Processamento da Informação
– Teoria –

Algoritmos de ordenação (Terceira parte)

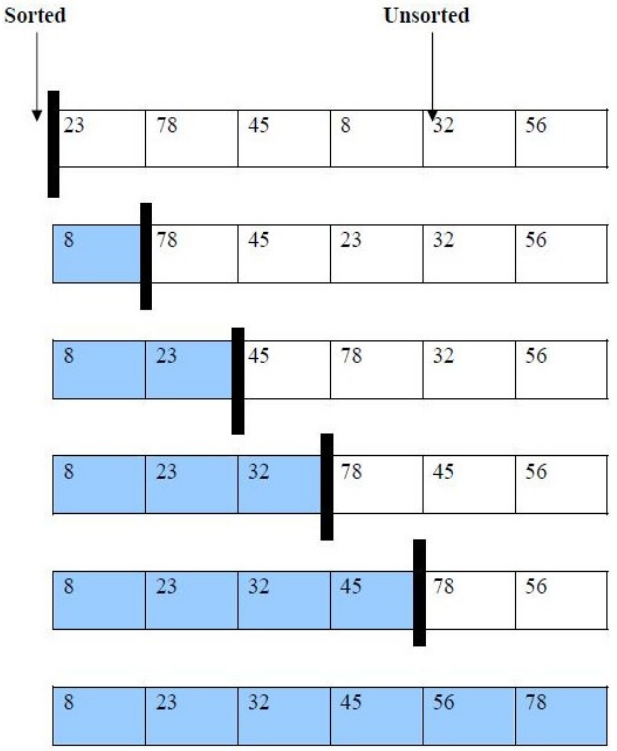
Semana 10

Prof. Jesús P. Mena-Chalco

26/06/2013

A. Selection sort

```
def selection_sort(L):
    for i in range(0,len(L)-1):
        menor = i
        for j in range(i+1, len(L)):
            if L[menor]>L[j]:
                menor = j
        L[i], L[menor] = L[menor],L[i]
    return L
```

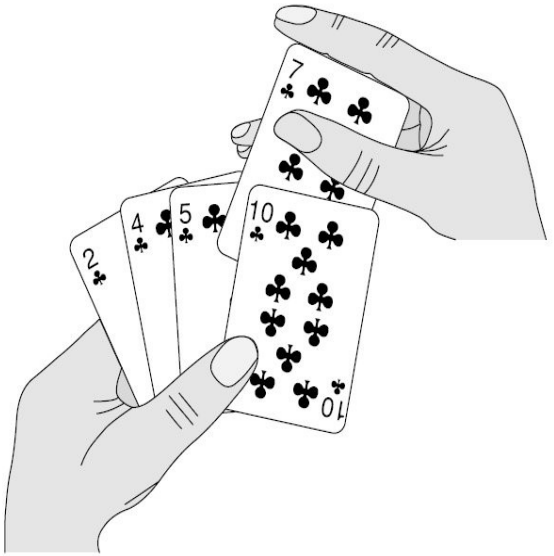


B. Bubble sort

```
def bubble_sort(L):
    j = len(L)-1
    while j>0:
        for i in range(0,j):
            if L[i]>L[i+1]:
                L[i],L[i+1] = L[i+1],L[i]
        j = j-1
    return L
```

C. Insertion sort

```
def insertion_sort(L):
    for i in range(1,len(L)):
        j = i
        while j>=1 and L[j-1]>L[j]:
            L[j-1],L[j] = L[j],L[j-1]
            j = j-1
    return L
```



Classificação de algoritmos

A escolha de um algoritmo para ordenar uma lista de ***n*** elementos é o tempo gasto para ordena-lo.

Podemos classificar os algoritmos em 2 tipos:

- Ordenação interna.
- Ordenação externa.

Uma outra alternativa de classificação:

- Ordenação baseada em comparações.
- Ordenação não baseada em comparações.

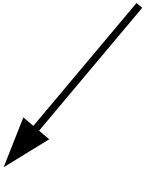
Classificação de algoritmos

A escolha de um algoritmo para ordenar uma lista de ***n*** elementos é o tempo gasto para ordena-lo.

Podemos classificar os algoritmos em 2 tipos:

- Ordenação interna.
- Ordenação externa.

Os três algoritmos tratados até agora podem ser classificados como



Uma outra alternativa de classificação:

- Ordenação baseada em comparações.
- Ordenação não baseada em comparações.

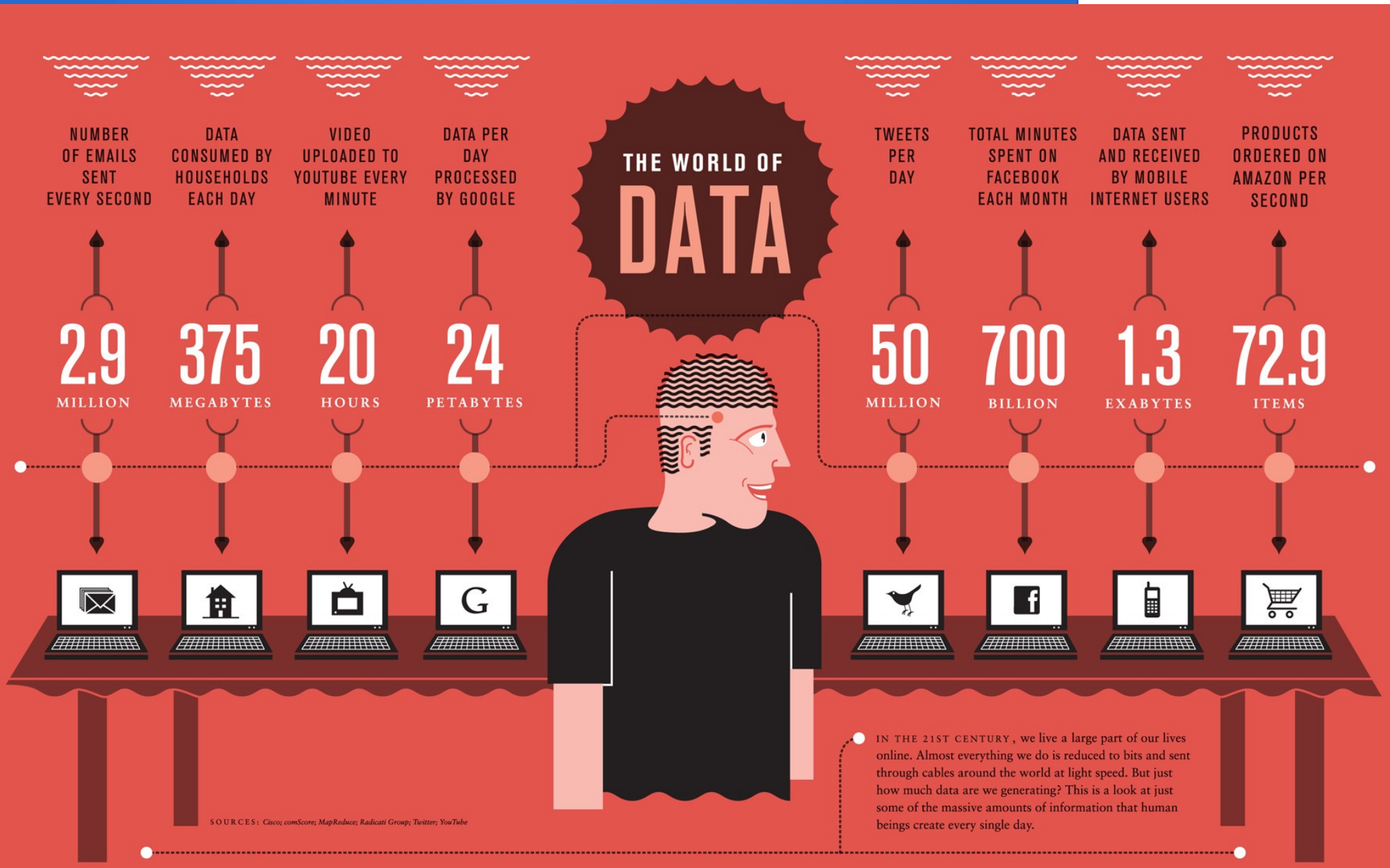
Ordenação interna vs externa

Se a lista a ser ordenada cabe toda na memória principal, então o método de ordenação é chamado de **ordenação interna**.

Neste caso o número de elementos é pequeno o bastante para caber em uma Lista.

Se os elementos não cabem na memória principal, e por isso, tem de ser armazenado em disco, então o método de ordenação é chamado de **ordenação externa**.

Big data


NUMBER OF EMAILS SENT EVERY SECOND
2.9
MILLION
DATA CONSUMED BY HOUSEHOLDS EACH DAY
375
MEGABYTES
VIDEO UPLOADED TO YOUTUBE EVERY MINUTE
20
HOURS
DATA PER DAY PROCESSED BY GOOGLE
24
PETABYTES
THE WORLD OF
DATA
TWEETS PER DAY
50
MILLION
TOTAL MINUTES SPENT ON FACEBOOK EACH MONTH
700
BILLION
DATA SENT AND RECEIVED BY MOBILE INTERNET USERS
1.3
EXABYTES
PRODUCTS ORDERED ON AMAZON PER SECOND
72.9
ITEMS
SOURCES: Cisco; comScore; MapReduce; Radicati Group; Twitter; YouTube
IN THE 21ST CENTURY, we live a large part of our lives online. Almost everything we do is reduced to bits and sent through cables around the world at light speed. But just how much data are we generating? This is a look at just some of the massive amounts of information that human beings create every single day.
A COLLABORATION BETWEEN GOOD AND OLIVER MUNDAY
IN PARTNERSHIP WITH IBM

Big data

Prefix	Power	Decimal	Name
yotta	10^{24}	1,000,000,000,000,000,000,000,000	septillion
zetta	10^{21}	1,000,000,000,000,000,000,000	sextillion
exa	10^{18}	1,000,000,000,000,000,000	quintillion
peta	10^{15}	1,000,000,000,000,000	quadrillion
tera	10^{12}	1,000,000,000,000	trillion
giga	10^{9}	1,000,000,000	billion
mega	10^{6}	1,000,000	million
kilo	10^{3}	1,000	thousand
(none)	10^{0}	1	one
milli	10^{-3}	.001	thousandth
micro	10^{-6}	.000 001	millionth
nano	10^{-9}	.000 000 001	billionth
pico	10^{-12}	.000 000 000 001	trillionth
femto	10^{-15}	.000 000 000 000 001	quadrillionth
atto	10^{-18}	.000 000 000 000 000 001	quintillionth
zepto	10^{-21}	.000 000 000 000 000 000 001	sextillionth
yocto	10^{-24}	.000 000 000 000 000 000 000 001	septillionth

Ordenação interna vs externa

- Ordenação interna: Cada elemento pode ser imediatamente acessado.

- Ordenação externa: Os elementos são acessados sequencialmente ou em grandes blocos.

Ordenação *in situ*

A quantidade extra de memória auxiliar utilizada no algoritmo é também um aspecto importante.

O uso econômico da memória disponível é um requisito importante na ordenação interna.

Na ordenação ***in situ*** é denominada à ordenação que executa as permutações (trocas) dos elementos na própria lista.

Ordenação *in situ*

A quantidade extra de memória auxiliar utilizada no algoritmo é também um aspecto importante.

O uso econômico da memória disponível é um requisito importante na ordenação interna.

Na ordenação ***in situ*** é denominada à ordenação que executa as permutações (trocas) dos elementos na própria lista (e.g., selection-, insertion-, bubble-sort)

Lembrando...

Questao 3: Indique o resultado apresentará a execução da seguintes função.

Considere como parâmetros de entrada:

L1=[1,3,4]
L2=[-1,0,2,5,7,9,10].

O que faz a função?

```
def funcao3(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

Questão 3: Indique o resultado apresentará a execução da seguintes função.

Considere como parâmetros de entrada:

L1=[1,3,4]
L2=[-1,0,2,5,7,9,10].

O que faz a função?

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

Questao 3: Indique o resultado apresentará a execução da seguintes função.

L1=[1,3,4]
L2=[-1,0,2,5,7,9,10].

Resposta:
[-1, 0, 1, 2, 3, 4, 5, 7, 9, 10]

O que faz a função?
Intercala listas ordenadas

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

```
L1=[1, 3, 4]
L2=[-1, 0, 2, 5, 7, 9, 10]


n1 = 3
n2 = 7

i = 0
j = 0

L3 = []
```

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[**1**, 3, 4]
L2=[**-1**, 0, 2, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 0
j = 0

L3 = []

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[**1**, 3, 4]
L2=[**-1**, 0, 2, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 0
j = 0

L3 = [-1]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[**1**, 3, 4]
L2=[-1, **0**, 2, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 0
j = 1

L3 = [-1]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[**1**, 3, 4]
L2=[-1, **0**, 2, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 0
j = 1

L3 = [-1]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[**1**, 3, 4]
L2=[-1, **0**, 2, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 0
j = 1

L3 = [-1,0]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[**1**, 3, 4]
L2=[-1, 0, **2**, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 0
j = 2

L3 = [-1,0]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[**1**, 3, 4]
L2=[-1, 0, **2**, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 0
j = 2

L3 = [-1,0]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[**1**, 3, 4]
L2=[-1, 0, **2**, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 0
j = 2

L3 = [-1,0,1]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, **3**, 4]
L2=[-1, 0, **2**, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 1
j = 2

L3 = [-1,0,1]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, **3**, 4]
L2=[-1, 0, **2**, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 1
j = 2

L3 = [-1,0,1]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, **3**, 4]
L2=[-1, 0, **2**, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 1
j = 2

L3 = [-1,0,1,2]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, **3**, 4]
L2=[-1, 0, 2, **5**, 7, 9, 10]

n1 = 3
n2 = 7

i = 1
j = 3

L3 = [-1,0,1,2]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, **3**, 4]
L2=[-1, 0, 2, **5**, 7, 9, 10]

n1 = 3
n2 = 7

i = 1
j = 3

L3 = [-1,0,1,2]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, **3**, 4]
L2=[-1, 0, 2, **5**, 7, 9, 10]

n1 = 3
n2 = 7

i = 1
j = 3

L3 = [-1,0,1,2,3]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, **5**, 7, 9, 10]

n1 = 3
n2 = 7

i = 2
j = 3

L3 = [-1,0,1,2,3]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, **5**, 7, 9, 10]

n1 = 3
n2 = 7

i = 2
j = 3

L3 = [-1,0,1,2,3]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, **5**, 7, 9, 10]

n1 = 3
n2 = 7

i = 2
j = 3

L3 = [-1,0,1,2,3,4]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, **5**, 7, 9, 10]

n1 = 3
n2 = 7

i = 3
j = 3

L3 = [-1,0,1,2,3,4]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, **5**, 7, 9, 10]

n1 = 3
n2 = 7

i = 3
j = 3

L3 = [-1,0,1,2,3,4]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, **5**, 7, 9, 10]

n1 = 3
n2 = 7

i = 3
j = 3

L3 = [-1,0,1,2,3,4]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, **5**, 7, 9, 10]

n1 = 3
n2 = 7

i = 3
j = 3

L3 = [-1,0,1,2,3,4]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, **5**, 7, 9, 10]

n1 = 3
n2 = 7

i = 3
j = 3

L3 = [-1,0,1,2,3,4,5]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, 5, **7**, 9, 10]

n1 = 3
n2 = 7

i = 3
j = 4

L3 = [-1,0,1,2,3,4,5]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, 5, **7**, 9, 10]

n1 = 3
n2 = 7

i = 3
j = 4

L3 = [-1,0,1,2,3,4,5]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, 5, **7**, 9, 10]

n1 = 3
n2 = 7

i = 3
j = 4

L3 = [-1,0,1,2,3,4,5,7]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, 5, 7, **9**, 10]

n1 = 3
n2 = 7

i = 3
j = 5

L3 = [-1,0,1,2,3,4,5,7]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, 5, 7, **9**, 10]

n1 = 3
n2 = 7

i = 3
j = 5

L3 = [-1,0,1,2,3,4,5,7]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, 5, 7, **9**, 10]

n1 = 3
n2 = 7

i = 3
j = 5

L3 = [-1,0,1,2,3,4,5,7,9]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, 5, 7, 9, **10**]

n1 = 3
n2 = 7

i = 3
j = 6

L3 = [-1,0,1,2,3,4,5,7,9]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, 5, 7, 9, **10**]

n1 = 3
n2 = 7

i = 3
j = 6

L3 = [-1,0,1,2,3,4,5,7,9]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, 5, 7, 9, **10**]

n1 = 3
n2 = 7

i = 3
j = 6

L3 = [-1,0,1,2,3,4,5,7,9,10]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 3
j = 7

L3 = [-1,0,1,2,3,4,5,7,9,10]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 3
j = 7

L3 = [-1,0,1,2,3,4,5,7,9,10]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

Semana 06 ...

L1=[1, 3, **4**]
L2=[-1, 0, 2, 5, 7, 9, 10]

n1 = 3
n2 = 7

i = 3
j = 7

L3 = [-1,0,1,2,3,4,5,7,9,10]

```
def intercala(L1, L2):
    n1 = len(L1)
    n2 = len(L2)
    i = 0
    j = 0
    L3 = list([])
    while i<n1 and j<n2:
        if L1[i]<L2[j]:
            L3.append(L1[i])
            i = i+1
        else:
            L3.append(L2[j])
            j = j+1
    while i<n1:
        L3.append(L1[i])
        i = i+1
    while j<n2:
        L3.append(L2[j])
        j = j+1
    return L3
```

D. Merge sort

Proposto por John Von Neumann (1945), é baseado em uma estrategia de resolução de problemas conhecido como: **divisão e conquista**.

• Essa técnica permite decompor um problema, em unidades menores.

• Cada unidade menor deve ser resolvida (em geral de forma recursiva)

• Por fim, utilizar as soluções parciais para obter a solução da instância original.

D. Merge sort

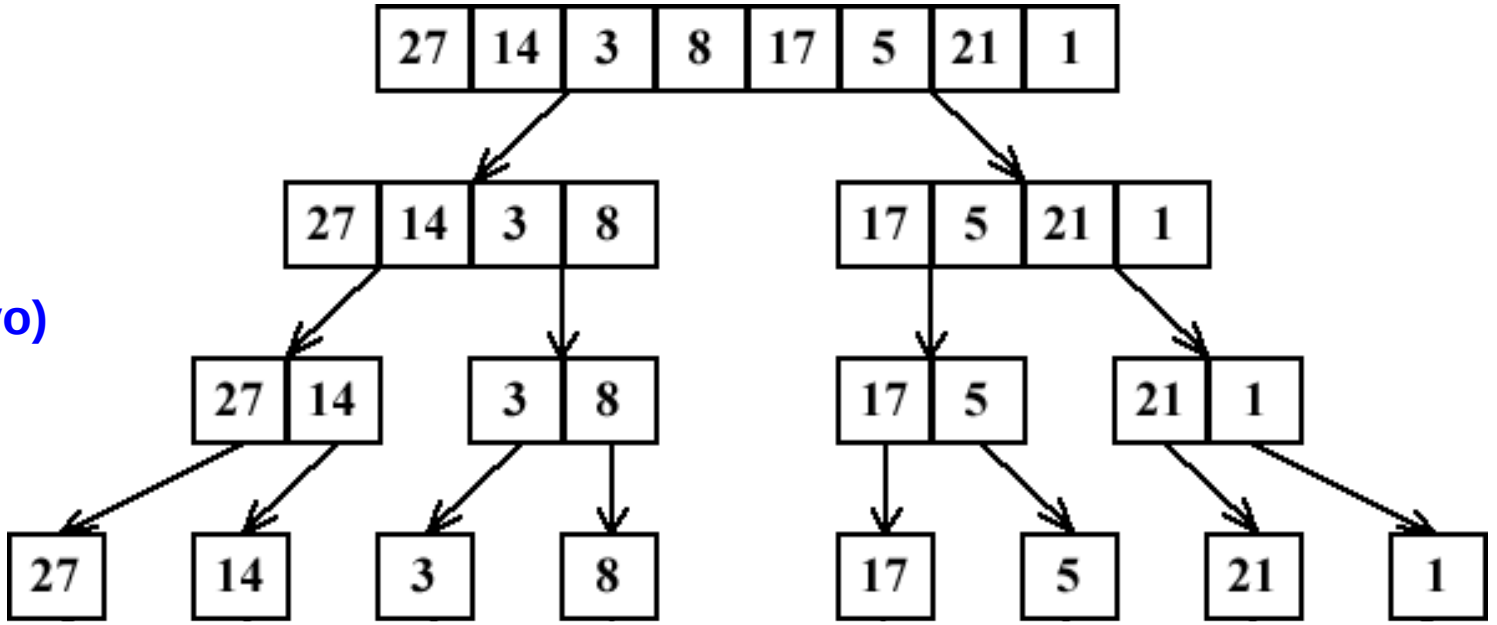
Uma **instância trivial**, seria uma lista de 1 elemento, pois essa lista **já está ordenada**.

O problema de ordenação pode ser dividido recursivamente até chegar em uma **instância trivial**.

No Merge-sort dividimos a lista em 2 metades, essas metades são ordenadas recursivamente.

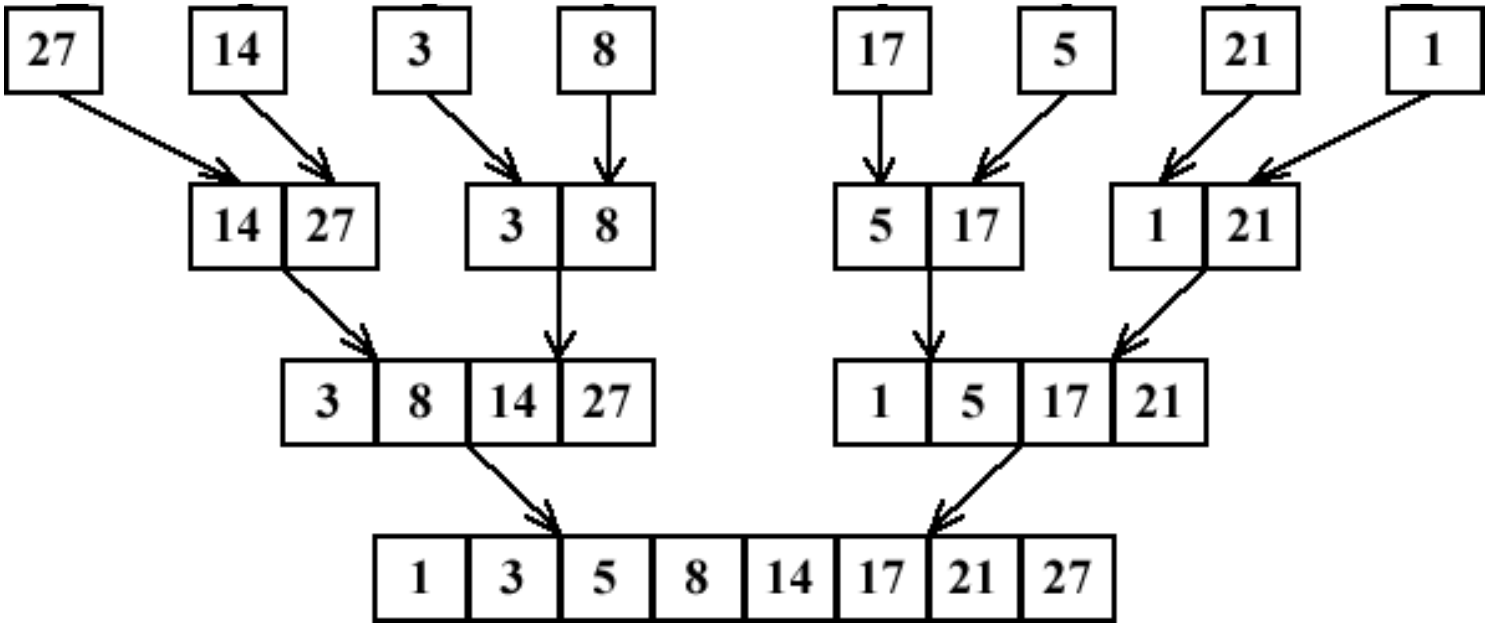
D. Merge sort

Divisão
(processo recursivo)



D. Merge sort

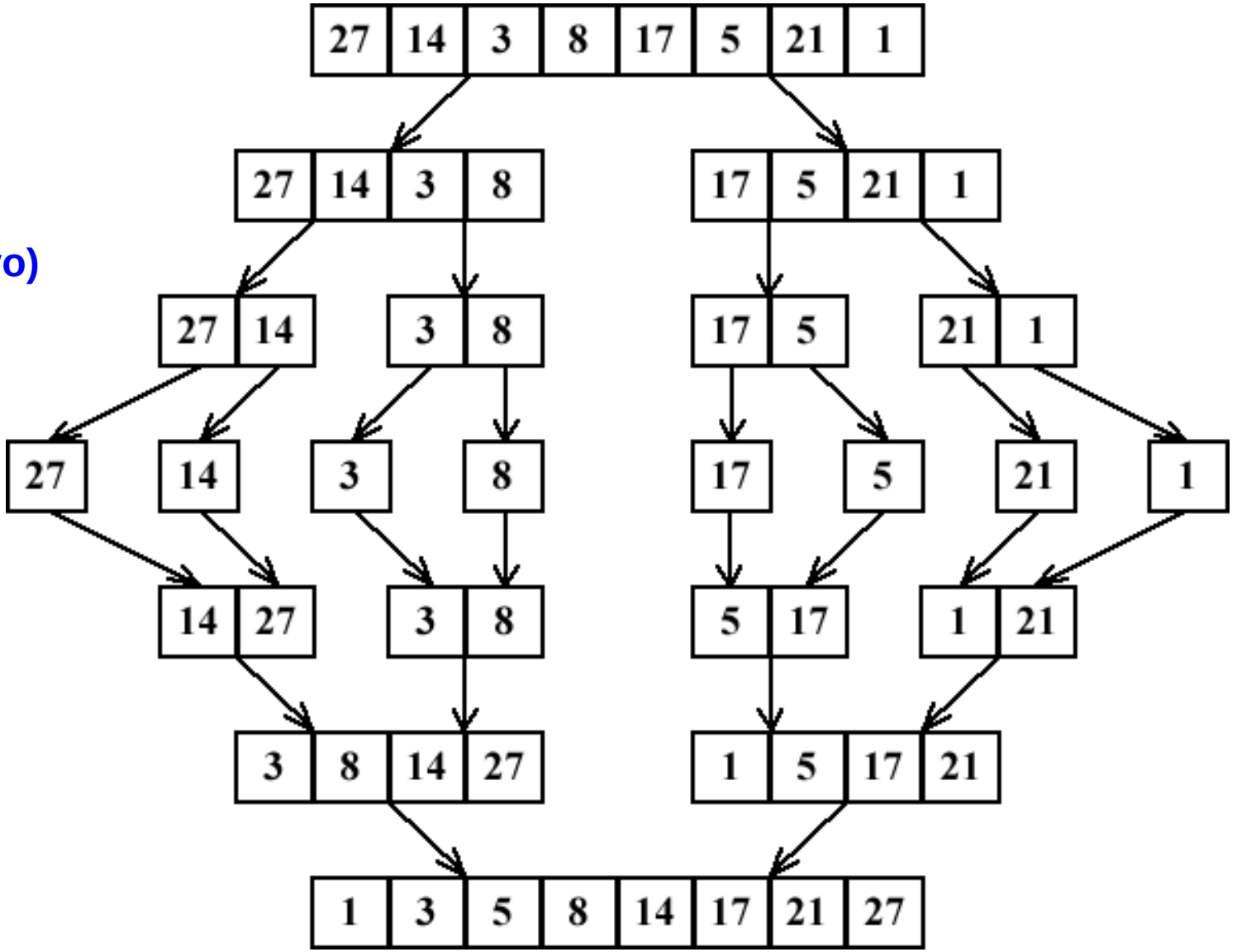
Conquista (intercala)



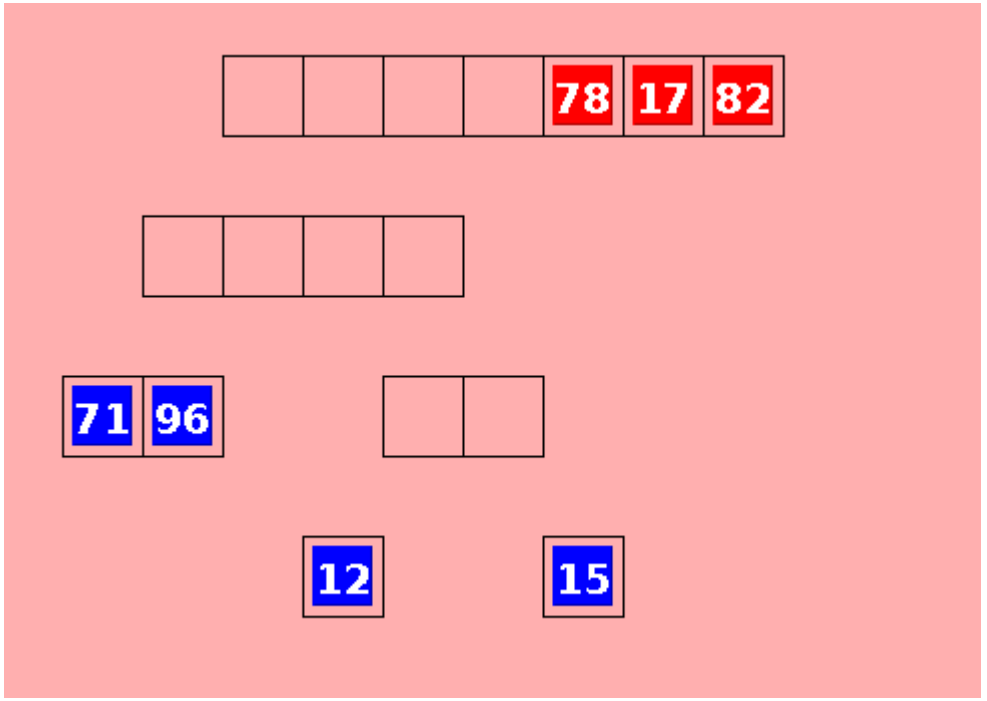
D. Merge sort

Divisão
(processo recursivo)

Conquista
(intercala)



D. Merge sort



Simulação:
http://www.cse.iitk.ac.in/users/dsrkg/cs210/applets/sortingII/mergeSort/mergeSort.html
http://upload.wikimedia.org/wikipedia/commons/c/cc/Merge-sort-example-300px.gif

Obs(*): Teste com par é impar. Como é feita a subdivisão?

D. Merge sort

Crie uma função **merge_sort** que permita ordenar de forma ascendente uma lista de n elementos.

Cabeçalho: def merge_sort(L):

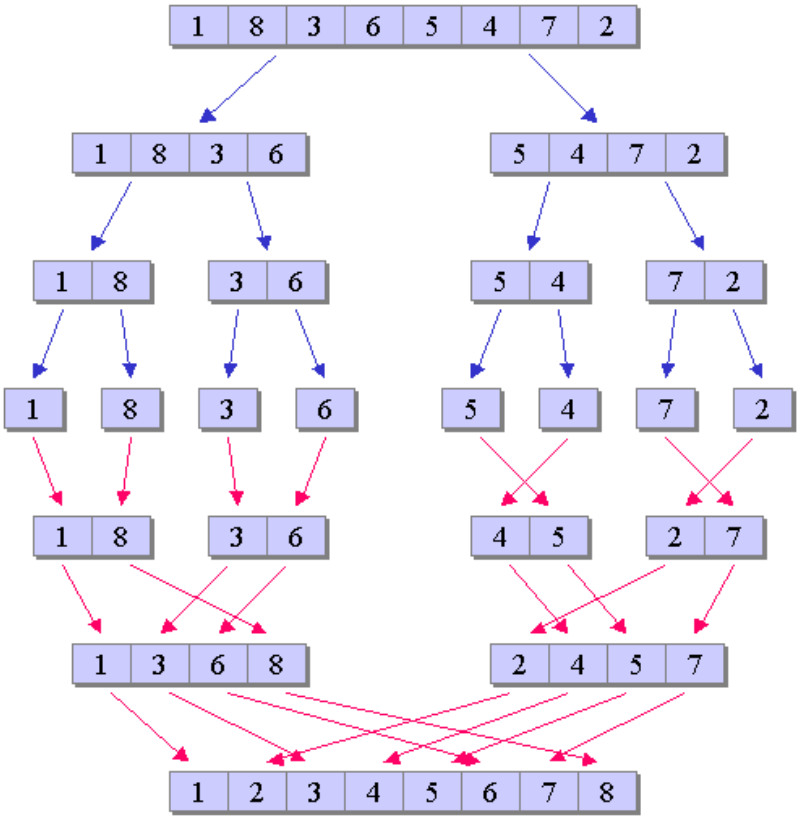
Dica: Use recursividade e a função intercala(L1,L2)

D. Merge sort

```
def merge_sort(L):
    if len(L) <= 1:
        return L
    meio = len(L)/2
    L1 = merge_sort(L[:meio])
    L2 = merge_sort(L[meio:])
    return intercala(L1, L2)
```

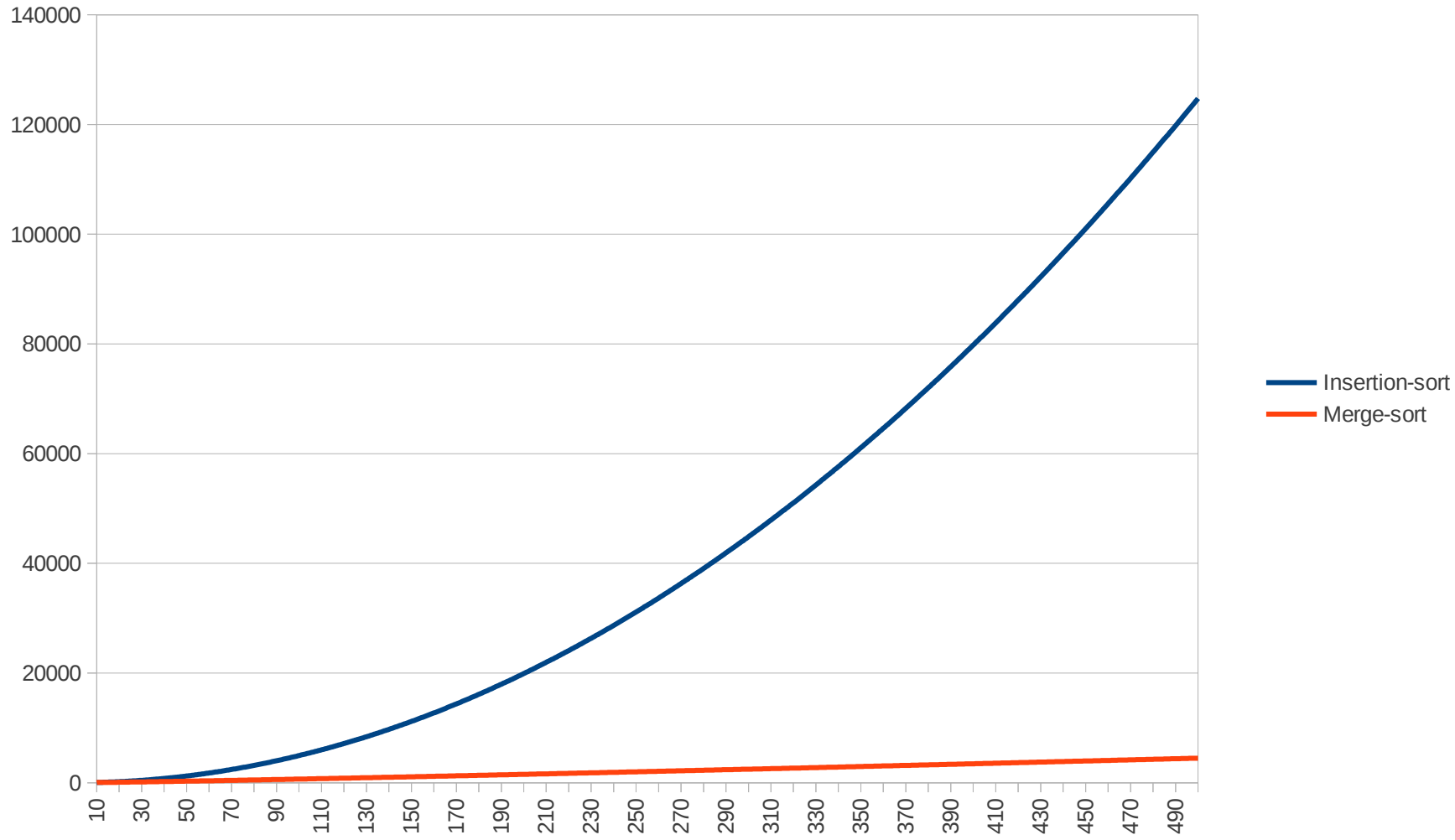
D. Merge sort (tempo)

```
def merge_sort(L):
    if len(L) <= 1:
        return L
    meio = len(L)/2
    L1 = merge_sort(L[:meio])
    L2 = merge_sort(L[meio:])
    return intercala(L1, L2)
```



$T(n) = T(n/2) + T(n/2) + n$
$T(n) = n \lg (n)$

D. Merge sort (tempo)



Comparando algoritmos