Estruturas de Dados I

Ordenação - Divisão e Conquista - MergeSort

Prof. Dr. Lidio Mauro Lima de Campos

limadecampos@gmail.com

Universidade Federal do Pará – UFPA
ICEN
FACOMP

Agenda

- **Introdução (Sorting).**
- **Abordagem Dividir para Conquistar**
  - **Algoritmo de Ordenação por Intercalação (Merge Sort)**
  - **Complexidade.**

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral: MERGE (Fusão) de dois arrays já ordenados**

n=13

a	**7**	**10**	**19**	**20**	**32**	**39**	**43**	**48**	**60**	**74**	**91**	**98**	**99**
	i	i											

m=11

b	**9**	**12**	**17**	**19**	**28**	**30**	**31**	**50**	**80**	**88**	**89**
	j										

if (a[i]≤b[j]) {c[k]=a[i];i++,k++;}

m+n=24

c	**7**																							
	k	k																						

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a													
	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i											

m=11

b											
	9	12	17	19	28	30	31	50	80	88	89
	j	j									

if (a[i]≤b[j]) {c[k]=a[i];i++,k++;}

m+n=24

c																								
	7	9																						
	k	k	k																					

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a

7	10	19	20	32	39	43	48	60	74	91	98	99
i	i	i										

m=11

b

9	12	17	19	28	30	31	50	80	88	89
j	j									

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c

7	9	10																					
k	k	k	k																				

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a

7	10	19	20	32	39	43	48	60	74	91	98	99
i	i	i										

m=11

b

9	12	17	19	28	30	31	50	80	88	89
j	j	j								

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c

7	9	1 0	1 2																				
k	k	k	K	k																			

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a	**7**	**10**	**19**	**20**	**32**	**39**	**43**	**48**	**60**	**74**	**91**	**98**	**99**
	i	i	i										

m=11

b	**9**	**12**	**17**	**19**	**28**	**30**	**31**	**50**	**80**	**88**	**89**
	j	j	j	j							

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c	**7**	**9**	**10**	**12**	**17**																			
	K	K	K	K	K	K																		

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a													
	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i	i	i									

m=11

b											
	9	12	17	19	28	30	31	50	80	88	89
	j	j	j	j							

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c																								
	7	9	10	12	17	19																		
	k	k	k	K	k	k	k																	

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a

7	10	19	20	32	39	43	48	60	74	91	98	99
i	i	i	i									

m=11

b

9	12	17	19	28	30	31	50	80	88	89
j	j	j	j	j						

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c

7	9	10	12	17	19	19																	
k	k	k	K	k	k	k	k																

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a													
a	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i	i	i	i								

m=11

b											
b	9	12	17	19	28	30	31	50	80	88	89
	j	j	j	j	j						

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c	7	9	10	12	17	19	19	20																
	k	k	k	K	k	k	k	k	k															

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a													
	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i	i	i	i								

m=11

b											
	9	12	17	19	28	30	31	50	80	88	89
	j	j	j	j	j	j					

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c																								
	7	9	10	12	17	19	19	20	28															
	k	k	k	K	k	k	k	k	K	K														

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a													
	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i	i	i	i								

m=11

b											
	9	12	17	19	28	30	31	50	80	88	89
	j	j	j	j	j	j	j				

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c																								
	7	9	10	12	17	19	19	20	28	30														
	k	k	k	K	k	k	k	k	K	K														

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a	**7**	**10**	**19**	**20**	**32**	**39**	**43**	**48**	**60**	**74**	**91**	**98**	**99**
	i	i	i	i	i								

m=11

b	**9**	**12**	**17**	**19**	**28**	**30**	**31**	**50**	**80**	**88**	**89**
	j	j	j	j	j	j	j	j			

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c	**7**	**9**	**10**	**12**	**17**	**19**	**19**	**20**	**28**	**30**	**31**													
	K	K	K	K	K	K	K	K	K	K	K	K												

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a													
	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i	i	i	i	i							

m=11

b											
	9	12	17	19	28	30	31	50	80	88	89
	j	j	j	j	j	j	j	j			

if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}

m+n=24

c																								
	7	9	10	12	17	19	19	20	28	30	31	32												
	K	K	K	K	K	K	K	K	K	K	K	K	K											

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a

7	10	19	20	32	39	43	48	60	74	91	98	99
i	i	i	i	i	i	i						

m=11

b

9	12	17	19	28	30	31	50	80	88	89
j	j	j	j	j	j	j	j			

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c

7	9	10	12	17	19	19	20	28	30	31	32	39											
K	K	K	K	K	K	K	K	K	K	K	K	K	K										

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a													
	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i	i	i	i	i	i	i					

m=11

b											
	9	12	17	19	28	30	31	50	80	88	89
	j	j	j	j	j	j	j	j			

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c																								
	7	9	10	12	17	19	19	20	28	30	31	32	39	43										
	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K									

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a													
	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i	i	i	i	i	i	i	i				

m=11

b											
	9	12	17	19	28	30	31	50	80	88	89
	j	j	j	j	j	j	j	j			

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c																								
	7	9	10	12	17	19	19	20	28	30	31	32	39	43	48									
	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K								

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a	**7**	**10**	**19**	**20**	**32**	**39**	**43**	**48**	**60**	**74**	**91**	**98**	**99**
	i	i	i	i	i	i	i	i	i				

m=11

b	**9**	**12**	**17**	**19**	**28**	**30**	**31**	**50**	**80**	**88**	**89**
	j	j	j	j	j	j	j	j	j		

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c	**7**	**9**	**10**	**12**	**17**	**19**	**19**	**20**	**28**	**30**	**31**	**32**	**39**	**43**	**48**	**50**								
	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K							

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i	i	i	i	i	i	i	i	i			

m=11

b	9	12	17	19	28	30	31	50	80	88	89
	j	j	j	j	j	j	j	j	j		

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c	7	9	10	12	17	19	19	20	28	30	31	32	39	43	48	50	60							
	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K						

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a													
	7	10	19	20	32	39	43	48	60	**74**	91	98	99
	i	i	i	i	i	i	i	i	i	i	i		

m=11

b											
	9	12	17	19	28	30	31	50	**80**	88	89
	j	j	j	j	j	j	j	j	j		

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c																								
	7	9	10	12	17	19	19	20	28	30	31	32	39	43	48	50	60	74						
	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K					

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a													
	7	**10**	**19**	**20**	**32**	**39**	**43**	**48**	**60**	**74**	**91**	**98**	**99**
	i	i	i	i	i	i	i	i	i	i	i		

m=11

b											
	9	**12**	**17**	**19**	**28**	**30**	**31**	**50**	**80**	**88**	**89**
	j	j	j	j	j	j	j	j	j	j	

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c																								
	7	**9**	**10**	**12**	**17**	**19**	**19**	**20**	**28**	**30**	**31**	**32**	**39**	**43**	**48**	**50**	**60**	**74**	**80**					
	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K				

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i	i	i	i	i	i	i	i	i	i		

m=11

b	9	12	17	19	28	30	31	50	80	88	89
	j	j	j	j	j	j	j	j	j	j	j

```
if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}
```

m+n=24

c	7	9	10	12	17	19	19	20	28	30	31	32	39	43	48	50	60	74	80	88			
	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K		

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a													
	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i	i	i	i	i	i	i	i	i	i		

m=11

b											
	9	12	17	19	28	30	31	50	80	88	89
	j	j	j	j	j	j	j	j	j	j	j

if (a[i]≤b[j])
 {c[k]=a[i];i++,k++;}
else
{c[k]=b[j];j++,k++;}

m+n=24

c																								
	7	9	10	12	17	19	19	20	28	30	31	32	39	43	48	50	60	74	80	88	89			
	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K		

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a

7	10	19	20	32	39	43	48	60	74	91	98	99
i	i	i	i	i	i	i	i	i	i	i	i	

m=11

b

9	12	17	19	28	30	31	50	80	88	89
j	j	j	j	j	j	j	j	j	j	j

```
while (j<a. length && i<b.length)
{
 if (a[i]≤b[j])
  {c[k]=a[i];i++,k++;}
 else
 {c[k]=b[j];j++,k++;}
}
while(i<a. length)
{c[k]=a[i];i++,k++;}
while(j<b. length)
{c[k]=b[j];i++,k++;}
```

c

m+n=24

7	9	1 0	1 2	1 7	1 9	1 9	2 0	2 8	3 0	3 1	3 2	3 9	4 3	4 8	5 0	6 0	7 4	8 0	8 8	8 9	9 1		
K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i	i	i	i	i	i	i	i	i	i	i	i

m=11

b	9	12	17	19	28	30	31	50	80	88	89
	j	j	j	j	j	j	j	j	j	j	j

```
while (j<a. length && i<b.length)
{
 if (a[i]≤b[j])
  {c[k]=a[i];i++,k++;}
 else
 {c[k]=b[j];j++,k++;}
}
while(i<a. length)
{c[k]=a[i];i++,k++;}
while(j<b. length)
{c[k]=b[j];i++,k++;}
```

c

m+n=24

7	9	10	12	17	19	19	20	28	30	31	32	39	43	48	50	60	74	80	88	89	91	98	
K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K

Ordenação por Intercalação (Merge Sort)

- **Ideia Geral:**

n=13

a	7	10	19	20	32	39	43	48	60	74	91	98	99
	i	i	i	i	i	i	i	i	i	i	i	i	i

m=11

b	9	12	17	19	28	30	31	50	80	88	89
	j	j	j	j	j	j	j	j	j	j	j

```
while (j<a. length && i<b.length)
{
 if (a[i]≤b[j])
  {c[k]=a[i];i++,k++;}
 else
 {c[k]=b[j];j++,k++;}
}
while(i<a. length)
{c[k]=a[i];i++,k++;}
while(j<b. length)
{c[k]=b[j];i++,k++;}
```

c

m+n=24

7	9	10	12	17	19	19	20	28	30	31	32	39	43	48	50	60	74	80	88	89	91	98	99
K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K	K

Ordenação por Intercalação (Merge Sort)

- O que fazemos então é **organizar os dados no array a ser ordenado de forma que uma parte dele esteja ordenada e outra também**.
- Assim, **no Merge Sort não fazemos o merge de dois arrays**, mas **fazemos o merge de duas partes ordenadas de um mesmo array**.

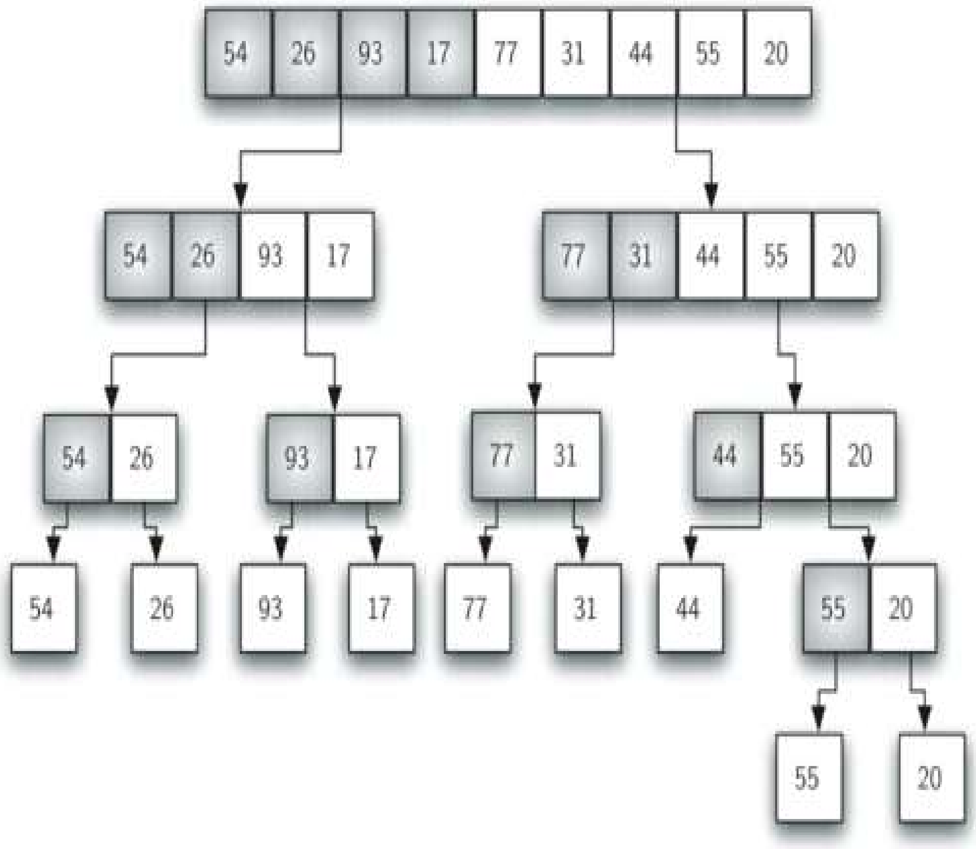
Ordenação por Intercalação (Merge Sort)

- Agora nós voltamos nossa atenção para usar a estratégia de "**dividir para conquistar**" como uma forma de melhorar o desempenho dos algoritmos de ordenação.
- O primeiro algoritmo que iremos estudar é o Merge Sort, um **algoritmo recursivo que divide uma lista continuamente pela metade**.
- Se a **lista** estiver **vazia** ou **tiver um único item**, ela **está ordenada por definição (o caso base).**

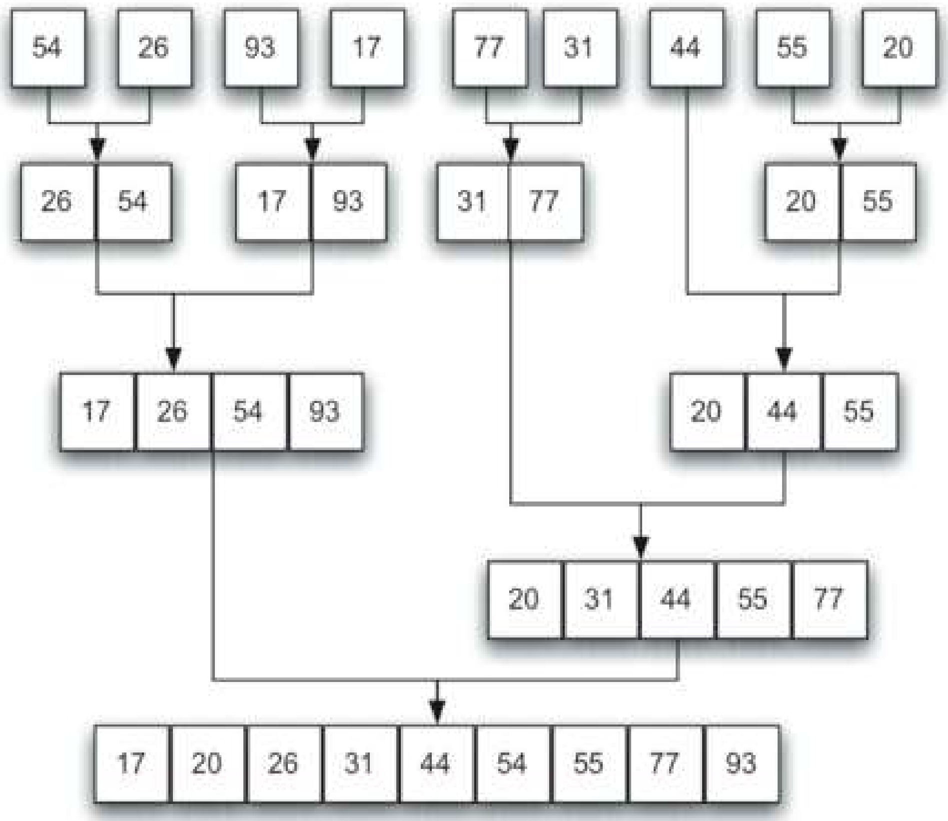
Ordenação por Intercalação (Merge Sort)

- Se a **lista tiver mais de um item**, **dividimos a lista e invocamos recursivamente um Merge Sort em ambas as metades**.
- Assim **que as metades estiverem ordenadas**, **a operação fundamental**, chamada de **intercalação (Merge)**, é realizada.
- Intercalar **é o processo de pegar duas listas menores ordenadas e combiná-las de modo a formar uma lista nova, única e ordenada**.

Ordenação por Intercalação (Merge Sort)



Dividindo a Lista no Merge Sort



Intercalação das Listas

Ordenação por Intercalação (Merge Sort)

https://www.youtube.com/watch?v=3j0SWDX4AtU

```python
def mergeSort(alist):
    print("Splitting ",alist)
    if len(alist)>1:
        mid = len(alist)//2
        lefthalf = alist[:mid]
        righthalf = alist[mid:]
        mergeSort(lefthalf)
        mergeSort(righthalf)
        i=0
        j=0
        k=0
        while i < len(lefthalf) and j < len(righthalf):
            if lefthalf[i] < righthalf[j]:
                alist[k]=lefthalf[i]
                i=i+1
            else:
                alist[k]=righthalf[j]
                j=j+1
            k=k+1

        while i < len(lefthalf):
            alist[k]=lefthalf[i]
            i=i+1
            k=k+1

        while j < len(righthalf):
            alist[k]=righthalf[j]
            j=j+1
            k=k+1
    print("Merging ",alist)
alist = [54,44,55,20]
mergeSort(alist)
print(alist)
```

Ordenação por Intercalação (Merge Sort) – Análise de Complexidade

- **Divisão : A etapa de divisão simplesmente calcula o ponto médio do subarranjo, o que demora um tempo constante.**
- **Conquista: Resolvemos recursivamente dois subproblemas, cada um de tamanho n/2, o que contribui com 2T(n/2) para o tempo de execução.**
- **Combinação: Já observamos que o procedimento MERGE em um subarranjo de n elementos leva o tempo Ɵ(n).**

Ordenação por Intercalação (Merge Sort) – Análise de Complexidade

- **Passo Base: Um vetor com um elemento já esta ordenado.**
- **T(1)=ϴ(1) ou T(1)=1**
- **Passo Recorrente: Um vetor com mais de 1(um) elemento**
- **T(n)=** $T\left(\frac{n}{2}\right) + T\left(\frac{n}{2}\right)$ **+ ϴ(n), n>1**
- **T(n)=2T**$\left(\frac{n}{2}\right)$**+n , n>1**

Ordenação por Intercalação (Merge Sort) – Análise de Complexidade

- **Expandir: T(n)=2T$\left(\frac{n}{2}\right)$+n , n>1**
- **K=1 , T(n)=2T$\left(\frac{n}{2}\right)$+n , T(n/2)=2T(n/2/2)+n/2=2T$\left(\frac{n}{4}\right)$+$\frac{n}{2}$**
- **K=2 , T(n)=2[2T$\left(\frac{n}{4}\right)$+$\frac{n}{2}$]+n= 4T$\left(\frac{n}{4}\right)$+2n , T$\left(\frac{n}{4}\right)$=2T(n/4/2)+n/4=2T$\left(\frac{n}{8}\right)$+$\frac{n}{4}$**
- **K=3, T(n)=4[2T$\left(\frac{n}{8}\right)$+$\frac{n}{4}$]+2n= 8T$\left(\frac{n}{8}\right)$+3n**
- **Conjecturar:**
- **T(n)=$2^k . T\left(\frac{n}{2^k}\right) + kn$**
- **A expansão irá para n=2^k → k=log_2^n**
- **T(n)=n.T(n/n)+nlog_2^n=n.T(1)+nlog_2^n**
- **T(n)=n+nlog_2^n**
- **ϴ(nlog_2^n)**

Ordenação por Intercalação (Merge Sort) – Análise de Complexidade

- https://www.youtube.com/watch?v=3j0SWDX4AtU

Referências

- Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Algoritmos – Teoria e Prática, Tradução da Segunda Edição. Campus, 2016.
- Neto, Nelson Cruz Sampaio. Notas de Aula, P.A. Algoritmos, 2021.
- U. Manber, Algorithms: A Creative Approach, Addison-Wesley (1989).
- J. Kleinberg e E. Tardos, Algorithm Design, Addison Wesley, (2005).