**FACULDADE DE COMPUTAÇÃO E INFORMÁTICA**

**CIÊNCIAS DA COMPUTAÇÃO E SISTEMAS DE INFORMAÇÃO**

**Linguagem de Programação I**

**AULA 08: ORDENAÇÃO EM LISTAS**

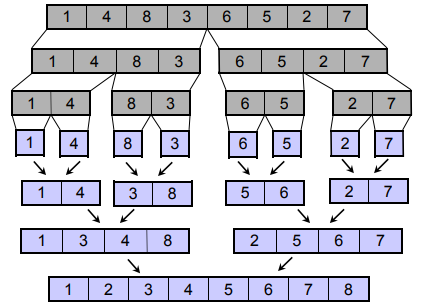
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  | | --- | --- | |  | Nossos objetivos nesta aula são:   * Conhecer o problema da ordenação em listas. * Implementar as estratégias de ordenação pelo método Merge (*MergeSort*) | |  | As referências para esta aula são: | |  | O **Capítulo 11** do livro:  DIERBACH, C. *Introduction to Computer Science Using Python: A Computational Problem Solving Focus.* 1st Edition, New York: Wiley, 2012. | |  |  | | Macintosh HD:Users:lucianosilva:Desktop:6780034g1.jpg | o Capítulo 14, Seção 14.6 (Classificação) do livro:  HORSTMANN, C. Conceitos de Computação com Java. 5.ed. Porto Alegre: Bookman, 2009. | |

# O MÉTODO DE ORDENAÇÃO MERGESORT

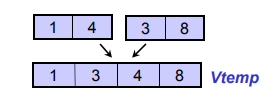
O Método de Ordenação por Intercalação **(Mergesort)** é um outro algoritmo clássico de ordenação de listas. A ideia básica deste algoritmo é ir subdividindo a lista até que se encontre um subvetor que saibamos ordenar facilmente (por exemplo, uma lista com apenas um elemento). Uma vez que os sublistas estejam ordenadas, o algoritmo faz uso de uma rotina de intercalação (merge) para juntá-los. Para uma introdução lúdica ao algoritmo MergeSort, veja o vídeo abaixo:

https://www.youtube.com/watch?v=XaqR3G\_NVoo

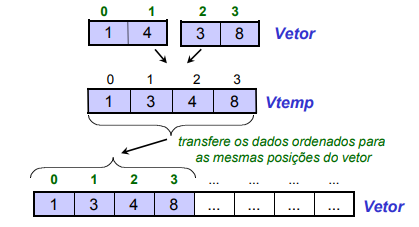
A imagem abaixo ilustra a execução do algoritmo MergeSort:



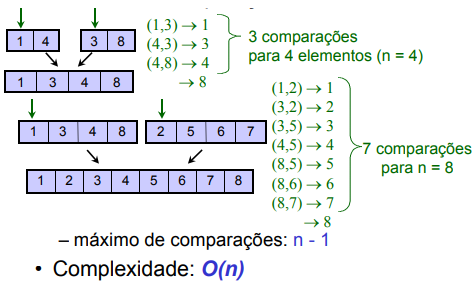
O algoritmo MergSort utiliza um vetor temporário (Vtemp) para manter o resultado da ordenação dos 2 sub-vetores



Após a ordenação, o conteúdo de Vtemp é transferido para o vetor



Qual seria o número de comparações?



**EXERCÍCIO**

1. Implemente um procedimento chamado **merge**, que recebe dois trechos ordenados de um vetor e produz a intercalação ordenada destes dois trechos, usando a descrição abaixo:

**Algoritmo** merge (V,L,R)

**Entrada**: A lista L e as sublistas L e R

**Saída**: A lista L em ordem crescente

1. nL=len(L)
2. nR=len(R)
3. i=0
4. j=0
5. k=0
6. **enquanto** (i<nL e j<nR) **faça**
7. **se** (L[i]<=R[j])
8. **então** V[k]=L[i]
9. i=i+1
10. **senão** V[k]=R[j]
11. j=j+1
12. **fimse**
13. k=k+1
14. **fimenquanto**
15. **enquanto** (i<nL) **faça**
16. V[k]=L[i]
17. i=i+1
18. k=k+1
19. **fimenquanto**
20. **enquanto** (j<nR) **faça**
21. V[k]=R[j]
22. j=j+1
23. k=k+1
24. **fimequanto**
25. Implemente o Algoritmo Mergesort descrito abaixo:

**Algoritmo MergeSort (V)**

**Entrada**: Uma lista V

**Saída**: A lista L ordenada segundo a estratégia de ordenação por intercalação

1. n=len(V)
2. Se n>1 **então**
3. **meio=n//2**
4. **para i de 0 até meio-1** **faça**
5. left[i]=V[i]
6. **fimpara**
7. **para i de meio até n-1** **faça**
8. right[i-meio]=V[i]
9. **fimpara**
10. mergeSort(left)
11. mergeSort(right)
12. merge(V,left, right)