UFC - UNIVERSIDADE FEDERAL DO CEARÁ

Centro de Ciências - Física e Matemática

Departamento de Computação

FP - Fundamentos de Programação - 2024.1

Modelos de Ordenação

Professor: José Caminha Alencar Araripe Júnior

Fortaleza, 25 de agosto de 2024

1. **Modelos de Ordenação:** Bolha, Seleção, Shell e Quicksort.

Ordenação é o processo de arranjar um conjunto de informações semelhantes numa ordem crescente ou decrescente.

Para se realizar a ordenação podem ser utilizados três métodos gerais para que um conjunto de informações de um vetor seja colocado em uma ordem crescente o decrescente, por exemplo:

- Por troca;
- Por seleção;
- Por inserção.

Dentre esses métodos, existem diversos modelos de algoritmos desenvolvidos para realizar a ordenação. A vantagem de um para outro pode ser definido pela **velocidade** que a ordenação será realizada. Isso pode ser medido através do número de comparações e o número de trocas que ocorrem para se chegar ao final da ordenação das informações.

A seguir serão detalhados alguns modelos utilizados para essa finalidade.

2. A ordenação **Bolha** - Método de trocas:

A ordenação mais simples e considerada uma das piores. É uma ordenação por trocas, envolve repetidas comparações e quando necessário a troca é realizada entre dois itens adjacentes.

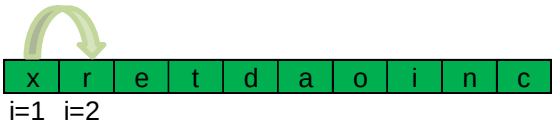
É o modelo de algoritmo de ordenação mais simples que tem como princípio de funcionamento comparações e trocas conforme os procedimentos a seguir:

Vetor inicial

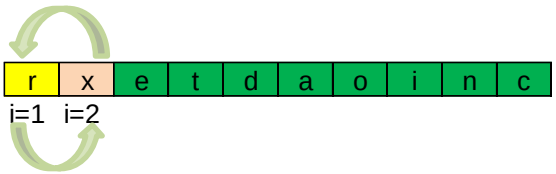
	x	r	e	t	d	a	o	i	n	c
posições	1	2	3	4	5	6	7	8	9	10

os elementos do **Vetor** com valores na ordem inicial.

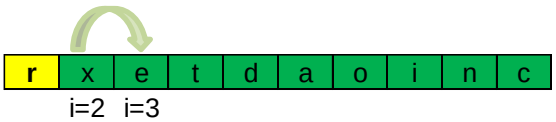
(i) **1º Passo:** Iniciar pelo primeiro elemento do vetor, na <u>posição corrente</u> (i = 1) e comparar o valor desse item com o valor do item da <u>posição seguinte</u> (i = 2);



(ii) **2º Passo:** Conforme o resultado da comparação realizada no 1º Passo, coloque item de <u>menor</u> valor na **posição corrente (i = 1)** e coloque o item de <u>maior</u> valor na **posição seguinte (i = 2)**. Registrar em uma **variável de controle** caso tenha ocorrido uma troca;

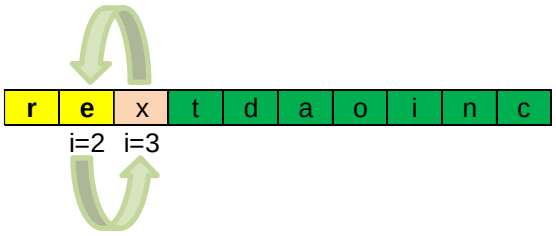


(iii) **3º Passo:** Agora o item corrente passa a ser o item seguinte do passo anterior;



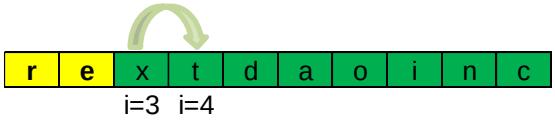
comparar o valor da posição <u>corrente (i = 2)</u> do vetor
com o valor da posição <u>seguinte (i = 3)</u> do vetor;

(iv) **4º Passo:** Conforme o resultado da comparação realizada no 3º Passo, coloque o item de <u>menor</u> valor na **posição corrente** e coloque o item de <u>maior</u> valor na **posição seguinte**. Registrar em uma **variável de controle** caso tenha ocorrido uma troca;

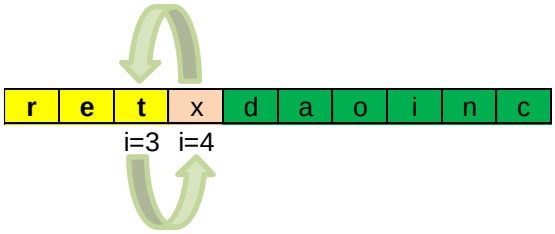


(v) **5º Passo:** Continuar executando os passos 3º e 4º até que a posição correte (i = n-1);

A seguir ilustração para a execução mais uma vez dos passos 3º e 4º



comparar o valor da posição corrente (i = 3) do vetor
com o valor da posição seguinte (i = 4) do vetor;



coloque o menor valor na **posição corrente (i = 3)**;
coloque o maior valor na **posição corrente (i = 4)** ;
Registrar na **variável de controle** a ocorrencia de troca;

(vi) **6º Passo:** Se a **variável de controle** sinalizar que houve troca, então voltar a executar o 1º Passo.

```
1  Inicio
2      Variavel Vet[50], Cont, I, Aux: Inteira;
3      Repita para I = 1, 50
4          Escreva ("Digite o elemento do vetor indice ",I);
5          Leia (Vet[I]);
6      FimRepita;
7      Cont = 1;
8      Repita Enquanto Cont = 1
9          Cont = 0;
10         Repita para I = 1, 49
11             Se (Vet[I] > Vet[I+1])
12                 Cont = 1;
13                 Aux = Vet[I];
14                 Vet[I] = Vet[I+1];
15                 Vet[I+1] = Aux;
16             FimSe;
17         FimRepita;
18     Fim Repita;
19     Escreva ("Vetor Ordenado:");
20     Repita para I = 1, 50
21         Escreva (Vet[I]);
22     FimRepita;
23 Fim.
```

3. Método **Seleção**:

É um dos modelos de algoritmos de ordenação mais simples que tem como princípio de funcionamento os procedimentos a seguir:

Vetor inicial

	x	r	e	t	d	a	o	i	n	c
posições	1	2	3	4	5	6	7	8	9	10

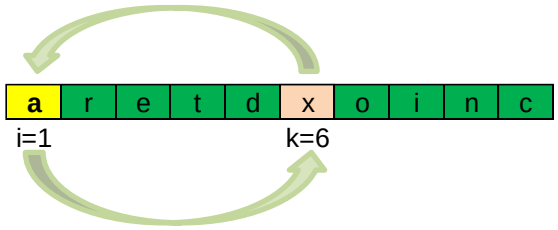
os elementos do **Vetor** com valores na ordem inicial.

(i) **1º Passo:** Pesquise e selecione o **item de menor valor** contido no vetor que se deseja ordenar e guarde a posição desse item no vetor (posição = k);

x	r	e	t	d	**a**	o	i	n	c
i = 1					k = 6				

o item de menor valor está na posição 6 do vetor;

(ii) **2º Passo:** A seguir coloque esse **item** identificado de menor valor na posição (i = 1) do vetor;



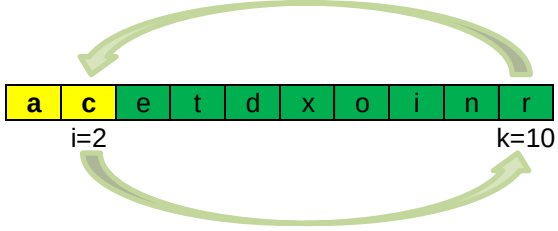
(iii) **3º Passo:** Transferira o item que estava na posição (i = 1) para a posição onde foi localizado o item de menor valor (posição = k);

(iv) **4º Passo:** A partir do item que está na **posição (i + 1)**, seguinte ao de menor valor identificado no procedimento **anterior**, pesquise e selecione o próximo item de menor valor contido no restante do vetor, nos [n - (i – 1)] elementos, e guarde a posição desse item no vetor (posição = k);

a	r	e	t	d	x	o	i	n	c
	i=2								k=10

o item de menor valor está na posição 10 do vetor;

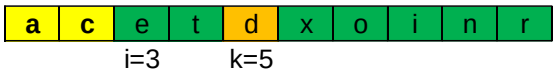
(v) **5º Passo:** A seguir coloque esse **item** identificado na posição (i + 1) do vetor;



(vi) **6º Passo:** Transferira o item que estava na posição (i + 1) para a posição onde foi localizado o item de menor valor (posição = k);

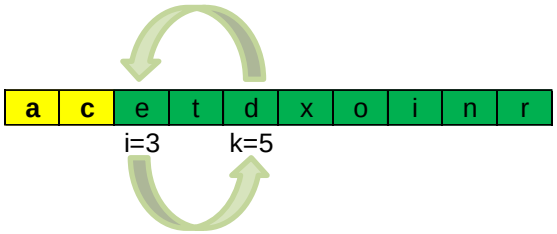
(vii) **7º Passo:** Execute os passos 4 ao 6 até que reste somente um elemento do vetor.

A seguir ilustração para a execução mais uma vez dos passos 4º ao 6º



i=3 k=5

o item de menor valor está na posição 5 do vetor;



i=3 k=5

4. Ordenação **Shell**:

O modelo de ordenação Shell foi proposto em 1959 por D. L. Shell. O método geral é derivado da ordenação por inserção. Consiste em realizar comparações, para fins de rearranjos, de itens que estão separados por *h* posições, que aqui valos chamar de "*Salto*". No caso, compara-se o item da posição "*i*" com o item da posição "*i + Salto*" e realiza-se a troca caso o item da posição "*i + Salto*" seja de valor inferior ao que está na posição "*i*".

Inicialmente, o valor do "*Salto*" pode ser igual ao valor do número de itens do vetor dividido por 2, valor arredondado. A partir desse momento passa-se a realizar comparações dos itens da posição "*i*", com valor inicial igual a 1, com os itens da posição "*i + Salto*", até que "*i + Salto*" alcance o valor do número de itens do vetor. Então determina-se um novo valor para "*Salto*" que será igual ao valor anterior de "*Salto*" dividido por 2, valor arredondado. O processo deve ser realizado até que seja concluída a comparação dos itens com o valor de "*Salto*" igual a 1.

Vetor inicial

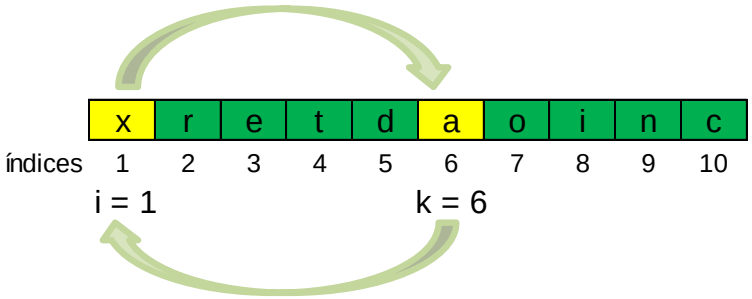
	x	r	e	t	d	a	o	i	n	c
índices	1	2	3	4	5	6	7	8	9	10

os elementos do **Vetor** com valores na ordem inicial.

(i) **1º Passo:** Determinar o valor do salto;

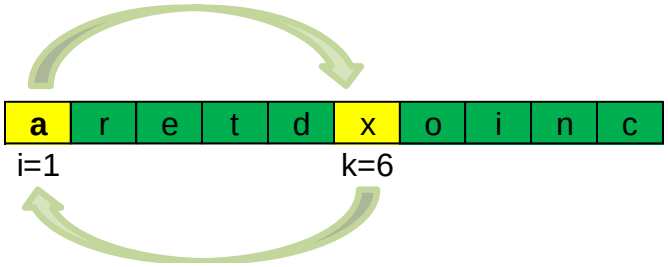
Salto = Numero_Elementos / 2 + 0,5 = 10 / 2 + 0,5 = **5**

(ii) **2º Passo:** Comparar os valores da posição ***1*** com o da posição ***1+Salto=6***;

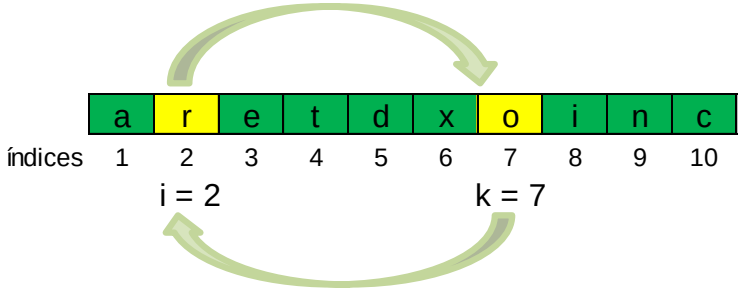


o item de menor valor está na posição 6 do vetor;

(iii) **3º Passo:** Coloque item de menor valor na posição (i = 1). Transferira o item que estava na posição (i = 1) para a posição onde estava o item de menor valor, na posição (k = 6);

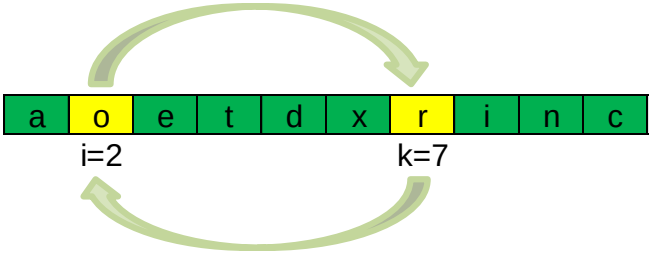


(iv) **4º Passo:** A partir do item que está na **posição (i + 1)**, seguinte ao de menor valor identificado no procedimento **anterior**, comparar o valor do item dessa posição (i = 2) com o item que está na **posição (*i* + *Salto*)** 2+Salto=7. Selecione no vetor inicial o item de menor valor;

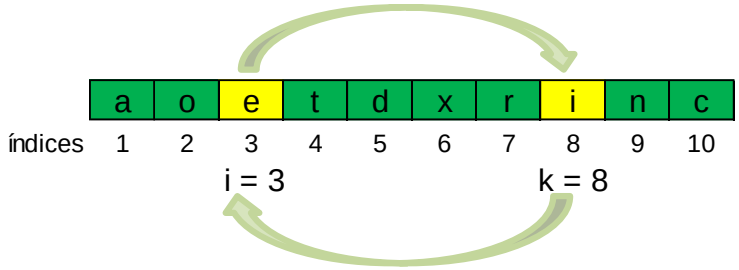


o item de menor valor está na posição 7 do vetor;

(v) **5º Passo:** Então, coloque o item de menor valor na posição (i = 2). Transferira o item que estava na posição (i = 2) para a posição onde estava o item de menor valor posição (k = 7);

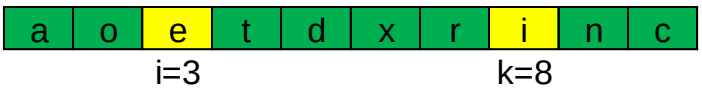


(vi) **6º Passo:** Comparar os valores da posição (i=3) com o da posição (k=3+Salto=8). Pesquise e selecione no vetor inicial o item de menor valor;



o item de menor valor está na posição (i=3) do vetor;

(vii) **7º Passo:** Deixar o item de menor valor na posição (i = 3). Nesse caso não realizar transferência. O item da posição (i = 3) é menor. Manter o item da posição (k = 8);



(viii) **8º Passo:** Continuar realizando as comparações e trocas, conforme os Passos 6 e 7, até que o valor de k seja igual ao Numero_Elementos (no exemplo k=10).

(ix) **9º Passo:** Calcular o novo valor do Salto, valor inteiro arredondado, que vai ser igual:

Salto = Salto / 2 + 0,5 = 5 / 2 + 0,5 = **3**

(x) **9º Passo:** Repetir a execução a partir do 2º Passo. Utilizar o novo valor do Salto calculado no 9º Passo.

Comparações Salto = 3:
(i=1 com k=4), (i=2 com k=5), (i=3 com k=6), (i=4 com k=7), (i=5 com k=8), (i=6 com k=9), (i=7 com k=10).

Comparações Salto = 2:
(i=1 com k=3), (i=2 com k=4), (i=3 com k=5), (i=4 com k=6), (i=5 com k=7), (i=6 com k=8), (i=7 com k=9), (i=8 com k=10).

Comparações Salto = 1:

(i=1 com k=2), (i=2 com k=3), (i=3 com k=4), (i=4 com k=5), (i=5 com k=6), (i=6 com k=7), (i=7 com k=8), (i=8 com k= 9), (i=9 com k=10).

5. Ordenação **QuickSort**:

O Quicksort é considerado o melhor algoritmo de ordenação de propósito geral, sendo baseado no método de ordenação por trocas, baseado na ideia de partições.

O procedimento geral consiste em selecionar um elemento do **vetor** cujo valor seja central dentre todos os valores dos elementos do **vetor** a ser ordenado, chamado de ***comparando*** ou de ***pivô***. Assim, fazer a partição do **vetor** em duas seções, uma contendo os elementos com valores maiores ou iguais ao valor do pivô, e a outra com os elementos cujos valores sejam menores que o valor do pivô.

Esse processo é repetido para cada seção restante até que o vetor esteja ordenado. Por exemplo, dado o **vetor** [f, e, d, a, c, b] e usando o valor de "**d**" como primeiro pivô, o primeiro passo da quicksort rearruma o vetor como apresentado a seguir:

Início: [f, c, d, a, e, b]

Passo 1: [c, b, a, d, f, e]

Esse processo é repetido para a seção 1: [c, b, a]; e para a seção 2: [d, f, e].

Portanto, a questão da ordenação consiste em rearrumar um vetor v[0 .... n-1] em ordem crescente dos valores de seus elementos, ou seja, permutar os elementos do vetor de modo que tenhamos $v[0] \leq v[1] \leq \ldots \leq v[n-1]$.

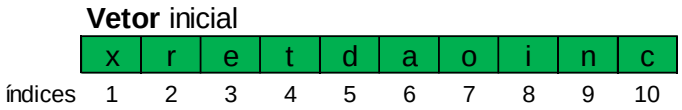
O problema da separação

O algoritmo Quicksort rearruma os valores de um vetor de forma que todos os elementos pequenos fiquem na parte esquerda do vetor e todos os elementos grandes fiquem na parte direita.

O ponto de partida para a solução desse problema é a escolha de um ***pivô*** ou ***comparando***. Os elementos do vetor que forem maiores que o ***pivô*** serão considerados grandes e os demais serão considerados pequenos. É importante escolher o ***pivô*** de tal modo que as duas partes do vetor rearrumado sejam estritamente menores que o vetor todo. A dificuldade está em resolver o problema da separação rapidamente sem usar um vetor auxiliar (como espaço de trabalho).

O algoritmo Quicksort foi desenvolvido por C.A.R. Hoare em 1962.

Exemplo:

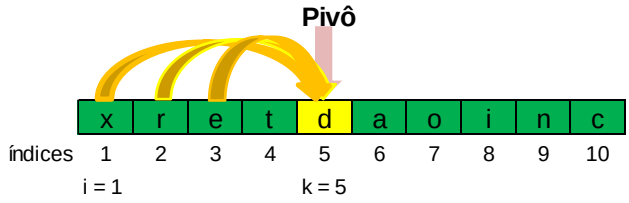


São os elementos do Vetor com valores na ordem inicial.

1º Passo: Determinar o valor da posição do Pivô. No exemplo, o valor do Pivô = “d”, na posição 5;

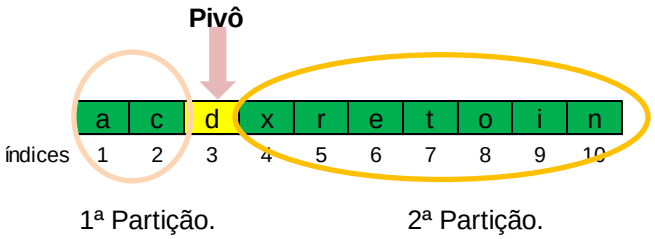
2º Passo: Comparar o valor de cada elemento do vetor com o valor do Pivô = "d". Se for menor então esse elemento deve ficar na 1ª partição;

Em caso contrário, esse elemento deve ficar na 2ª partição;

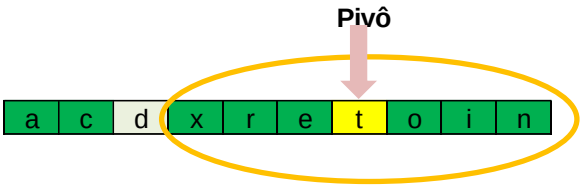


Todos os valores menores que o valor do Pivô ficarão na 1ª partição.

Todos os valores maior/igual que o valor do Pivô ficarão na 2ª partição.



3º Passo: Determinar o valor do Pivô para a 2ª partição = "t";

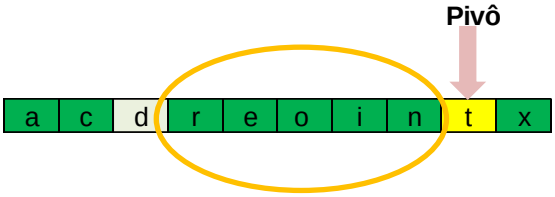


1ª Partição. 2ª Partição.

4º Passo: Comparar o valor de cada elemento da 2ª partição com o valor do Pivô da 2ª partição = "t".

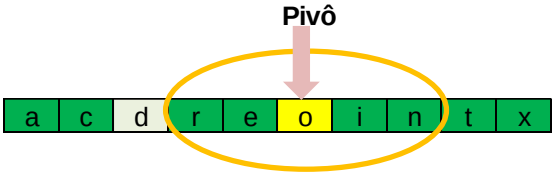
Se for menor então esse elemento deve ficar na 2ª partição da 1ª;

Em caso contrário, esse elemento deve ficar na 2ª partição da 2ª;



1ª Partição / 1ª. 2ª Partição / 1ª

5º Passo: Determinar o valor do Pivô para a 1ª partição / 2ª = "o";

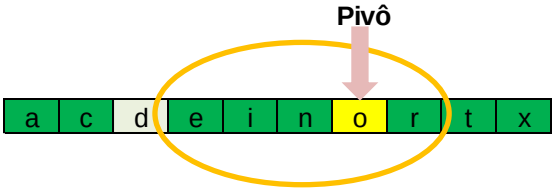


2ª Partição / 1ª

6º Passo: Comparar o valor de cada elemento da 1ª partição / 2ª com o valor do Pivô da 2ª partição / 1ª = "o".

Se for menor então esse elemento deve ficar na 2ª partição / 1ª;

Em caso contrário, esse elemento deve ficar na 2ª partição / 2ª;



1ª Partição / 2ª.

Vetor final

	a	c	d	e	i	n	o	r	t	x
índices	1	2	3	4	5	6	7	8	9	10

os elementos do **Vetor** com valores ordenados.

x-x-x-x-x-x-x-x-x-x-x-x-x-x