Shell, que o criou em 1959 com a ideiade dividir um grande vetor de dados em vetores menores, ordenando-os e fazendo isso novamente para ter um único vetor praticamente ordenado e então trabalhar em cima dele, que seria mais prático e rápido. O algoritmo de Shell Sort em si mesmo não ordena nada, mas aumenta a eficiência de outros algoritmos de ordenação (como o da inserção e seleção, que serão vistos no próximo post de EDA), o que definitivamente faz sentido quando se entende como ele funciona.

Princípios básicos:

\*O Shell Sort se baseia em uma variável chamada de incremento de sequência, ou incremento de shell, que é dado por he ao decorrer da execução do algoritmo, é decrementada até 1.

\*Utilizando o incremento de shell, o algoritmo compara elementos distantes em um vetor, em vez de comparar os adjacentes.

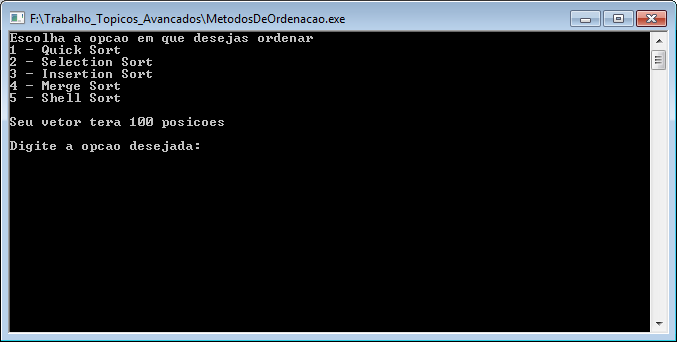
\*No algoritmo, a ordenação é realizada em vários passos, usando uma sequência de valores do incremento de shell <h1, h2, h3…hN> onde começando por hNselecionamos apenas os valores que estão hNelementos distantes um do outro, então ordenamos esses elementos com algum algoritmo de ordenação simples como bolha, seleção ou inserção. Deste modo, apenasos elementos selecionados serão ordenados, os outros são todos ignorados. Exemplo:



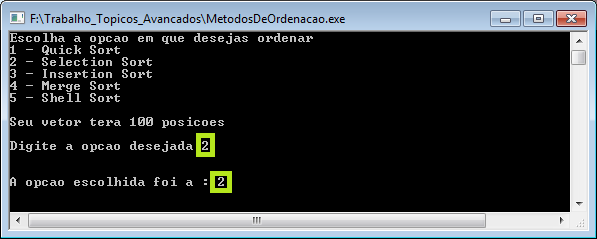
TUTORIAL

Já tivemos toda a explicação de cada Método de Ordenação existente em nosso programa. Agora temos aqui um Tutorial para demonstrar ao usuário como utilizar esse Programa:

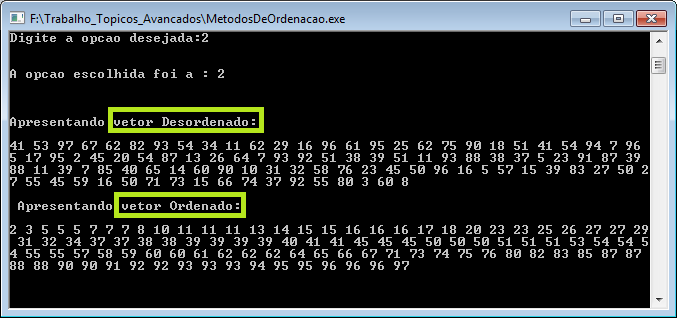
1. Aqui você deve escolher a opção em que desejas ordenar seu vetor de 100 posições, basta você digitar o número da opção e dar um “Enter” no teclado.



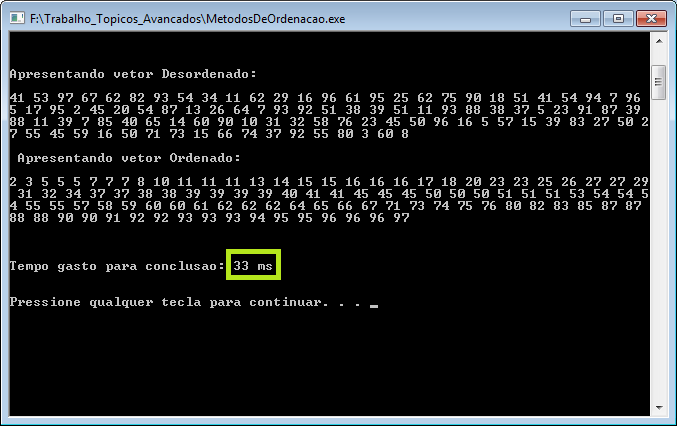
1. Veja aqui que neste exemplo eu escolhi a opção 2- Selection Sort, imediatamente o programa informa que foi essa a opção escolhida.



1. Veja que também instantaneamente é mostrado na sua tela o vetor desordenado e sucessivamente o vetor ordenado.



1. Veja que logo abaixo já é mostrado o tempo gasto em Milissegundos (ms) para a conclusão da ordenação escolhida.



Após tudo isso, basta dar um “Enter” para o Programa ser encerrado.