

PANC: Projeto e Análise de Algoritmos

Aula 04: Paradigma de Divisão e Conquista e Ordenação utilizando MergeSort (Lógica e Complexidade)

Lista de Exercício – Listex 03

Breno Lisi Romano

<http://sites.google.com/site/blromano>

Instituto Federal de São Paulo – IFSP São João da Boa Vista
Bacharelado em Ciência da Computação – 3º Semestre



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
SÃO PAULO
Campus São João da Boa Vista



Instruções Gerais para a Listex

■ Instruções:

- Todos os exercícios desta Listex devem ser realizados **individualmente**
- Estimula-se a **discussão** com outros **colegas** de **turma** para o **desenvolvimento**, mas **difícilmente soluções individualizadas** serão iguais → **Trabalho Idênticos serão penalizados**: Oportunidade de Aprender Errando
- Todos os **exercícios** desta Listex precisam ser **entregues na Plataforma Google Classroom**, na **atividade** criada, seguindo a data e hora definidos como **prazo de entrega**
- Quando os **exercícios envolverem programação**, **compactem o projeto com os arquivos** para solução dos exercícios, diferenciando cada um deles, o seguinte **padrão de nomes**:
 - Modelo: Listex01-Exercício01-NomeSobrenomeAluno.zip
 - Exemplo: Listex01-Exercício01-BrenoRomano.zip
- Quando os **exercícios envolverem pesquisar, textos escritos, manipulações matemáticas ou outros casos semelhantes**, **entreguem** o exercício em um arquivo na **extensão .PDF**, seguindo o **padrão de nomes**:
 - Modelo: Listex01-Exercício01-NomeSobrenomeAluno.pdf
 - Exemplo: Listex01-Exercício01-BrenoRomano.pdf



Trabalhos para Casa (1)

- **Exercício 01 – Ilustração Prática do *Mergesort*:**
 - Utilizando a Figura do Slide 25 (Ilustração do Algoritmo Exemplo Completo) como modelo, realize uma ilustração completa da operação de Ordenação por Intercalação (*Mergesort*) para o array $A = \langle 3, 41, 52, 26, 38, 57, 9, 49 \rangle$
 - Deixar claro as 03 etapas do algoritmo: Divisão, Conquista e Combinação
 - Salvar em um arquivo .doc e submeter



Trabalhos para Casa (2)

- **Exercício 02 – MergeSort Recursivo com Sentinela:**
 - Implementar o MergeSort em C (Top-Down / Recursivo) utilizando-se o algoritmo apresentado nos slides como base e também desenvolver o algoritmo Intercala adotando-se Sentinela para controle
 - Deve-se criar um algoritmo que gere um Array de números inteiros aleatórios de tamanho N, fornecido pelo usuário
 - Deve-se criar duas funções para realizar a operação de ordenação por intercalação: MergeSortRecursivo() e IntercalaComSentinela()
 - Deve-se imprimir cada etapa da ordenação na Console

```
C:\Users\blromano\Desktop\Aula04-Ex02-MergesortRecursivoComSentinela\bin\Debu...
Aula 04 - Exercício 02 - MergeSort Recursivo com Sentinela:
Entre com o tamanho do Array de Inteiros: 10
Ordenacao
Array Desord. [] = 89 26 87 49 76 8 79 32 6 69
Array Ord. [] = 89 26 87 49 76 8 79 32 6 69
Array Ord. [] = 89 26 87 49 76 8 79 32 6 69
Array Ord. [] = 26 89 87 49 76 8 79 32 6 69
Array Ord. [] = 26 89 87 49 76 8 79 32 6 69
Array Ord. [] = 26 87 89 49 76 8 79 32 6 69
Array Ord. [] = 26 87 89 49 76 8 79 32 6 69
Array Ord. [] = 26 87 89 49 76 8 79 32 6 69
Array Ord. [] = 26 49 76 87 89 8 79 32 6 69
Array Ord. [] = 26 49 76 87 89 8 79 32 6 69
Array Ord. [] = 26 49 76 87 89 8 79 32 6 69
Array Ord. [] = 26 49 76 87 89 8 79 32 6 69
Array Ord. [] = 26 49 76 87 89 8 79 32 6 69
Array Ord. [] = 26 49 76 87 89 8 79 32 6 69
Array Ord. [] = 26 49 76 87 89 8 32 79 6 69
Array Ord. [] = 26 49 76 87 89 8 32 79 6 69
Array Ord. [] = 26 49 76 87 89 8 32 79 6 69
Array Ord. [] = 26 49 76 87 89 8 32 79 6 69
Array Ord. [] = 26 49 76 87 89 6 8 32 69 79
Array Ord. [] = 6 8 26 32 49 69 76 79 87 89
Array Ord. [] = 6 8 26 32 49 69 76 79 87 89
Process returned 0 (0x0)   execution time : 1.253 s
```

5

- Implementar o MergeSort em C (Top-Down / Recursivo) utilizando-se o algoritmo apresentado nos slides como base e também desenvolver o algoritmo Intercala sem a utilização do Sentinela para controle

- ```
C:\Users\blromano\Desktop\Aula04-Ex03-MergeSort RecursivoSemSentinela\bin\Debug... - □ ×
Aula 04 - Exercício 03 - MergeSort Recursivo sem Sentinela:
Entre com o tamanho do Array de Inteiros: 12

Ordenacao
Array Desord.[] = 88 45 97 27 78 53 29 50 88 46 59 8
Array Ord. [] = 88 45 97 27 78 53 29 50 88 46 59 8
Array Ord. [] = 88 45 97 27 78 53 29 50 88 46 59 8
Array Ord. [] = 45 88 97 27 78 53 29 50 88 46 59 8
Array Ord. [] = 45 88 97 27 78 53 29 50 88 46 59 8
Array Ord. [] = 45 88 97 27 78 53 29 50 88 46 59 8
Array Ord. [] = 45 88 97 27 78 53 29 50 88 46 59 8
Array Ord. [] = 45 88 97 27 78 53 29 50 88 46 59 8
Array Ord. [] = 45 88 97 27 78 53 29 50 88 46 59 8
Array Ord. [] = 45 88 97 27 78 53 29 50 88 46 59 8
Array Ord. [] = 45 88 97 27 53 78 29 50 88 46 59 8
Array Ord. [] = 27 45 53 78 88 97 29 50 88 46 59 8
Array Ord. [] = 27 45 53 78 88 97 29 50 88 46 59 8
Array Ord. [] = 27 45 53 78 88 97 29 50 88 46 59 8
Array Ord. [] = 27 45 53 78 88 97 29 50 88 46 59 8
Array Ord. [] = 27 45 53 78 88 97 29 50 88 46 59 8
Array Ord. [] = 27 45 53 78 88 97 29 50 88 46 59 8
Array Ord. [] = 27 45 53 78 88 97 29 50 88 46 59 8
Array Ord. [] = 27 45 53 78 88 97 29 50 88 46 59 8
Array Ord. [] = 27 45 53 78 88 97 29 50 88 8 46 59
Array Ord. [] = 27 45 53 78 88 97 8 29 46 50 59 88
Array Ord. [] = 8 27 29 45 46 50 53 59 78 88 88 97
Array Ord. [] = 8 27 29 45 46 50 53 59 78 88 88 97
Process returned 0 (0x0) execution time : 1.076 s
Press any key to continue.
```





## Trabalhos para Casa (4)

- **Exercício 04 – MergeSort Iterativo sem Sentinela:**
  - Implementar o MergeSort em C (Bottom-Up / Iterativo) e utilizar o algoritmo Intercala Sem Sentinela
    - Deve-se criar um algoritmo que gere um Array de números inteiros aleatórios de tamanho N, fornecido pelo usuário
    - Deve-se criar duas funções para realizar a operação de ordenação por intercalação: MergeSortIterativo() e IntercalaSemSentinela()
    - Deve-se imprimir cada etapa da ordenação na Console

```
C:\Users\blromano\Desktop\Aula04-Ex04-MergesortIterativoSem...
Aula 04 - Exercício 04 - MergeSort Iterativo sem Sentinela:
Entre com o tamanho do Array de Inteiros: 12

Ordenacao
Array Desord. [] = 75 35 59 36 25 51 42 56 82 26 13 1
Array Ord. [] = 35 75 59 36 25 51 42 56 82 26 13 1
Array Ord. [] = 35 75 36 59 25 51 42 56 82 26 13 1
Array Ord. [] = 35 75 36 59 25 51 42 56 82 26 13 1
Array Ord. [] = 35 75 36 59 25 51 42 56 82 26 13 1
Array Ord. [] = 35 75 36 59 25 51 42 56 26 82 13 1
Array Ord. [] = 35 75 36 59 25 51 42 56 26 82 1 13
Array Ord. [] = 35 36 59 75 25 51 42 56 26 82 1 13
Array Ord. [] = 35 36 59 75 25 42 51 56 26 82 1 13
Array Ord. [] = 35 36 59 75 25 42 51 56 1 13 26 82
Array Ord. [] = 25 35 36 42 51 56 59 75 1 13 26 82
Array Ord. [] = 1 13 25 26 35 36 42 51 56 59 75 82
Array Ord. [] = 1 13 25 26 35 36 42 51 56 59 75 82
Process returned 0 (0x0) execution time : 2.543 s
Press any key to continue.
```



# Trabalhos para Casa (5)

## Exercício 05 – Busca Binária com o Paradigma de Divisão e Conquista:

- O usuário deve fornecer um **array ordenado de tamanho N** e um **valor x a ser procurado** no array
  - Deve-se partir do pressuposto que o **array** de números inteiros encontra-se **ordenado**
- Deve-se criar uma função `BuscaBinariaRecursiva()` que encontre o índice  $i$  do elemento  $x$  a ser encontrado no array, tal que  $A[i] = x$ , ou se o valor não foi encontrado
  - A função receberá o array, o início e o fim do (sub)array e o item  $x$  a ser encontrado
  - A cada iteração deve-se chamar a função `BuscaBinariaRecursiva()` recursivamente reduzindo o espaço de busca dividindo o array pela metade, através das variáveis de início e fim do sub-array

|   |    |    |   |   |    |    |    |    |    |    |
|---|----|----|---|---|----|----|----|----|----|----|
| V | 0  | 1  | 2 | 3 | 4  | 5  | 6  | 7  | 8  | 9  |
|   | -8 | -5 | 1 | 4 | 14 | 21 | 23 | 54 | 67 | 90 |

elem 4 Elemento procurado

|        |    |    |   |   |    |    |    |    |    |    |
|--------|----|----|---|---|----|----|----|----|----|----|
| meio=4 | 0  | 1  | 2 | 3 | 4  | 5  | 6  | 7  | 8  | 9  |
|        | -8 | -5 | 1 | 4 | 14 | 21 | 23 | 54 | 67 | 90 |

Valor é menor:  
buscar no início

|        |    |    |   |   |    |    |    |    |    |    |
|--------|----|----|---|---|----|----|----|----|----|----|
| meio=1 | 0  | 1  | 2 | 3 | 4  | 5  | 6  | 7  | 8  | 9  |
|        | -8 | -5 | 1 | 4 | 14 | 21 | 23 | 54 | 67 | 90 |

Valor é maior:  
buscar no final

|        |    |    |   |   |    |    |    |    |    |    |
|--------|----|----|---|---|----|----|----|----|----|----|
| meio=2 | 0  | 1  | 2 | 3 | 4  | 5  | 6  | 7  | 8  | 9  |
|        | -8 | -5 | 1 | 4 | 14 | 21 | 23 | 54 | 67 | 90 |

Valor é maior:  
buscar no final

|        |    |    |   |   |    |    |    |    |    |    |
|--------|----|----|---|---|----|----|----|----|----|----|
| meio=3 | 0  | 1  | 2 | 3 | 4  | 5  | 6  | 7  | 8  | 9  |
|        | -8 | -5 | 1 | 4 | 14 | 21 | 23 | 54 | 67 | 90 |

Valor é igual:  
terminar a busca

```
C:\Users\blromano\Desktop\Aula04-Ex05-BuscaBinariaRecursiva\bin\Debu...
Aula 04 - Exercício 05 - Busca Binaria:
Entre com o tamanho do Array de Inteiros: 12
Array Gerado Ordenado = 9 20 28 47 58 63 66 68 72 77 84 85
Entre com o valor inteiro a ser procurado: 20
O Valor 20 foi encontrado na posicao 1 do Array!
Process returned 0 (0x0) execution time : 4.201 s
Press any key to continue.
```

```
C:\Users\blromano\Desktop\Aula04-Ex05-BuscaBinariaRecursiva\bin\Debu...
Aula 04 - Exercício 05 - Busca Binaria:
Entre com o tamanho do Array de Inteiros: 12
Array Gerado Ordenado = 0 26 44 45 45 47 48 57 68 80 88 89
Entre com o valor inteiro a ser procurado: 1
O Valor 1 nao foi encontrado no Array!
Process returned 0 (0x0) execution time : 4.058 s
Press any key to continue.
```

- Em um arquivo .doc, analisar o Algoritmo desenvolvido:
  - Identificar as etapas da Divisão e Conquista (Dividir, Conquistar e Combinar) no algoritmo
  - Realizar a Análise da Complexidade  $T(n)$ : Número de Operações, Fórmula de Recorrência e Árvore (Similar ao *Mergesort*)
  - Entregar em um arquivo zipado: o .doc e o algoritmo da busca binária desenvolvida