Algoritmos de Ordenação

Algoritmos de Ordenação são algoritmos voltados para modificar uma coleção de dados, como um vetor, fazendo com que seus elementos sigam uma determinada ordenação, como uma ordem crescente em um vetor de números inteiros.

Seu principal objetivo é facilitar a localização de dados.

Existem diversos algoritmos desenvolvidos com a mesma finalidade, no entanto, utilizam técnicas diferentes, que podem ter melhor desempenho, dependendo do tamanho do vetor ou da disposição dos dados.

Como exemplo, podemos citar o Bubble Sort, Quick Sort ou Merge Sort.

O Bubble Sort é baseado em um algoritmo bastante simples, mas tem como comportamento percorrer o vetor diversas vezes, o que pode ser de bom desempenho com vetores pequenos e menos desordenados, mas perde desempenho em vetores médios ou grandes.

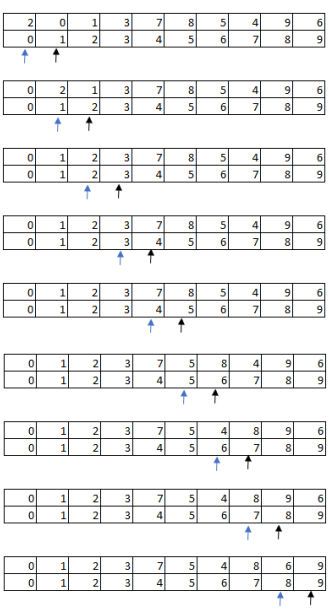
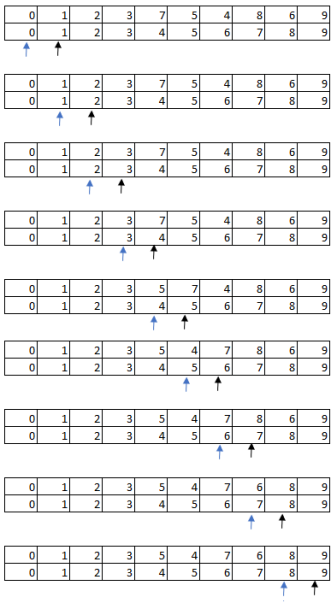
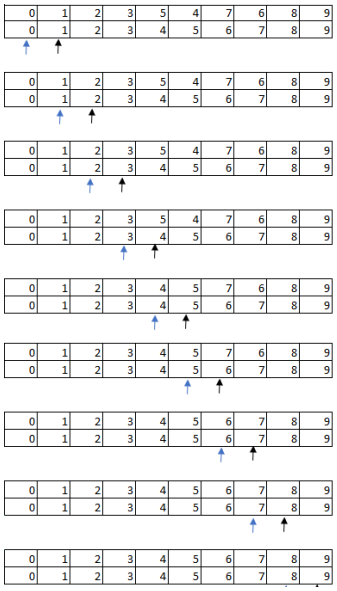
O Merge Sort tem bom desempenho em qualquer situação, seja referente ao tamanho do vetor ou à ordenação inicial dos dados. Baseado na técnica de divisão para conquista, utiliza a definição de uma posição central do vetor para dividir o vetor em sub vetor da esquerda e da direita e intercala os dados. Utiliza recursividade.

O Quick Sort é considerado um dos algoritmos com melhor desempenho. Pode ter menor desempenho que o Bubble Sort para vetores pequenos, mas tem um ótimo desempenho para vetores médios ou grandes. É baseado na técnica de divisão para conquista (Dividir para conquistar), sendo que a utilizada é denominada particionamento, utilizando um pivot, tal que, os números à esquerda são menores ou iguais ao pivot e os números à direita são maiores. Para melhor desenvolvimento, se usa recursividade.

Descrição do BubbleSort - https://www.youtube.com/watch?v=lyZQPjUT5B4

- Pegar um vetor
- O número de repetições deverá ser igual ao tamanho do vetor
- A dinâmica se dá como segue:
    - Da primeira posição do vetor, até a penúltima posição;
    - Verificar se a posição atual é maior que a próxima;
    - Se for, elas devem trocar de posição;
    - O controle de posição deve ser incrementado;
- Ao final, o vetor estará ordenado crescentemente

Exemplo:

Rodada 1	Rodada 2	Rodada 3
		 Ainda se repetirá 7 vezes

Descrição do MergeSort - https://www.youtube.com/watch?v=XaqR3G_NVoo

- Pegar um vetor (Ou parte dele)

- Identificar a posição central do vetor e dividi-lo em 2 (posição inicial – meio ; meio + 1 – posição final)

- Recursivamente, ir dividindo em sub vetores de esquerda e direita até que se encontre um vetor de 1 posição, retornando o próprio vetor;

- Nesse ponto, se retorna ao vetor anterior que deve intercalar os valores a fim de ordená-los

- A função de intercalar precisa conhecer a posição inicial, a posição final e o meio do vetor (ou sub vetor)

- A função de intercalar deve gerar um vetor auxiliar, com o tamanho do vetor inicial, mas com dados apenas nas posições do sub vetor passado como parâmetro

- Define-se como ponteiros:

    - esquerda iniciando na posição inicial

    - direita, iniciando na posição do meio + 1

- Intercalar significa fazer os seguintes testes aninhados, percorrendo o novo vetor de posição inicial para a posição final, controlando as posições por um contador

- Para cada posição do novo vetor:

    - Se o ponteiro da esquerda é maior que a posição do meio, o vetor inicial na posição do contador, recebe o valor do novo vetor na posição do ponteiro à direita. Incrementa-se o ponteiro da direita;

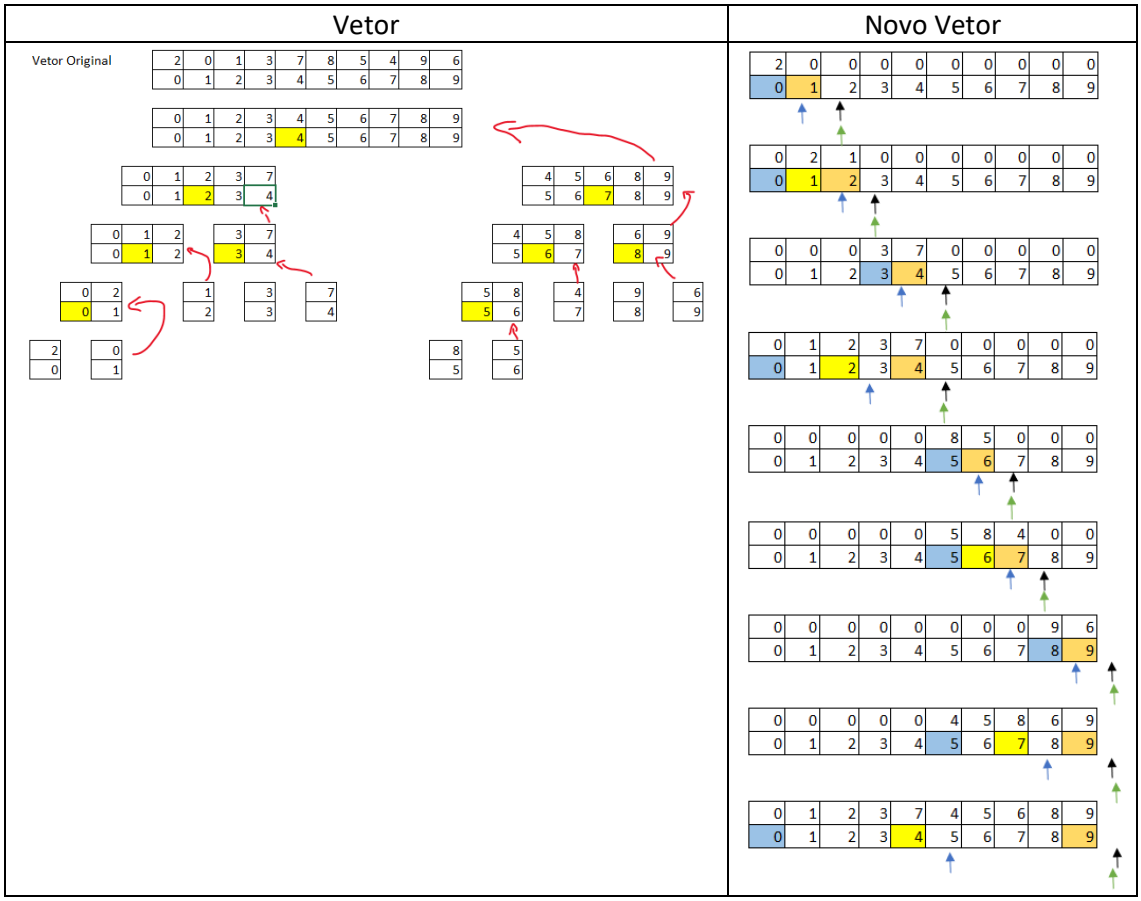
    - Senão, se o ponteiro da direta é maior que a posição do fim, o vetor inicial na posição do contador, recebe o valor do novo vetor na posição do ponteiro à esquerda. Incrementa-se o ponteiro da esquerda;

    - Senão, se o valor do novo vetor na posição do ponteiro à esquerda é menor que o valor do novo vetor na posição do ponteiro à direita, o vetor inicial na posição do contador, recebe o valor do novo vetor na posição do ponteiro à esquerda. Incrementa-se o ponteiro da esquerda;

    - Senão, o vetor inicial na posição do contador, recebe o valor do novo vetor na posição do ponteiro à direita. Incrementa-se o ponteiro da direita;

- O comportamento recursivo vai fazer com que os sub vetores retornem ordenados e desempilhando até a ordenação do vetor inicial

Exemplo:



Descrição do QuickSort - https://www.youtube.com/watch?v=ywWBy6J5gz8

- Pegar um vetor (Ou parte dele)

- Verifica se é um vetor de mais de 1 posição

    - Caso seja de 1 posição, já está ordenado

- Defina, arbitrariamente, a primeira posição como um pivô, portanto, a validação será feita da 2ª posição até a última

- Marcar a segunda posição com um ponteiro da esquerda e a última com um ponteiro da direita

- Enquanto o ponteiro da esquerda se mantiver à esquerda do ponteiro da direita, validar:

    - Enquanto o valor do ponteiro da esquerda for menor ou igual ao valor do pivô e o ponteiro da esquerda continuar à esquerda do ponteiro da direita, o ponteiro da esquerda incrementa 1. Se alguma condição falhar, o incremento cessa;

    - Enquanto o valor do ponteiro da direita for maior que o pivô e o ponteiro da esquerda continuar à esquerda ou igualar ao ponteiro da direita, o ponteiro da direita decrementa 1. Se alguma condição falhar, o decremento cessa;

    - Se incremento do ponteiro da esquerda e decremento do ponteiro da direita cessarem, mas o ponteiro da esquerda continuar à esquerda do ponteiro da direita, os valores cujos índices forem, ponteiro da esquerda e ponteiro da direita, mudam de lugar. Incrementa-se o ponteiro da esquerda e decrementa-se o ponteiro da direita;

- Quando o ponteiro da direita, passar à esquerda do ponteiro da esquerda, o valor do pivô troca de lugar com o valor cujo índice é o ponteiro da direita

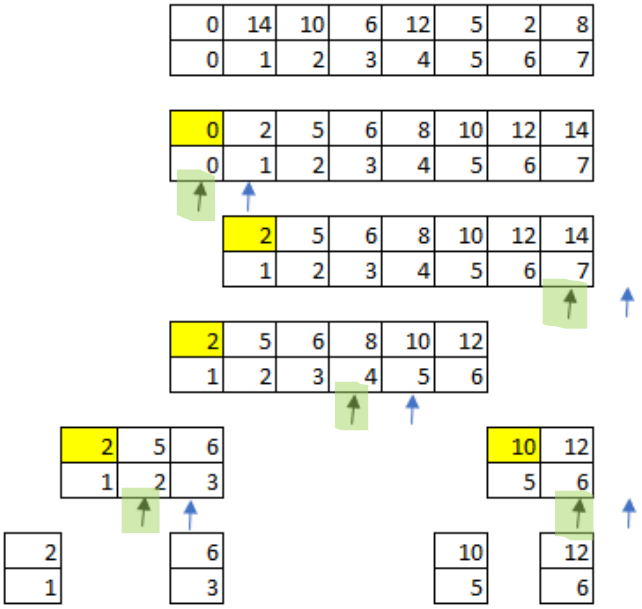
- A partir desse momento, com o pivô em algum lugar entre o início e o fim do vetor inicial, este valor já está na sua posição definitiva restando um (sub)vetor à esquerda e um (sub)vetor à direita.

-Recursivamente, a função usada com vetor inicial, deve ser chamada novamente (sem retorno), para ordenar o sub vetor da esquerda e, sequencialmente, o da direita.

- Essa recursividade deve seguir até que o subvetor fique com 1 única posição, condição de parada da recursividade.

-Neste ponto, o vetor estará ordenado

Exemplo1:



Exemplo 2: