Algoritmo II
Busca e Ordenação

Prof. Me. Rober Marcone Rosi
Unidade de Engenharia, Computação e Sistemas

Objetivo da Aula

- Ordenação
- Ordenação por Seleção
- Ordenação por Troca ou Permutação
- Busca/Pesquisa
- Busca Linear ou Sequencial
- Busca binária

Ordenação

- Em virtude das várias aplicações, foram desenvolvidos diversos algoritmos de ordenação que consistem em realizar comparações sucessivas e trocar os elementos de posição, os quais, muitas vezes, estão relacionados ou trabalham em conjunto com algoritmos de busca, destinados a localizar determinado elemento de forma mais eficiente.

Ordenação por Seleção – Seleção Direta

- Princípios de Funcionamento
  - Selecionar o item de menor chave e trocá-lo com o item que está na primeira posição do arquivo;
  - Repetir este procedimento para os (n-1) elementos restantes, depois para os (n-2) elementos restantes, e assim sucessivamente, até se obter o arquivo ordenado.

SELECAO DIRETA

Exemplo:

Ordenação por Seleção em Java

```
1. public class Exemplo81{
2.   public static void main (String args[]){
3.     int numeros[] = {23, 4, 33, 45, 19, 12, 28, 40};
4.     int menor, x;
5.     for(int i = 0; i < numeros.length - 1; i++){
6.       menor = i;
7.       x = numeros[i];
8.       for(int j = i + 1; j < numeros.length; j++){
9.         if(numeros[j] < x){
10.          menor = j;
11.          x = numeros[j];
12.        }
13.      }
14.      numeros[menor] = numeros[i];
15.      numeros[i] = x;
16.    }
17.    for(int i = 0; i < numeros.length; i++){
18.      System.out.printf("%5d", numeros[i]);
19.    }
20.  }
21.}
```

Ordenação por Troca – Método Bolha

- O bubble sort, ou ordenação por flutuação (literalmente “por bolha”), é um algoritmo de ordenação dos mais simples. A ideia é percorrer o vetor diversas vezes, a cada passagem fazendo flutuar para o topo o maior elemento da sequência. Essa movimentação lembra a forma como as bolhas em um tanque de água procuram seu próprio nível, e disso vem o nome do algoritmo.

Ordenação por Troca – Método Bolha

- Caracterizam-se por efetuarem a classificação por comparação entre pares de chaves, trocando-as de posição caso estejam fora de ordem no par.
- Em cada passo, faz-se a troca entre os pares de itens adjacentes, de modo que ao término do primeiro passo, o item de maior chave esteja no final do vetor, ao término do segundo passo o item com a segunda maior chave esteja na penúltima posição do vetor e assim sucessivamente.
- Em cada passo a maior chave vai sendo deslocada para o final do vetor.

Ordenação por Troca – Método Bolha

BUBBLESORT

Exemplo:

vetor original					
1º varredura	**28**	**26**	30	24	25
⋮	26	**28**	**30**	24	25
⋮	26	28	**30**	**24**	25
⋮	26	28	24	**30**	**25**
fim da 1º varredura	26	28	24	25	30
2º varredura	**26**	**28**	24	25	30
. . .					
fim da 2º varredura	26	24	25	28	30
3º varredura	**26**	**24**	25	28	30
. . .					
fim da 3º varredura	24	25	26	28	30

Ordenação Método da Bolha em Java

```
1. public static void main(String[] args) {
2.    final int tam = 4;
3.    int [] vet = new int [tam];
4.    Scanner insere = new Scanner(System.in);
5.    for (int i=0; i<tam; i++){
6.        System.out.print("digite o "+(i+1)+"° número");
7.        vet[i] = insere.nextInt();
8.    }
9.    int aux;
10.   for (int i=0; i<tam-1; i++){
11.     for (int j=0; j<tam-1-i; j++){
12.         if (vet[j]>vet[j+1]){
13.             aux = vet[j];
14.             vet[j] = vet[j+1];
15.             vet[j+1] = aux;
16.         }
17.     }
18.   }
19.   for (int i=0; i<tam; i++){
20.     System.out.println(vet[i]);
21.   }
22.}
```

Ordenação Método da Bolha em Java (troca inversa do algoritmo anterior)

```
1. public class Exemplo82 {
2.   public static void main (String args[]){
3.     int numeros[] = {23, 4, 33, 45, 19, 12, 28, 40};
4.     int x;
5.     for(int i = 1; i < numeros.length; i++){
6.       for(int j = numeros.length-1; j >= i; j--){
7.         if(numeros[j-1] > numeros[j]){
8.           x = numeros[j-1];
9.           numeros[j-1] = numeros[j];
10.          numeros[j] = x;
11.        }
12.      }
13.      System.out.printf("\n%s%2d", "Iteração", i);
14.      for(int k = 0; k < numeros.length; k++)
15.        System.out.printf("%5d", numeros[k]);
16.    }
17.  }
18.}
```

Ordenação por Inserção - Inserção Direta

➢Princípios de Funcionamento:

• Neste método, o arquivo é dividido em dois segmentos. Inicialmente, o primeiro segmento contém um único elemento e consequentemente está ordenado.

• O segundo segmento contém os n-1 elementos restantes. A cada passo, a partir de i = 2, o i-ésimo elemento é transferido de segundo segmento para o primeiro, sendo inserido em sua posição apropriada.

INSERCAO DIRETA

Exemplo:

vetor original	18	15	7	9	23	16	14
divisão inicial	18	15	7	9	23	16	14
	ordenado				desordenado		
1º iteração	15	18	7	9	23	16	14
	7	15	18	9	23	16	14
	7	9	15	18	23	16	14
	7	9	15	18	23	16	14
	7	9	15	16	18	23	14
6º iteração (ordenado)	7	9	14	15	16	18	23

Exemplo completo de ordenação Inserção Direta em Java

```
1. public static void main(String[] args) {
2.   int [] vet = new int [10];
3.   Scanner insere = new Scanner(System.in);
4.   for (int i = 0; i < vet.length; i++){
5.       System.out.print("digite o "+(i+1)+"° número");
6.       vet[i] = insere.nextInt();
7.   }
8.   int i, j, temp;
7.   for (i = 1; i < vet.length; i++){
8.           temp = vet[i];
9.           j = i-1;
10.          while ((j >= 0)&&(vet[j]>temp)){
11.                  vet [j+1] = vet[j--];
12.          }
13.          vet[j+1] = temp;
14.  }
15.  for (i=0; i<vet.length; i++){
16.     System.out.println(vet[i]);
17.  }
18.}
```

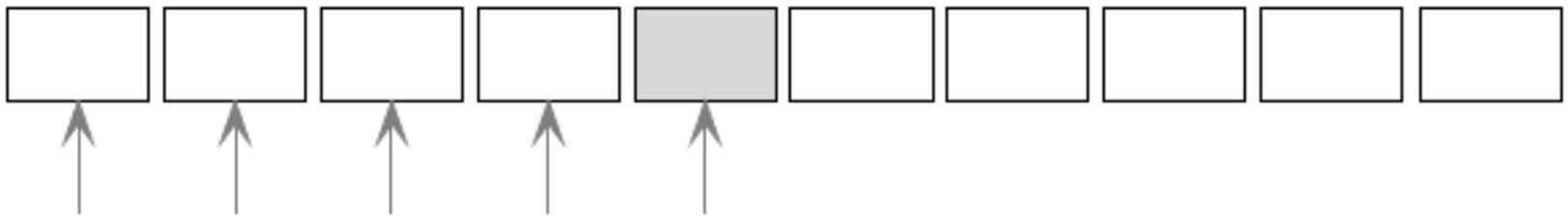
Busca/Pesquisa (1/2)

- Possuir os dados não ajuda em nada se o programador ou o usuário não souberem como recuperá-los eficientemente;
- Sistemas trabalham, frequentemente, com a busca de números, códigos, nomes, siglas e etc. e precisam de uma resposta rápida para não comprometer seu desempenho;

Busca/Pesquisa (2/2)

- Os algoritmos de busca são alguns dos mais utilizados no mundo da informática, sendo aplicados em bancos de dados, internet, jogos, entre outros;
- A escolha do método a ser utilizado para busca depende muito:
  - da quantidade de dados envolvidos,
  - do volume de operações de inclusão e exclusão a serem realizadas, entre outros.

Busca Linear ou sequencial



Comparações sucessivas são feitas entre o elemento que se procura e os elementos da lista, até que uma igualdade seja estabelecida

Exemplo 8.5 – Busca em Java

```
1.  // algoritmo de Busca {
2.   int dados[] = {23, 4, 33, 45, 19, 12, 28, 40};
3.   Scanner insere = new Scanner(System.in);
4.   System.out.print("digite um número");
5.   int x  = insere.nextInt();
6.   int i = 0;
7.   while((i < dados.length) && (x != dados[i])){
8.       i++;
9.   }
10.  if (i == dados.length){
11.    System.out.println("não achou”);
12.  }else{
13.    System.out.println(“achou e está na posição “+ i);
14.  }
```

Exemplo 8.6 – Uso de String - Java

```
1.  public class strings {
2.      public static void main(String[] args) {
3.          String dados[] = {"maria","ana", "joão", "paulo", "tadeu", "renata"};
4.          Scanner insere = new Scanner(System.in);
5.          System.out.print("digite um nome");
6.          String x  = insere.next();
7.          int i = 0;
8.          while ( (i < dados.length) &&  (!x.equalsIgnoreCase(dados[i])) ) {
9.              i++;
10.         }
11.         if (i==dados.length){
12.             System.out.println("não achou");
13.         }else{
14.             System.out.println("achou");
15.         }
16.     }
17. }
```

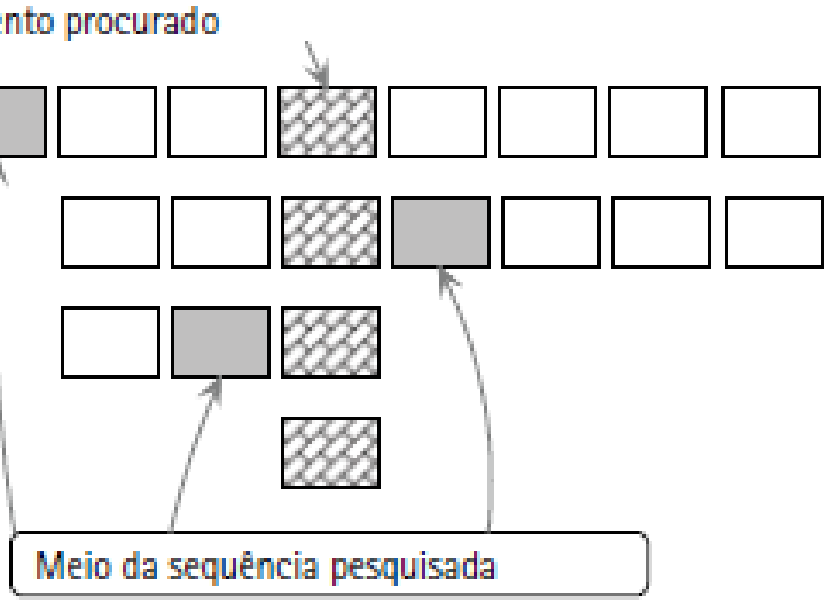
Busca Binária

- O método de busca linear é o mais adequado quando não se tem nenhuma informação a respeito da estrutura em que a busca será realizada;
- Se o elemento procurado estiver entre os últimos ou não estiver no conjunto, esse método poderá ser demasiadamente lento;
- Quando temos uma sequência ordenada de elementos, existem métodos que são muito mais adequados e eficientes por utilizar um número menor de comparações.

Busca Binária

Comparações sucessivas são feitas entre o elemento que se procura e o do meio da seqüência não pesquisada.

Cada não sucesso na busca reduz a seqüência pela metade, até que o elemento seja encontrado ou a seqüência não possa mais ser dividida.



Busca Binária em Java.
(Algoritmo em forma de função)

```
1.  public class BuscaBinaria
2.  {
3.    public static boolean binaria(int x, int numeros [])
4.    {
5.      int inicio = 0, fim = numeros.length-1;
6.      int meio;
7.      while (inicio <= fim)
8.      {
9.        meio = (inicio + fim) / 2;
10.       if (x == numeros[meio])
11.         return true;
12.       if (x < numeros[meio])
13.         fim = meio - 1;
14.       else
15.         inicio = meio + 1;
16.     }
17.     return false;
18.   }
19. }
```

Exercícios

1) Faça um programa em JAVA que cadastre 20 produtos com os seguintes atributos: código, descrição e preço, calcule e mostre a quantidade de comparações que devem ser feitas para pesquisar um código.

a) Usando pesquisa linear;

b) Usando pesquisa binária;