FACULDADE DE INFORMÁTICA
DE PRESIDENTE PRUDENTE

ALGORITMOS E TÉCNICAS DE PROGRAMAÇÃO II

Métodos de Busca e Ordenação

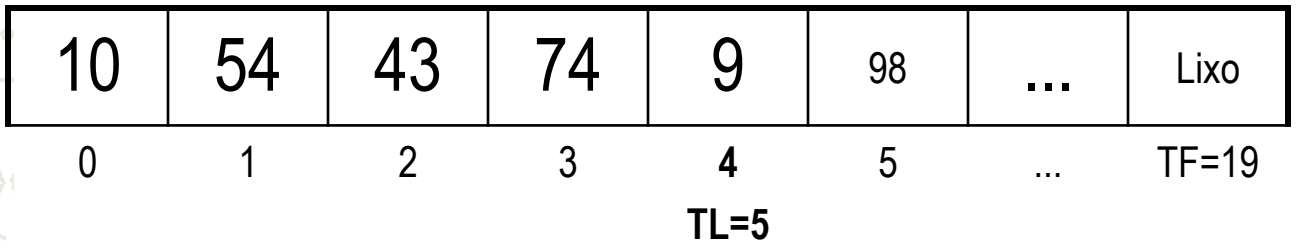
Professor:

Leandro Luiz de Almeida

Métodos de Busca

- Arranjos Desordenados:
  - Busca Exaustiva;
  - Busca Exaustiva com Sentinela.
- Arranjos Ordenados:
  - Busca Sequencial Indexada;
  - Busca Binária.

Busca Exaustiva



- O acesso às informações é realizado de forma sequencial até se encontrar o elemento ou todos os elementos do arranjo forem verificados.

Busca Exaustiva

```
int BuscaExaustiva(int v[TF], int TL, int Elem)
{
   int i=0;
   while (i<TL   &&   Elem != v[i])
                    i++;
   if (i<TL)      //achou
      return i;
   else
      return -1;
}
```

Busca com Sentinela

10	54	43	74	9	Lixo	...	Lixo
0	1	2	3	**4**	6	...	19

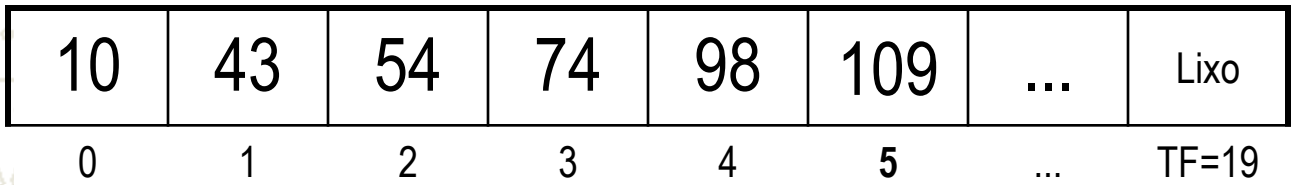
- Consiste em inserir o elemento a ser encontrado após o último componente válido do arranjo e, de forma sequencial, verificar a posição em que o mesmo se encontra.

10	54	43	74	9	**54**	...	Lixo
0	1	2	3	**4**	**5 (TL)**	...	19

Busca com Sentinela

```
int BuscaSentinela(int v[TF], int TL, int Elem)
{
   int i=0;
   v[TL] = Elem;
   while (Elem != v[i])
                  i++;
  if (i<TL)      //achou
     return i;
  else  return -1;
}
```

Busca Sequencial Indexada

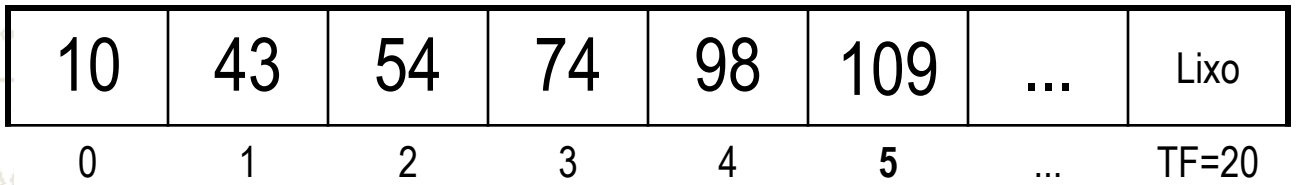


- O acesso às informações é realizado de forma sequencial até se encontrar o elemento maior ou idêntico ao que se busca, ou ainda, todos os elementos do arranjo forem verificados.

Busca Sequencial Indexada

```
int BuscaSeqInd(int v[TF], int TL, int Elem)
{
   int i=0;
   while (i<TL   &&   Elem > v[i])
                    i++;
  if (i<TL  &&  Elem == v[i])      //achou
      return i;
  else
      return -1;
}
```

Busca Binária



- A Busca Binária só pode ser executada em listas ordenadas. O método consiste em reduzir a lista sucessivamente a sublistas cada vez menores, diminuindo a *faixa* de registros em que se efetua a busca.

Busca Binária

- Como os registros estão em ordem crescente (ou decrescente), pela comparação do valor procurado *'x'* com um registro qualquer da lista, podemos tirar conclusões a respeito da localização de *'x'* na lista. Ao comparar *'x'* com um registro *A[meio]*, pode-se ocorrer:

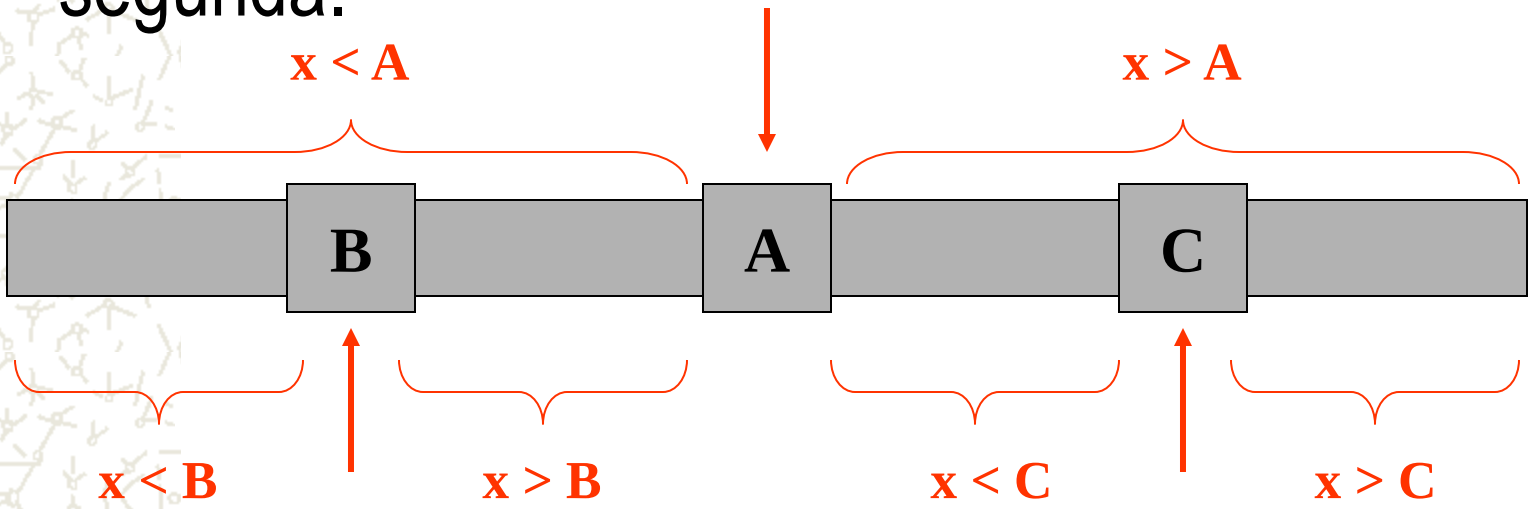
i. *x* = A[*meio*] → busca bem-sucedida;

ii. *x* < A[*meio*] → a busca deve prosseguir na sublista dos registros que procedem A[*meio*] (registros à esquerda de A[*meio*];

iii. *x* > A[*meio*] → a busca deve prosseguir na sublista da direita, formada pelos registros que sucedem A[*meio*].

Busca Binária

- Esboço gráfico:
- A busca é iniciada pelo elemento central. Se o elemento procurado for menor, procura-se novamente na primeira metade, caso contrário, na segunda.



Busca Binária

```
int BB (int v[TF],  int TL, int Elem)
{
   int inicio=0, fim=TL-1, meio;
    meio = fim/2;
    while (inicio<fim && Elem!=v[meio])
    {
        if (v[meio]<Elem)
                inicio = meio + 1;
        else    fim = meio;
        meio = (inicio+fim)/2;
    }
    if (Elem==v[meio])
            return meio;
    else    return -1;
}
```

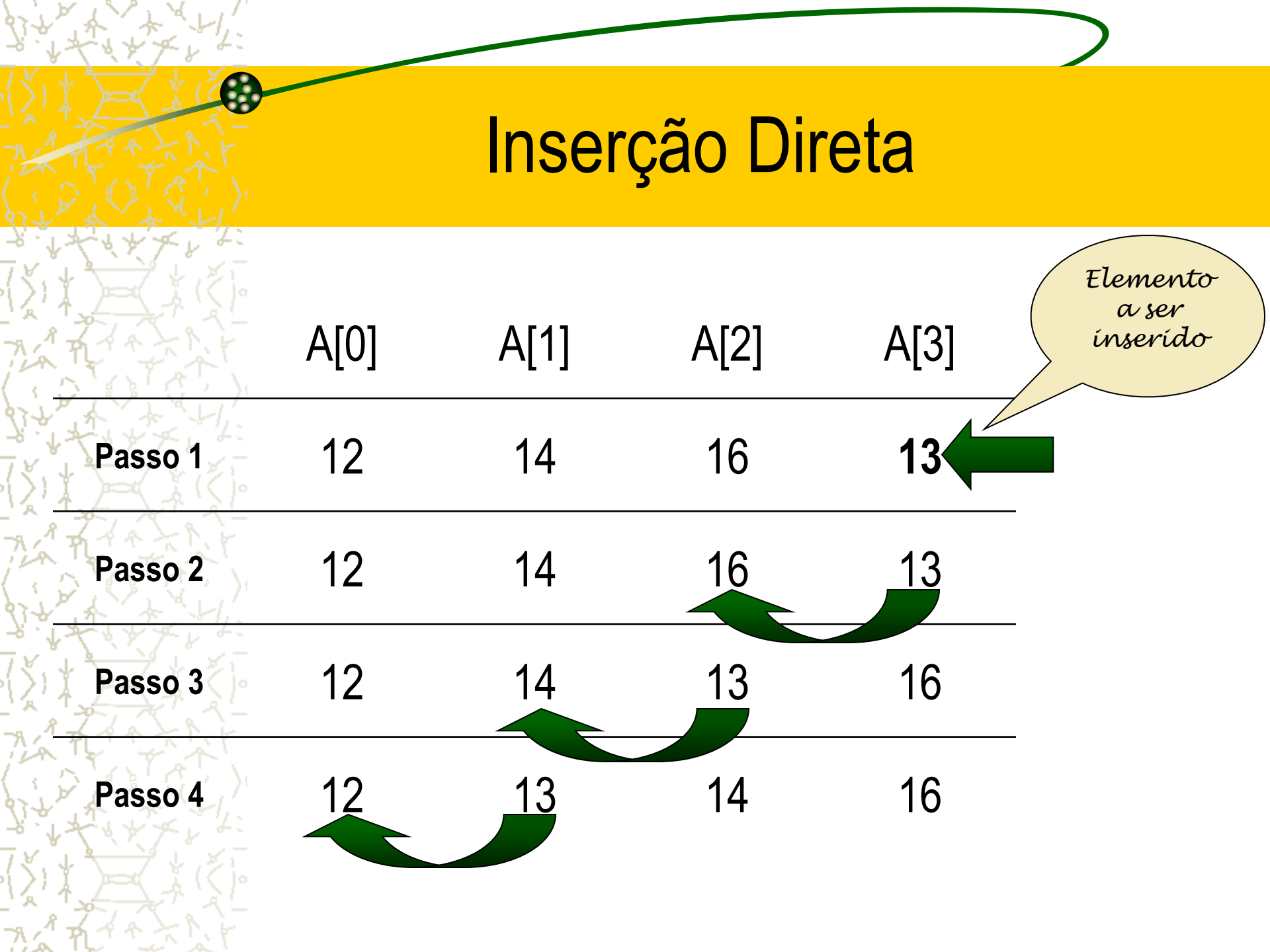
Métodos de Ordenação

- Inserção Direta *(Insertion Sort)*;
- Ordenação por Bolhas (*Bubble Sort*);
- Seleção Direta *(Selection Sort)*.

Inserção Direta

- A Inserção Direta é feita percorrendo-se os itens já ordenados da direita para esquerda, comparando-se cada item com aquele que vai ser inserido. Enquanto o item a ser inserido for menor, o item que está sendo comparado é deslocado uma posição à direita.
- O exemplo a seguir, simula a inserção do quarto item (TL=4) entre os três primeiros já ordenados.

Inserção Direta



	A[0]	A[1]	A[2]	A[3]
Passo 1	12	14	16	**13**
Passo 2	12	14	16	13
Passo 3	12	14	13	16
Passo 4	12	13	14	16

Inserção Direta

```
void InsDireta(int Vetor[TF], int TL)
{
   int p=TL-1, aux;
   while (p>0 && Vetor[p]<Vetor[p-1])
   {
      aux = Vetor[p];
      Vetor[p] = Vetor[p-1];
      Vetor[p-1] = aux;
      p--;
   }
}
```

Bubblesort (Ordenação por Bolhas)

- Um método simples de ordenação por permutação; efetuam-se varreduras repetidas sobre o vetor, deslocando-se, a cada passo, para a sua extremidade esquerda, o menor dos elementos. Se, para uma troca, o vetor for “visualizado” na posição vertical, e com o auxílio da imaginação – os elementos forem “bolhas em um tanque de água”, com densidades proporcionais ao valor das respectivas chaves, então cada varredura efetuada sobre o vetor resultaria na ascensão de uma bolha para o seu nível apropriado, de acordo com sua densidade.

Bubblesort (Ordenação por Bolhas)

Posição	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5	Passo 6	Passo 7	Passo 8
0	44	44	12	12	12	06	06	06
1	55	12	42	42	18	12	12	**12**
2	12	42	44	18	06	18	**18**	**18**
3	42	55	18	06	42	**42**	**42**	**42**
4	94	18	06	44	**44**	**44**	**44**	**44**
5	18	06	55	**55**	**55**	**55**	**55**	**55**
6	06	67	**67**	**67**	**67**	**67**	**67**	**67**
7	67	**94**	**94**	**94**	**94**	**94**	**94**	**94**

Bubblesort (Ordenação por Bolhas)

```
void OrdenaBolha (int v[TF], int TL)
{
  int   aux, a, qtde = TL;
  while (qtde>0)
  {
      for (a=0; a<qtde-1; a++)
          if (v[a] > v[a+1])
          {
             aux = v[a];
             v[a] = v[a+1];
             v[a+1] = aux;
          }
      qtde--;
  }
}
```

Desafio:

Fazer com que o algoritmo termine quando todos os elementos já estiverem na ordem correta

Seleção Direta

- 1º passo: Selecionar o maior elemento da lista e sua posição;
- 2º passo: Se a posição do maior elemento for menor que o tamanho do vetor, fazer a troca (coloca-se o maior elemento na última posição);
- 3º passo: Decrementar o tamanho, repetir os processos até o tamanho for igual a 1.

Seleção Direta

Posição	Passo 1	Passo 2	Passo 3	Passo 4	Passo 5	Passo 6	Passo 7	Passo 8	Passo 9
0	44	44	44	**44**	06	06	06	**06**	**06**
1	55	55	**55**	18	18	**18**	**12**	**12**	**12**
2	12	12	12	12	12	12	**18**	18	18
3	42	42	42	42	**42**	**42**	42	42	42
4	**94**	**67**	06	06	**44**	44	44	44	44
5	18	18	18	**55**	55	55	55	55	55
6	06	06	**67**	67	67	67	67	67	67
7	67	**94**	94	94	94	94	94	94	94

Seleção Direta

```
void SelecaoDireta (int v[TF], int TL)
{
   int Maior, PosMaior;
   while (TL > 0)
   {
      PosMaior = PosicaoMaior(v, TL);
      if (PosMaior < TL-1)
      {
         Maior = v[PosMaior];
         v[PosMaior]  = v[TL-1];
          v[TL-1] = Maior;
      }
      TL--;
   }
}
```

Seleção Direta

```
void SelecaoDireta (int v[TF], int TL)
{
  int Maior, PosMaior;
  while (TL > 0)
  {
    PosMaior = PosicaoMaior(v, TL);
    if (PosMaior < TL-1)
    {
      Maior = v[PosMaior];
      v[PosMaior]  = v[TL-1];
      v[TL-1] = Maior;
    }
    TL--;
  }
}
```

```
int PosicaoMaior (int v[TF], int TL)
{
        int PosMaior, Maior, i;
          Maior = v[0];
          PosMaior = 0;
          for (i=1; i<TL; i++)
                  if (Maior < v[i])
                  {
                      Maior = v[i];
                      PosMaior = i;
                  }
        return PosMaior;
}
```

BIBLIOGRAFIA

CELES, Waldemar; CERQUEIRA, Renato; RANGEL NETTO, José Lucas Mourão, “Introdução a estruturas de dados : com técnicas de programação em C”, Série Editora Campus. SBC - Sociedade Brasileira de Computação, 2004 – 5ª tiragem.

PEREIRA, Sílvio do Lago, “Estruturas de Dados Fundamen tais: conceitos e aplicações”, Editora Érica, 2004 – 8ª edição.

TENENBAUM, Aaron M., “Estrutura de dados Usando C“, São Paulo, Makron Books, 1995.

VILLAS, Marcos Vianna, “Estrutura de Dados“, Rio de Janeiro, Editora Campus, 1993.

WIRTH, N., “Algoritmos e estruturas de dados“, Editora Prentice Hall, 1993.