



ESTRUTURAS DE DADOS E ALGORITMOS

ALGORITMOS DE ORDENAÇÃO POR COMPARAÇÃO - II

Adalberto Cajueiro

Departamento de Sistemas e Computação

Universidade Federal de Campina Grande



ALGORITMOS VISTOS ANTERIORMENTE

- Focados em uma varredura linear da entrada
- Melhor caso: $O(n)$
 - Não considerado na prática
- Pior caso: $O(n^2)$

DIVIDIR PARA CONQUISTAR

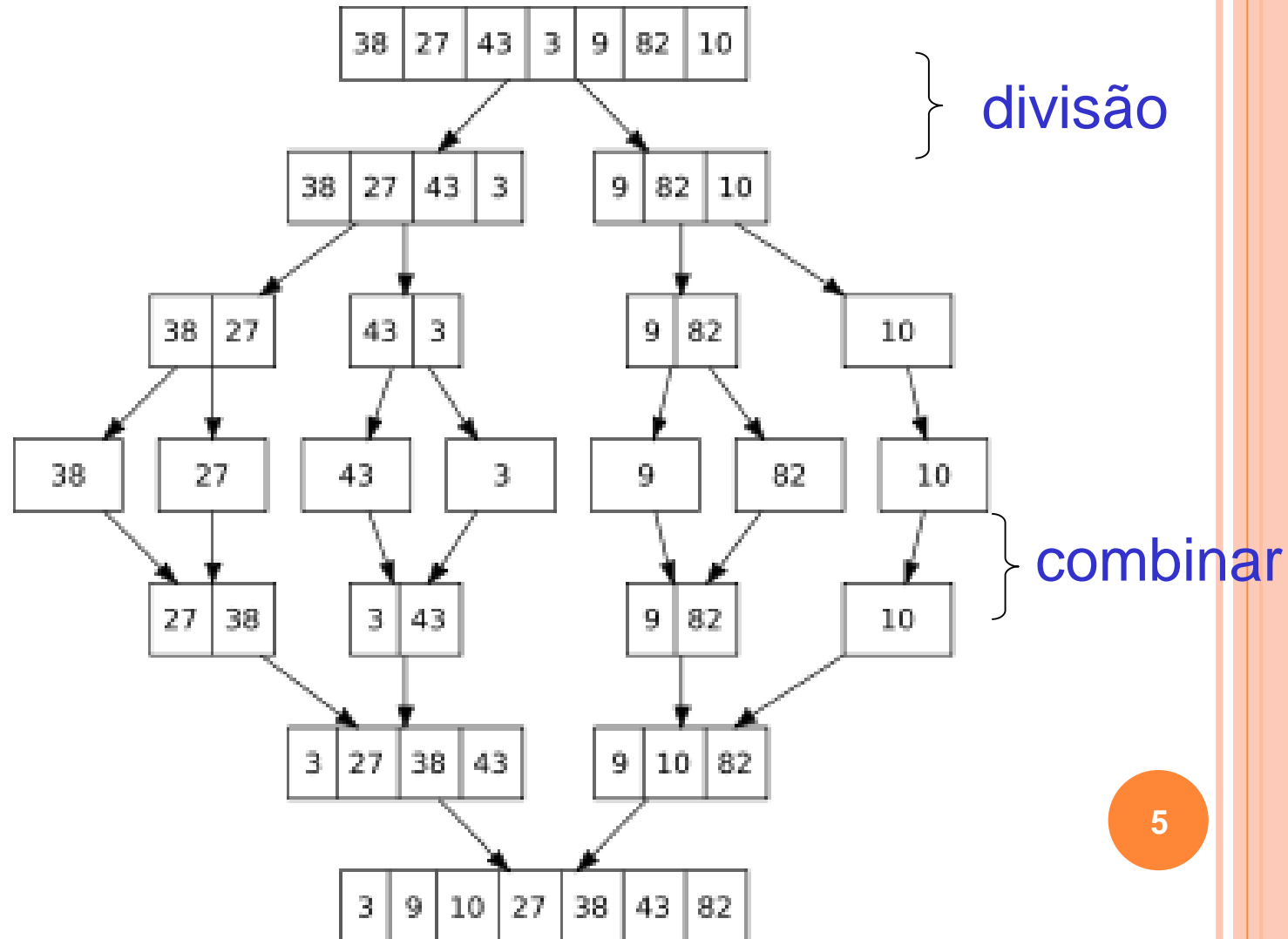
- Algoritmos focados em dividir o problema em problemas menores
- Resolver cada problema menor
- Combinar as respostas menores para obter a resposta maior
- Conseguem performance perto da ótima da ordenação por comparação

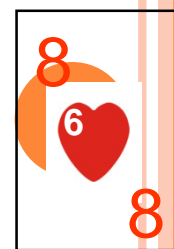
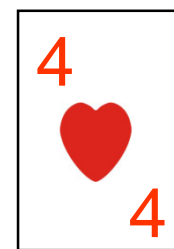
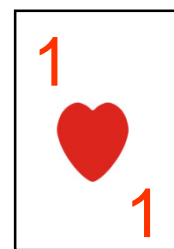
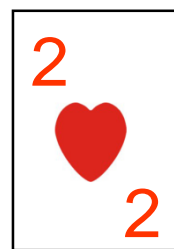
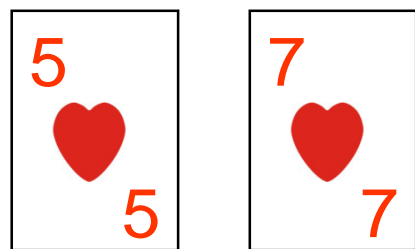
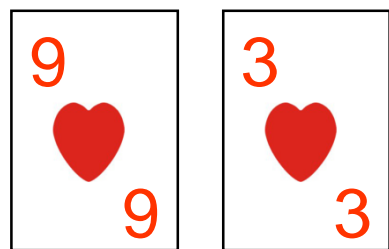
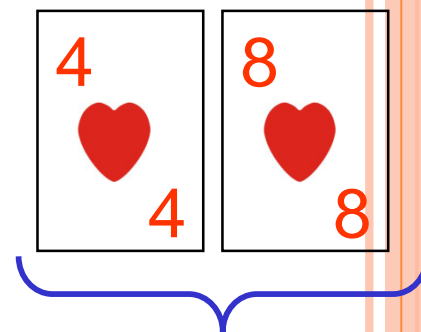
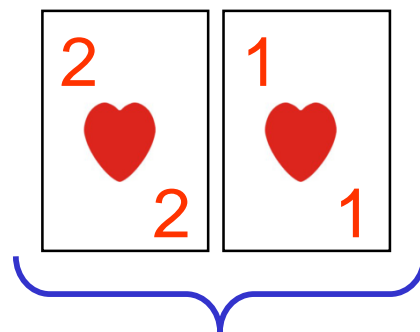
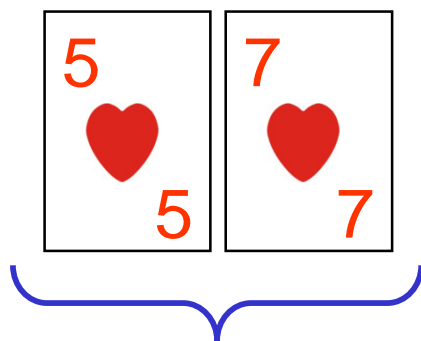
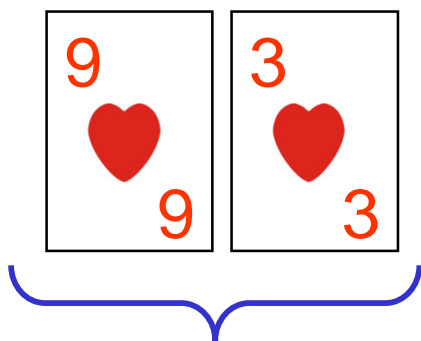
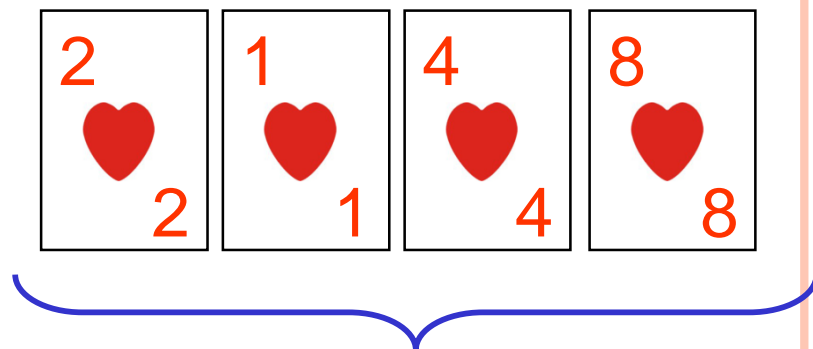
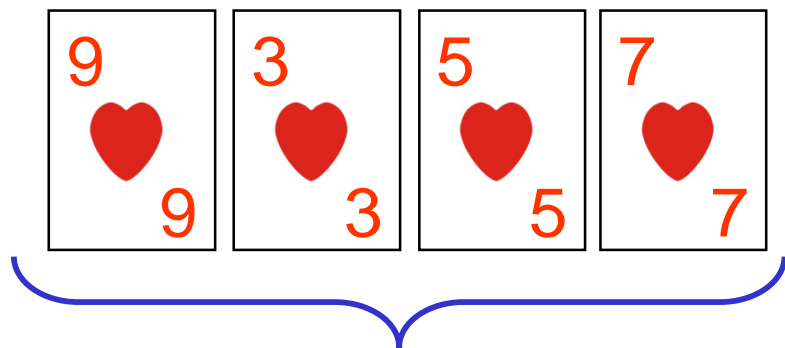
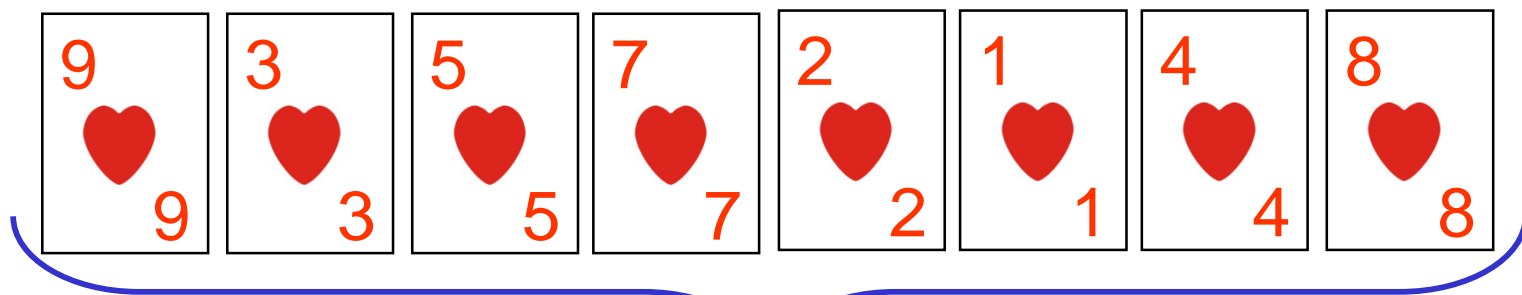
MERGE SORT

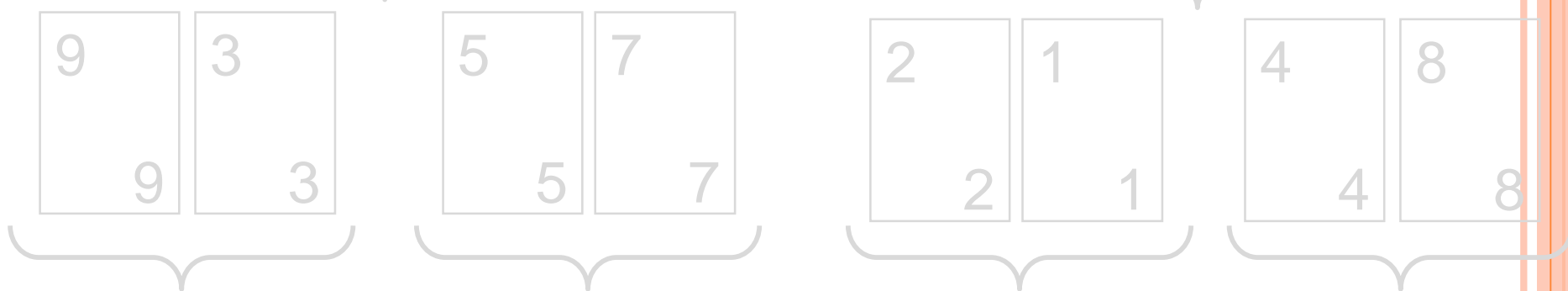
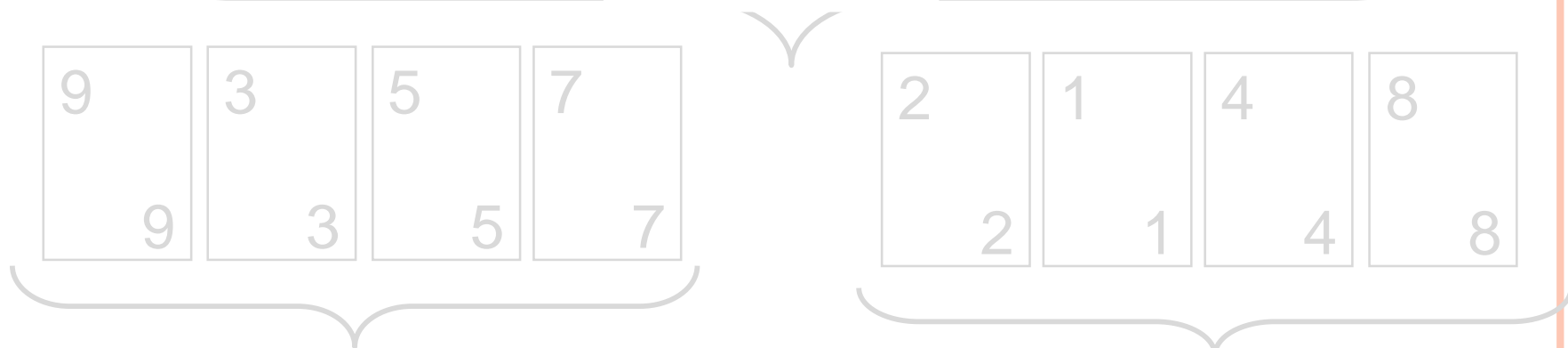
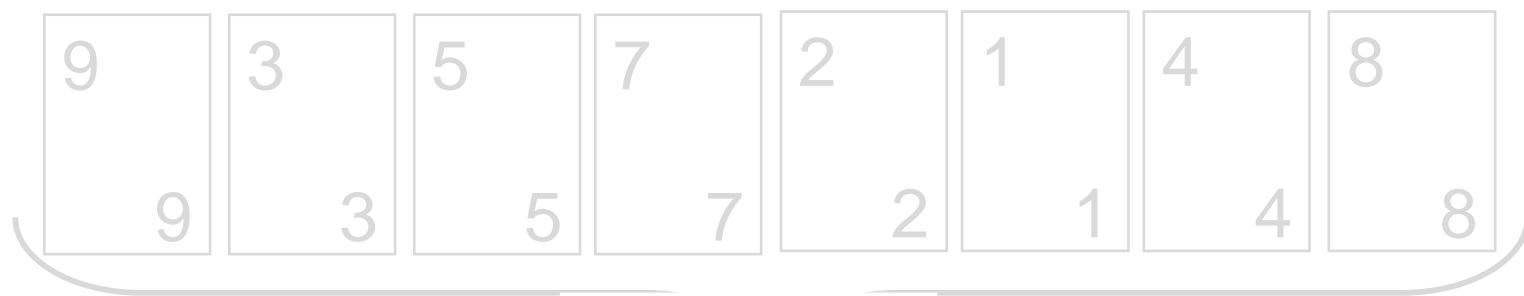
- Algoritmo do tipo **dividir para conquistar**
 - **Dividir**: se $\#sequ\tilde{e}ncia > 1$, divida em duas menores
 - **Conquistar**: ordene cada subsequ\~encia recursivamente
 - **Combinar**: junte as duas subsequ\~encias em uma seq\~encia ordenada
- Inventor: Neumann (1945)

MERGE SORT

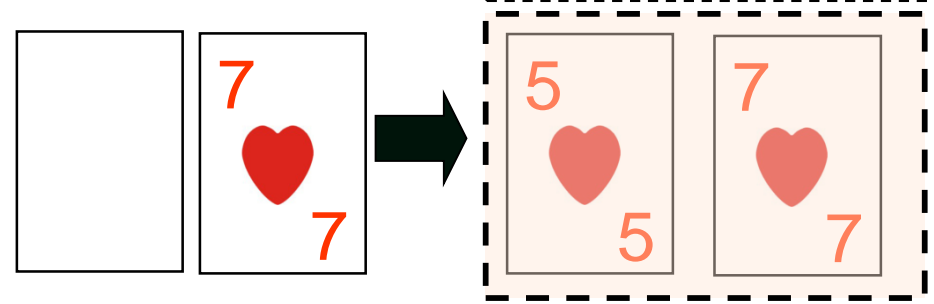
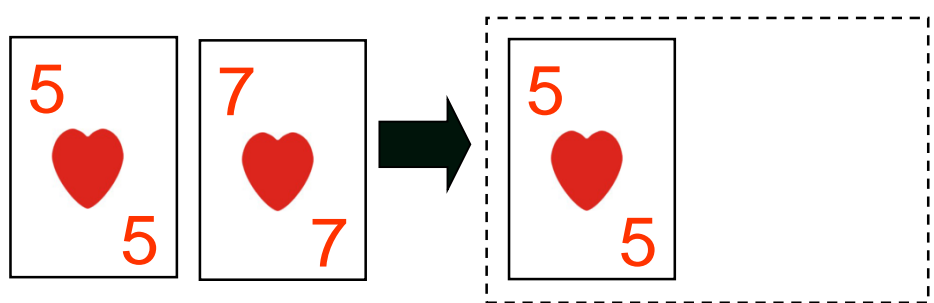
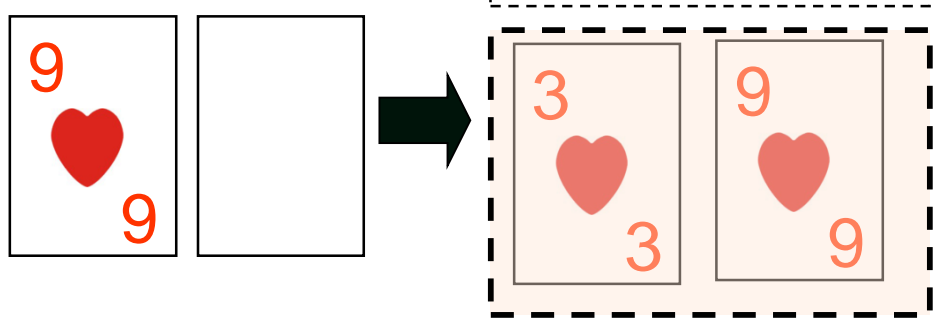
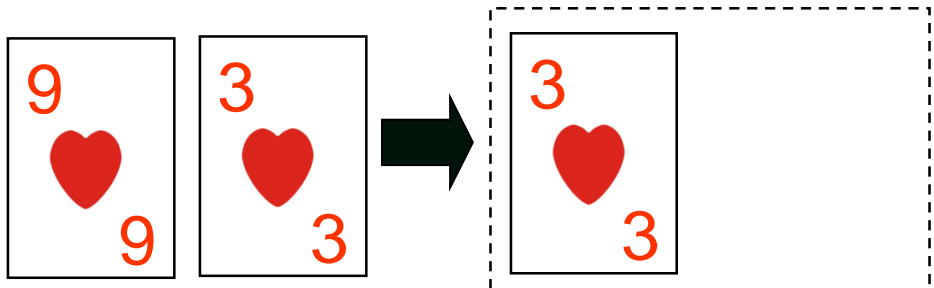
- Exemplo:

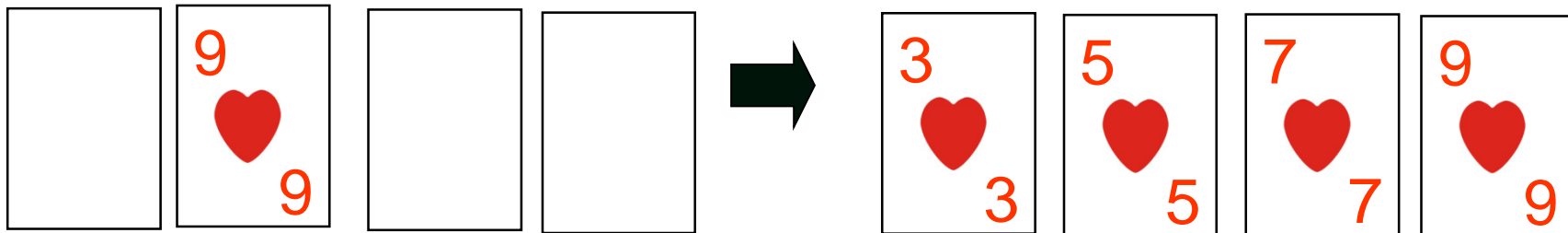
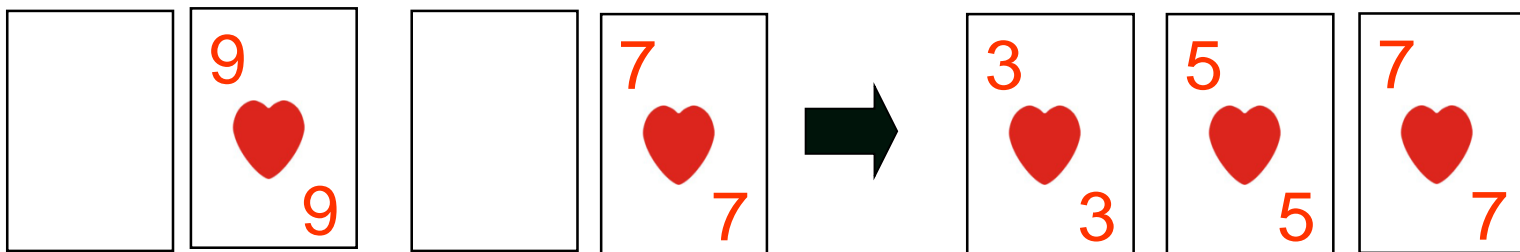
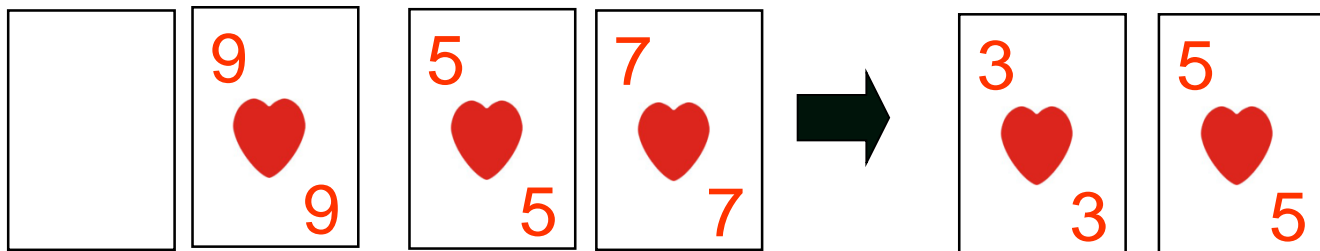
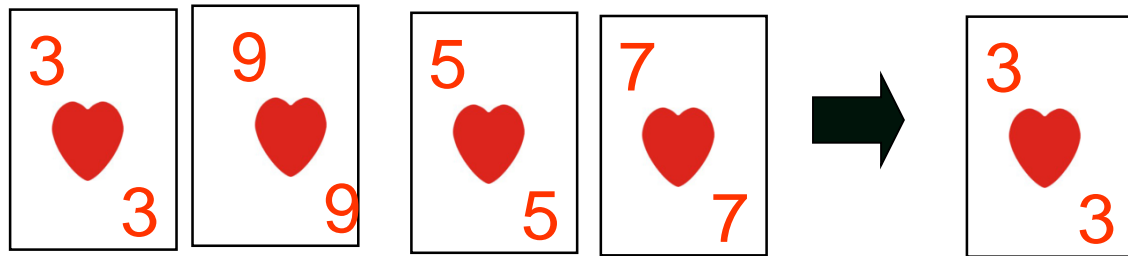


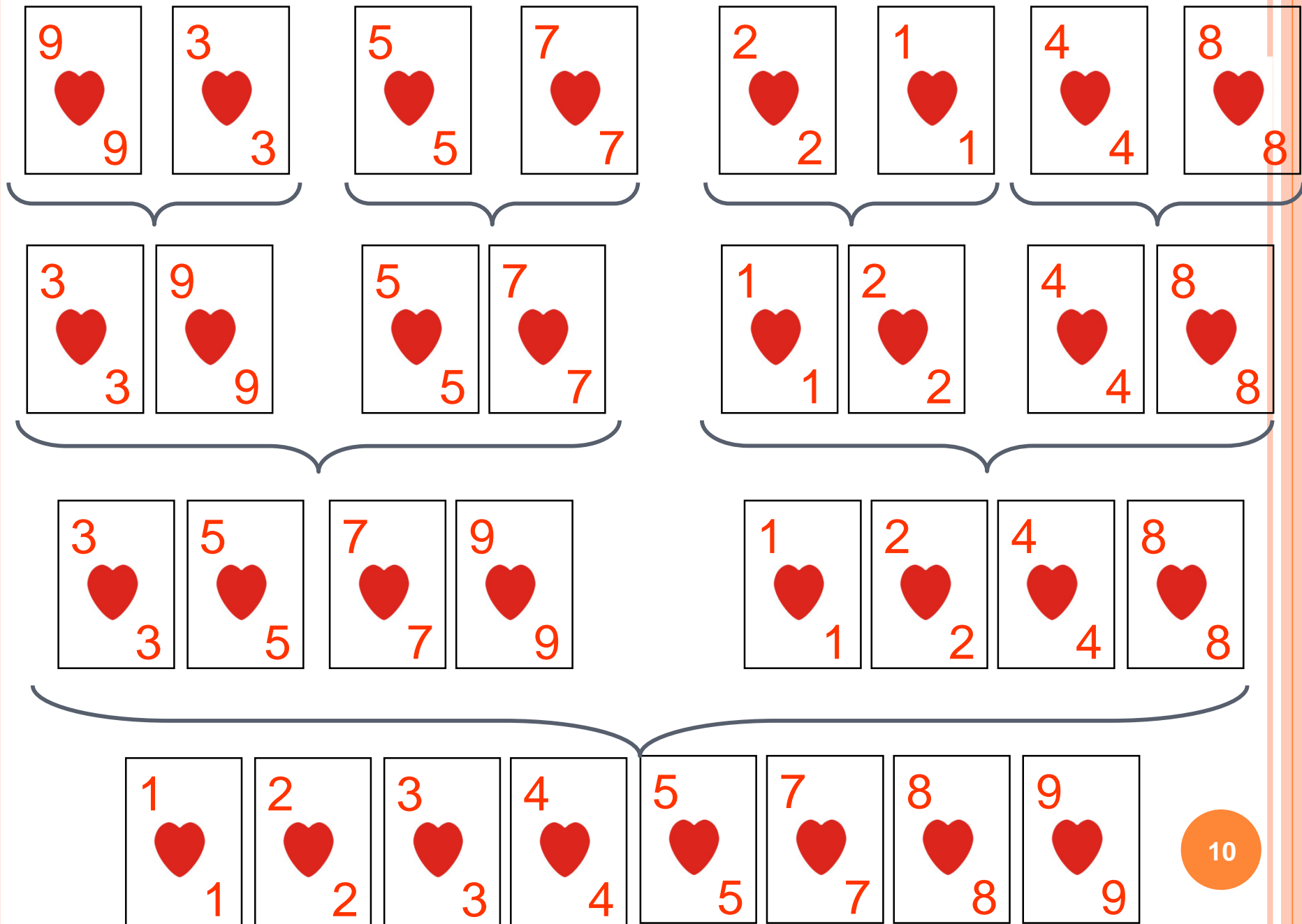




Divisão!!!







MERGE SORT

- Exercício:
 - Ordene as listas a seguir utilizando o merge sort
 - 21, 23, 2, 34, 245, 33, 66

MERGE SORT

- Como seria o algoritmo do Merge Sort?

```
MERGE-SORT(A)
```

```
  if  $n = 1$  done
```

```
  MERGE-SORT( $A[1.. \lceil n/2 \rceil]$ )
```

```
  MERGE-SORT( $A[\lceil n/2 \rceil + 1..n]$ )
```

```
  “Merge” the 2 sorted lists
```

MERGE SORT

- Como seria fazer o Merge de duas listas?

```
List merge(List left, List right) {  
    List result = new LinkedList<int>();  
    while (left.length() > 0 and right.length() > 0) {  
        if left.first() ≤ right.first() {  
            result.add(left.cabeca());  
            left = left.remove(left.first()); }  
        else {  
            result.add(right.cabeca());  
            right = right.remove(right.first()); }  
    }  
    if (left.length() > 0)  
        result.append(left);  
    if (right.length() > 0)  
        result.append(right);  
    return result; }
```

MERGE SORT

- Qual seria o tempo do Merge?

```
List merge(List left, List right) {  
    List result = new LinkedList<int>();  
    while (left.length() > 0 and right.length() > 0) {  
        if left.first() ≤ right.first() {  
            result.add(left.cabeca());  
            left = left.remove(left.first());  
        }  
        else {  
            result.add(right.cabeca());  
            right = right.remove(right.first());  
        }  
    }  
    if (left.length() > 0)  
        result.append(left);  
    if (right.length() > 0)  
        result.append(right);  
    return result;  
}
```

MERGE SORT

- Qual o tempo do Merge Sort?

MERGE-SORT(A)

if $n = 1$ done

MERGE-SORT($A[1.. \lceil n/2 \rceil]$)

MERGE-SORT($A[\lceil n/2 \rceil + 1..n]$)

“Merge” the 2 sorted lists

$\Theta(1)$

$T(\lceil n/2 \rceil)$

$T(\lceil n/2 \rceil)$

$\Theta(n)$

$T(n) = 1$, se $n=1$

$T(n) = 2T(n/2) + \Theta(n)$, se $n>1$

MERGE SORT - EXERCÍCIO

- Utilize o **método iterativo** e o **método mestre** para encontrar o limite assintoticamente restrito Θ .

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

$$f(n) < n^{\log_b a} \longrightarrow \text{Caso 1} \longrightarrow$$

$$T(n) = \Theta(n^{\log_b a})$$

$$f(n) = n^{\log_b a} \longrightarrow \text{Caso 2} \longrightarrow$$

$$T(n) = \Theta(n^{\log_b a} \lg n) \\ = \Theta(f(n) \cdot \lg n)$$

$$f(n) > n^{\log_b a} \longrightarrow \text{Caso 3} \longrightarrow$$

$$T(n) = \Theta(f(n))$$

CARACTERÍSTICAS

- Boa performance
 - $O(n \cdot \log n)$
- Não é In place
 - Precisa de espaço extra de $O(n)$
- Aplicações
 - Java
 - `Arrays.sort()`, depende da estrutura de dado

MERGE SORT - EXERCÍCIO

- Quem é melhor: bubble, selection, insertion ou merge sort?

The slide features a light beige background with several vertical stripes of varying widths and shades of orange and tan on the left side. A cluster of orange circles of different sizes is positioned on the left, with the number '19' inside one of them. The title 'QUICK SORT' is written in a bold, dark blue font.

QUICK SORT

19

A single orange circle is located in the bottom right corner of the slide.

QUICK SORT

- Dividir para Conquistar
- Artigo
 - <http://portal.acm.org/citation.cfm?id=SERIES11430.63445>

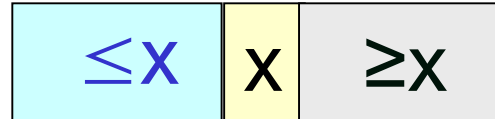


Tony Hoare

QUICK SORT - INTUIÇÃO

- Dividir

- Pivot: x



- Conquistar

- Ordenar cada subarray recursivamente

- Combinar

- Trivial

IDÉIA DO QUICK SORT

- Se a sequência possuir mais de 1 elemento, escolha um elemento da lista, denominado **pivô**;
- **Dividir**:
 - Rearranje a lista de forma que todos os elementos anteriores ao pivô sejam menores ou iguais a ele, e todos os elementos posteriores ao pivô sejam maiores ou iguais a ele. Ao fim do processo o pivô estará em sua posição final.
- **Conquistar**
 - Recursivamente ordene a sublista dos elementos menores e a sublista dos elementos maiores;
- **Combinar**
 - Junte as listas ordenadas e o pivot

QUICK SORT

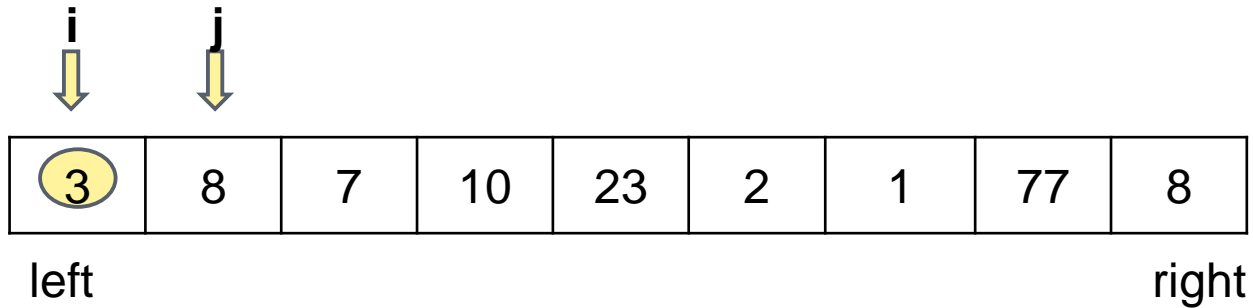
- Exercício:
 - Ordene as listas a seguir utilizando o quick sort
 - 21, 23, 2, 34, 245, 33, 66

QUICK SORT

- Como seria o algoritmo do Quick Sort?

```
void quicksort (int[] a, int l, int r) {  
    if (l >= r) return;  
    int m = partition(a, l, r);  
    quicksort(a, l, m-1);  
    quicksort(a, m+1, r);  
}
```

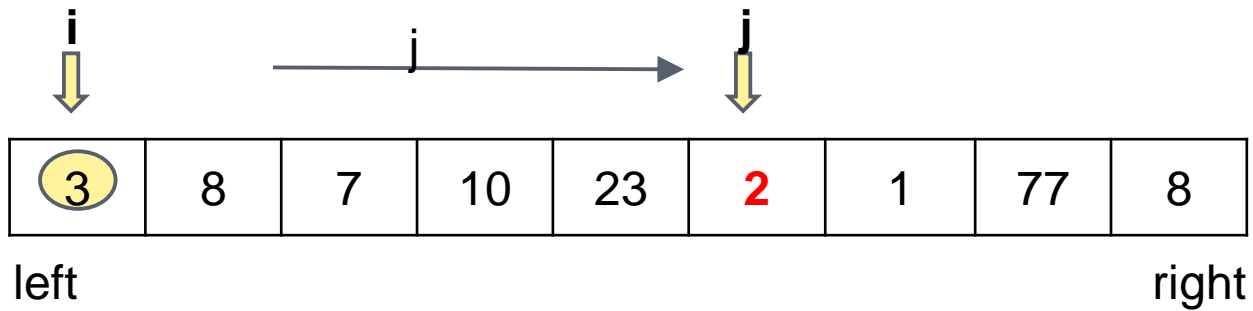

QUICK SORT: PARTITION



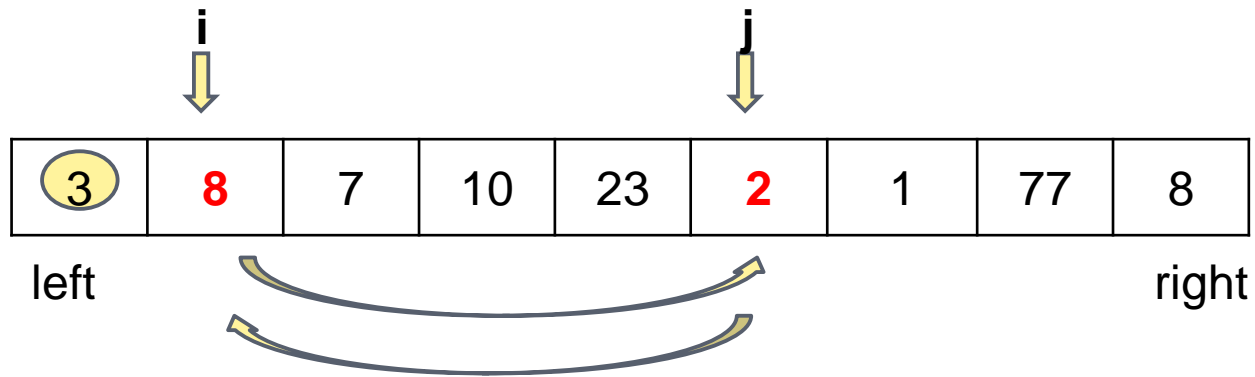
Início do particionamento.



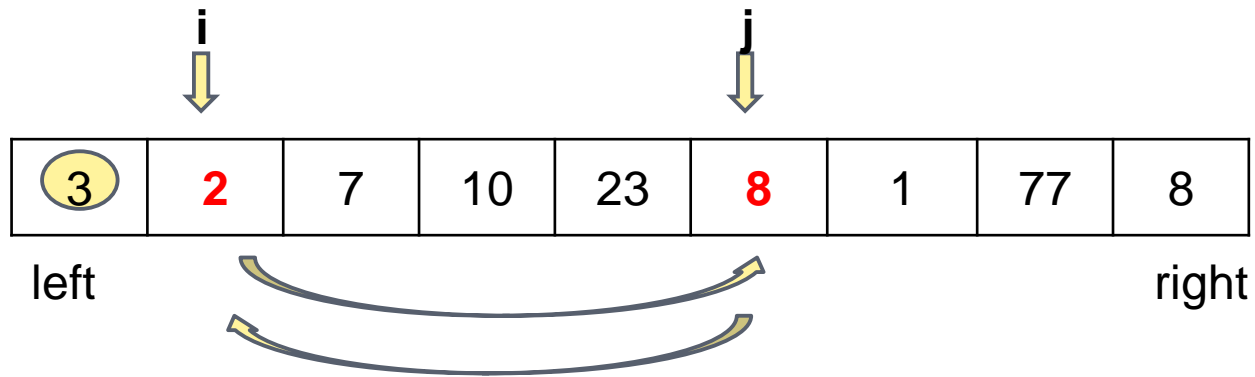
QUICK SORT: PARTITION



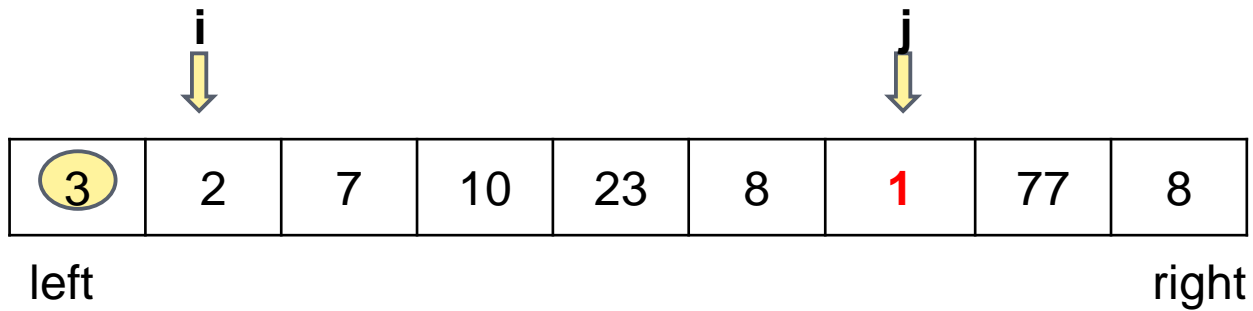
QUICK SORT: PARTITION



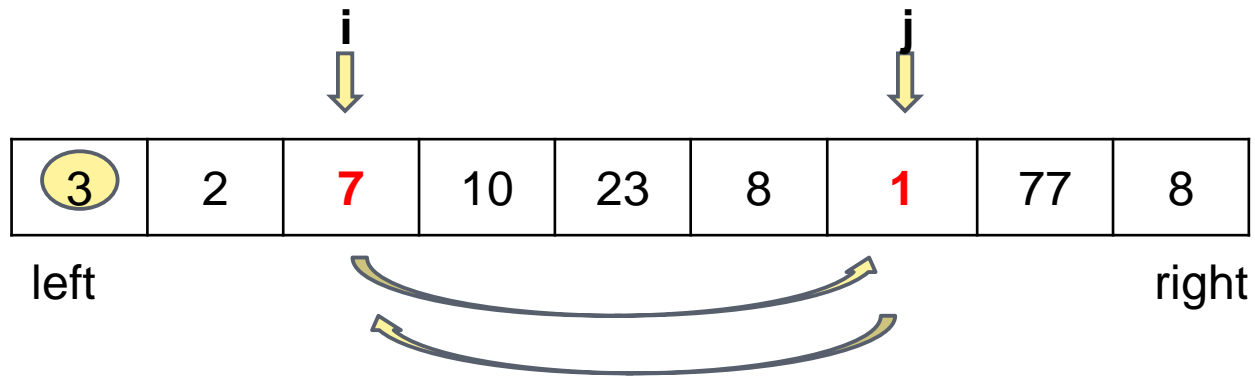
QUICK SORT: PARTITION



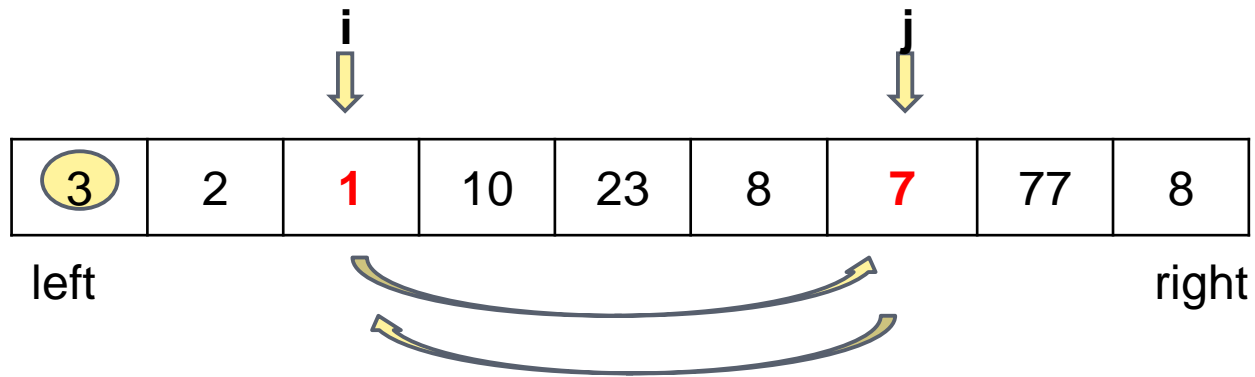
QUICK SORT: PARTITION



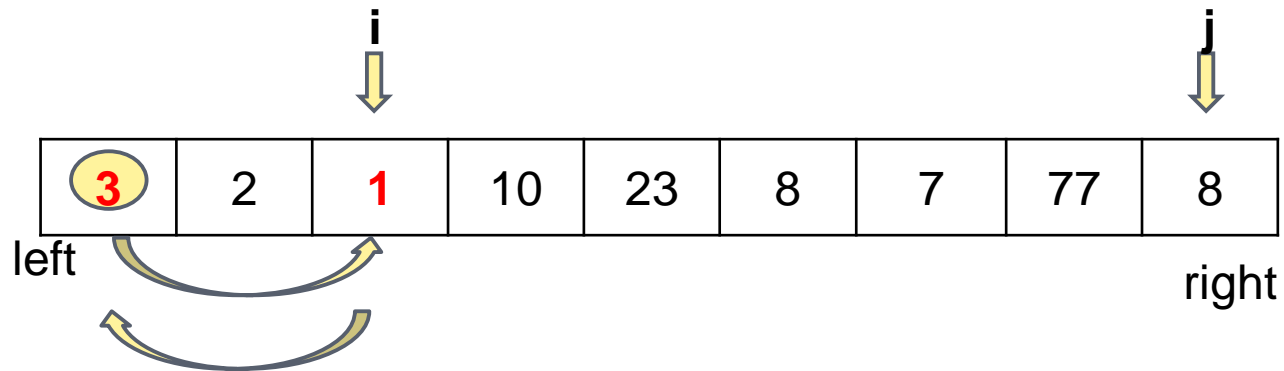
QUICK SORT: PARTITION



QUICK SORT: PARTITION



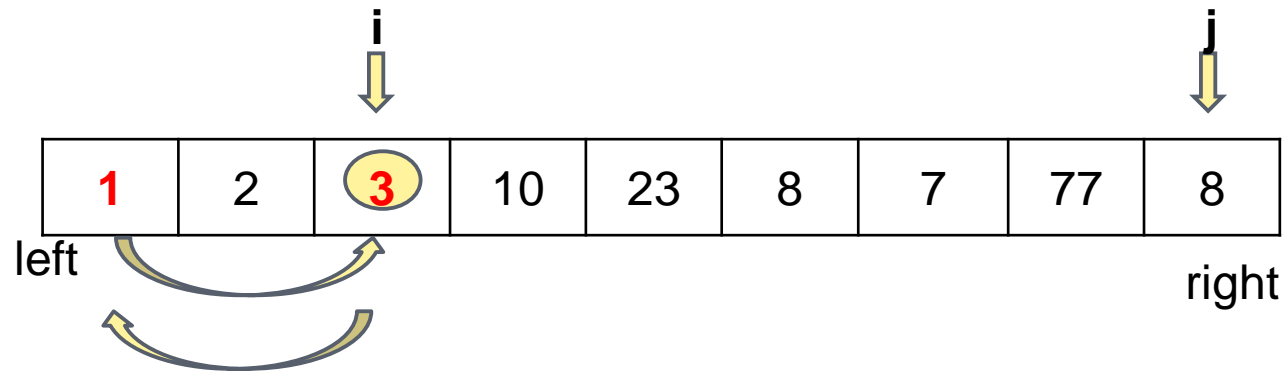
QUICK SORT: PARTITION



Fim da iteração ($j == \text{right}$)
Trocar left por i.



QUICK SORT: PARTITION



QUICK SORT: PARTITION

1	2	3	10	23	8	7	77	8
left			right					

Fim do particionamento

○ Respeita a invariante?



QUICK SORT: PARTITION

Próximo particionamento

1	2	3	10	23	8	7	77	8
left	right	left					right	

...

quickSort(v, left, pivot-1)

quickSort(v, pivot + 1, right)

...



QUICK SORT: PARTITION

Próximo particionamento

10	23	8	7	77	8
----	----	---	---	----	---

left

right



QUICK SORT: ALGORITMO DE PARTICIONAMENTO

```
partition(v, left, right)

    pivot = v[left]
    # position to place the pivot
    i = left
    for j = left+ 1 to right{
        if (v[j] <= pivot){
            i += 1
            v[i],v[j] = v[j],v[i]
        }
    }
    v[left],v[i] = v[i],v[left]

    return i
```



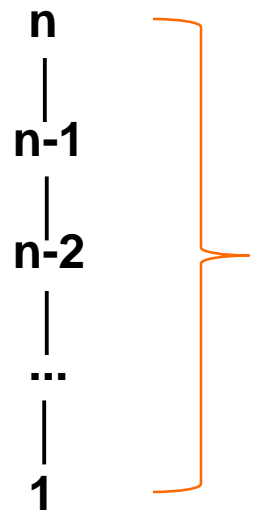
CARACTERÍSTICAS

- Boa performance
- In place
 - Gasto com a recursão (média de $O(\log n)$)
- Não é Stable
- Muito usado na prática
- Aplicações
 - Java
 - `Arrays.sort()`, depende da estrutura de dado

QUICK SORT

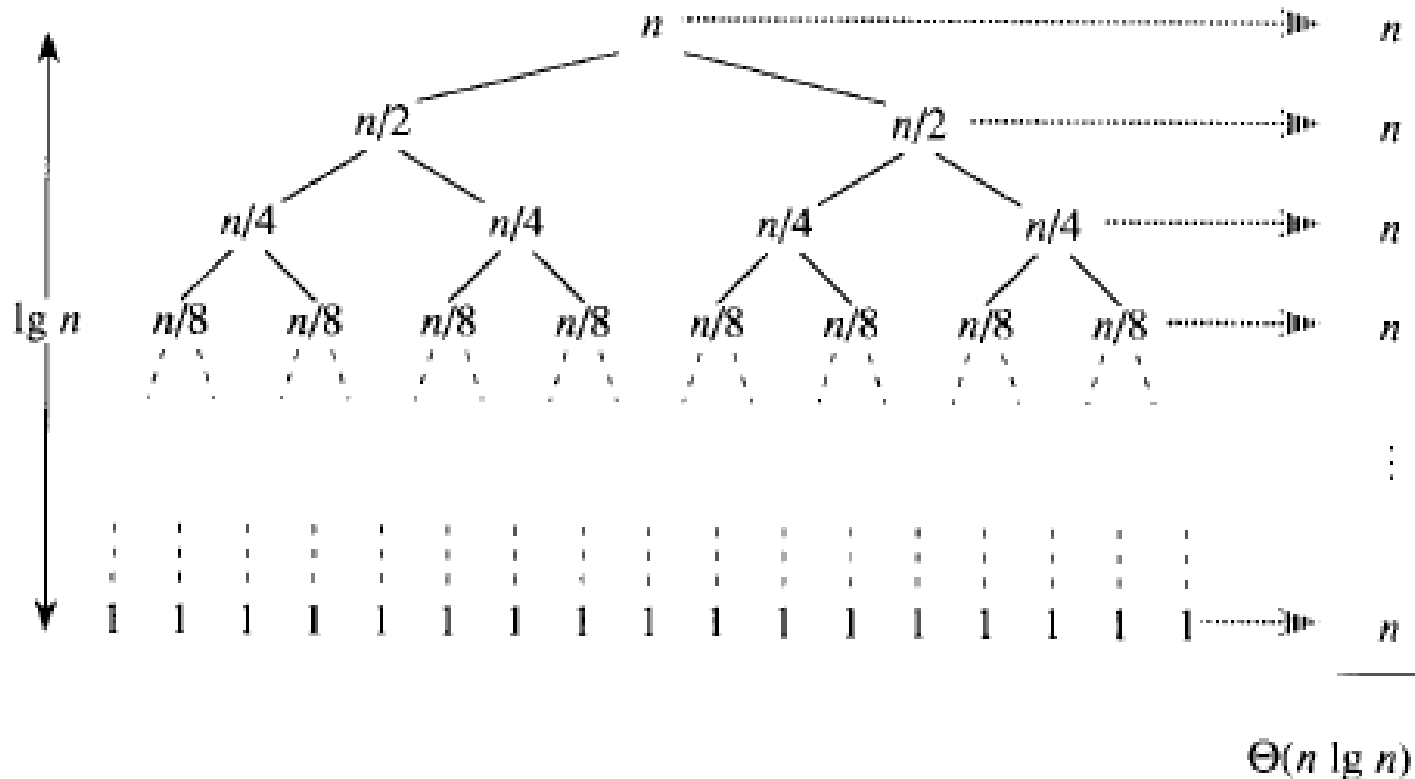
- Análise do Quick sort (pior caso)
 - Particionamento produz regiões com 1 e n-1 elementos
 - Assumir que isso ocorre a cada particionamento
 - Relação de recorrência

$$T(n) = T(n - 1) + \Theta(n)$$


$$\sum_{i=1}^n i = \frac{n(1 + n)}{2} = \theta(n^2)$$

QUICK SORT

- Análise do Quick sort (melhor caso)
 - Particionamento produz regiões com mesmo tamanho
 - Relação de recorrência $T(n) = 2T(n/2) + \Theta(n)$



QUICK SORT

- **Exercício:** qual o tempo de execução do quick sort quando o particionamento divide a entrada em proporções de 10% e 90%?
- Versao randomizada
 - Escolha aleatória de um bom pivot e permutacao da entrada para deixa-la mais próxima do melhor caso.
- Visualização
 - <http://www.aarondufour.com/tools/visualizer/>
 - <http://www.sorting-algorithms.com/>
 - <http://sortvis.org/visualisations.html>
 - <http://bost.ocks.org/mike/algorithms/>

QUESTÕES DE IMPLEMENTAÇÃO

- Adaptando o quicksort para trabalhar com um tipo genérico

QUESTÕES DE IMPLEMENTAÇÃO

○ Exemplo: quicksort

```
int partition(int[] a, int l, int r) {  
    int i=l+1;  
    int j=r;  
    int p = a[l];  
    while (i<=j) {  
        if (a[i]<=p) i++;  
        else if (a[j]>p) j--;  
        else a = swap(a,i,j);  
    }  
    a = swap(a,l,j);  
    return j; //posição do pivot  
}
```

```
void quicksort (int[] a, int l, int r) {  
    if (l>=r) return;  
    int m = partition(a, l, r);  
    quicksort(a,l,m-1);  
    quicksort(a,m+1,r);  
}
```

QUESTÕES DE IMPLEMENTAÇÃO

- Exemplo: quicksort

```
int partition(T[] a, int l, int r) {  
    int i=l+1;  
    int j=r;  
    T p = a[l];  
    while (i<=j) {  
        if (a[i].compareTo(p)<=0) i++;  
        else if (a[j].compareTo(p)>0) j--;  
        else a = swap(a,i,j);  
    }  
    a = swap(a,l,j);  
    return j; //posição do pivot  
}
```

```
void quicksort (T[] a, int l, int r) {  
    if (l>=r) return;  
    int m = partition(a, l, r);  
    quicksort(a,l,m-1);  
    quicksort(a,m+1,r);  
}
```

REFERÊNCIAS

- Capítulo 8
 - Quick Sort

