# Algoritmos de Ordenação

## Algoritmos de ordenação

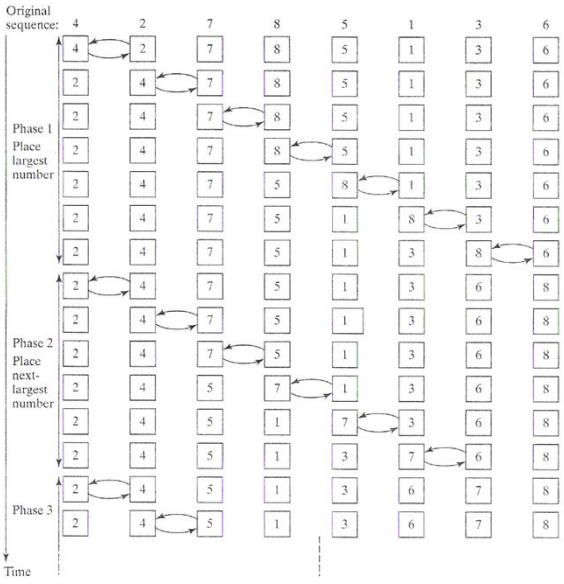
- Algoritmos de "comparação-e-troca"
  - Bubble Sort
  - Merge Sort
  - Quick Sort

## BUBBLE SORT

- Usa a estratégia de "comparação-e-troca"
- É constituído por várias fases
- Cada fase tem várias iterações
- Cada iteração consiste em comparar dois valores, e se necessário trocá-los
- A título de exemplo, vamos considerar um vector x, constituído por valores $x_0$, $x_1$, $x_2$, …, $x_n$ não ordenados
- Veja o vídeo em http://www.youtube.com/watch?v=P00xJgWzz2c

## BUBBLE SORT SEQUENCIAL

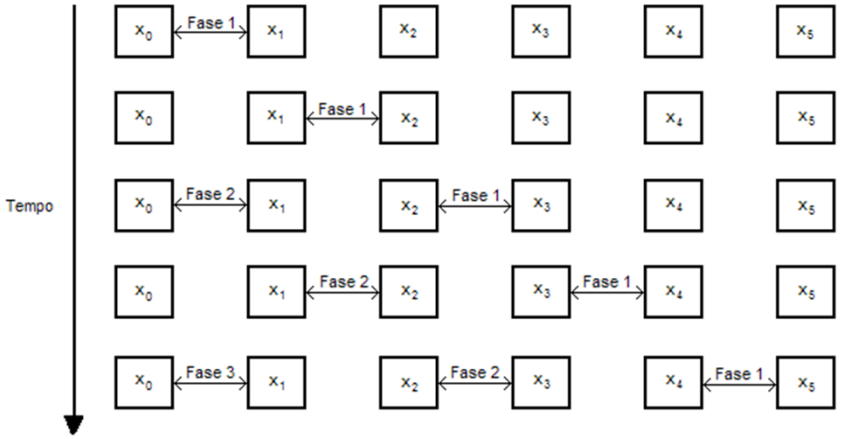
- Considerando uma ordenação crescente
- O elemento de maior valor vai para o final do vector
- Em cada fase que é realizada, o número de iterações diminui em uma unidade
- Só existe troca na iteração caso o elemento da esquerda seja superior ao da direita
- Na pior das hipóteses vão existir n-1 fases



## BUBBLE SORT COM *PIPELINING*

- O *pipelining* é possivel pois as iterações de cada fase vão percorrendo o vetor, não necessitando posteriormente de valores anteriores
- Ou seja, quando a primeira fase já se encontra na comparação de $x_2$ e $x_3$, a segunda fase já pode usar os valores $x_0$ e $x_1$ para comparação

## BUBBLE SORT COM *PIPELINING*

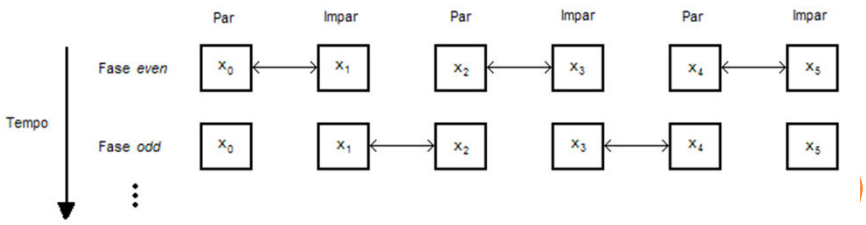


## ODD-EVEN SORT

- É uma variante do algoritmo **Bubble Sort**
- Constituído por duas fases distintas:
  - Fase *even* (par)
  - Fase *odd* (ímpar)
- Na fase *even* (par) comparam-se as posições (índices) pares com a posição seguinte a cada uma delas
- Na fase *odd* (ímpar) comparam-se as posições (índices) ímpares com a posição seguinte a cada uma delas

## ODD-EVEN SORT PARALELO

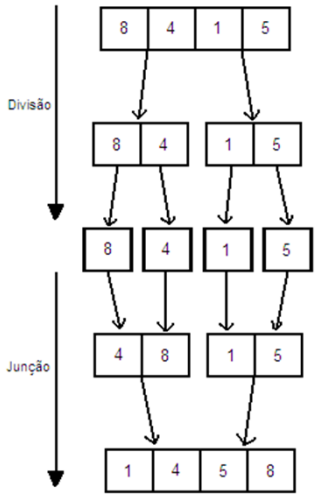
- Esta variante, realizada de maneira sequencial não traz qualquer benefício em relação ao Bubble Sort (sequencial)
- Mas, esta variante realizada em paralelo, apresenta notoriamente grande potencial



## MERGE SORT

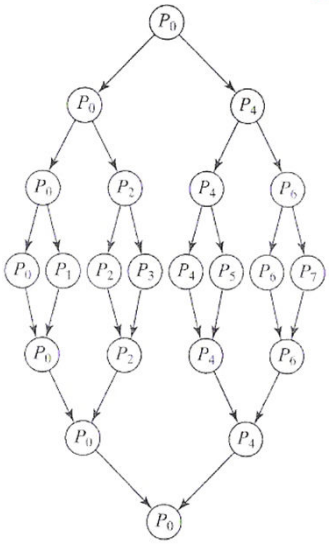
- Técnica "dividir-para-conquistar"
- Constituido por duas fases distintas:
  - Divisão
  - Junção (*merge*)
- Não é feita nenhuma computação na fase de divisão
- A ordenação acontece na fase de junção (*merge*)
- Veja o vídeo em http://www.youtube.com/watch?v=PKCMMSXQyJE&feature=related

## MERGE SORT SEQUENCIAL



## MERGE SORT PARALELO

- Na fase de divisão, atribui-se uma lista de elementos a cada processador
- Tem o problema de em algumas partes do algoritmo ser necessária bastante comunicação entre processadores
- Tem a vantagem de ser bastante leve a nível computacional em cada processador



# QUICKSORT

## CARACTERÍSTICAS

- Algoritmo recursivo
- Compara valores para ordenar uma lista
- Selecciona um número para *pivot*
- Divide a lista em duas sub-listas
- Chama-se a si próprio recursivamente para ordenar as duas sub-listas

## CARACTERÍSTICAS

- É um dos método mais rápidos de ordenação, apesar de às vezes partições desequilibradas poderem conduzir a uma ordenação lenta.
- Esse método de ordenação utiliza a técnica "dividir-para-conquistar" (dividir o problema inicial em dois subproblemas e resolver um problema menor utilizando a recursividade)

## CARACTERÍSTICAS

- Este método baseia-se na divisão da tabela em duas sub-tabelas, dependendo de um elemento chamado pivô.
- Uma das sub-tabelas contém os elementos menores que o pivô enquanto a outra contém os maiores.
- O pivô é colocado entre ambas, ficando na posição correta.
- As duas sub-tabelas são ordenadas de forma idêntica, até que se chegue à tabela com um só elemento.

- Esse método de ordenação divide-se em vários passos:

· Escolher para pivô o primeiro elemento da tabela (p=x[1])

·Se os elementos de x forem rearranjados de forma a que o pivô (p) sejam colocados na posição j e sejam respeitadas as seguintes condições:

1. todos os elementos entre as posições 1 e j-1 são menores ou iguais que o pivô (p)
2. todos os elementos entre as posições j+1 e n são maiores que o pivô (p)

Então p permanecerá na posição j no final do ordenamento.

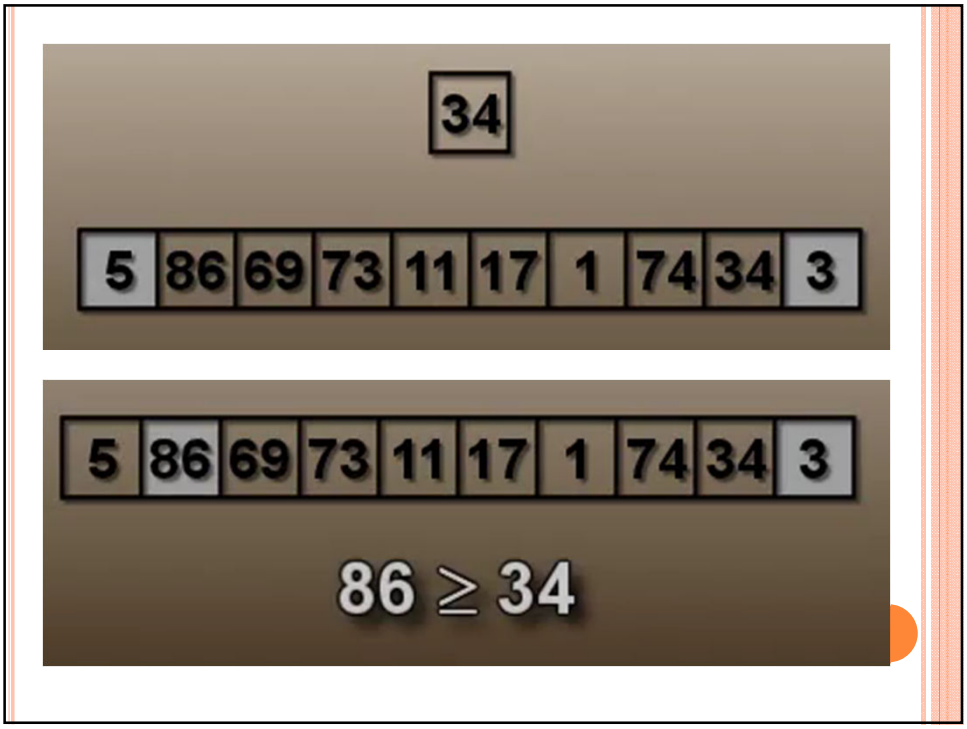
·Se este processo for repetido para as sub-tabelas x[1] a x[j-1] e x[j+1] a x[n] e para todas as sub-tabelas criadas nas iterações seguintes obteremos no final uma tabela ordenada.

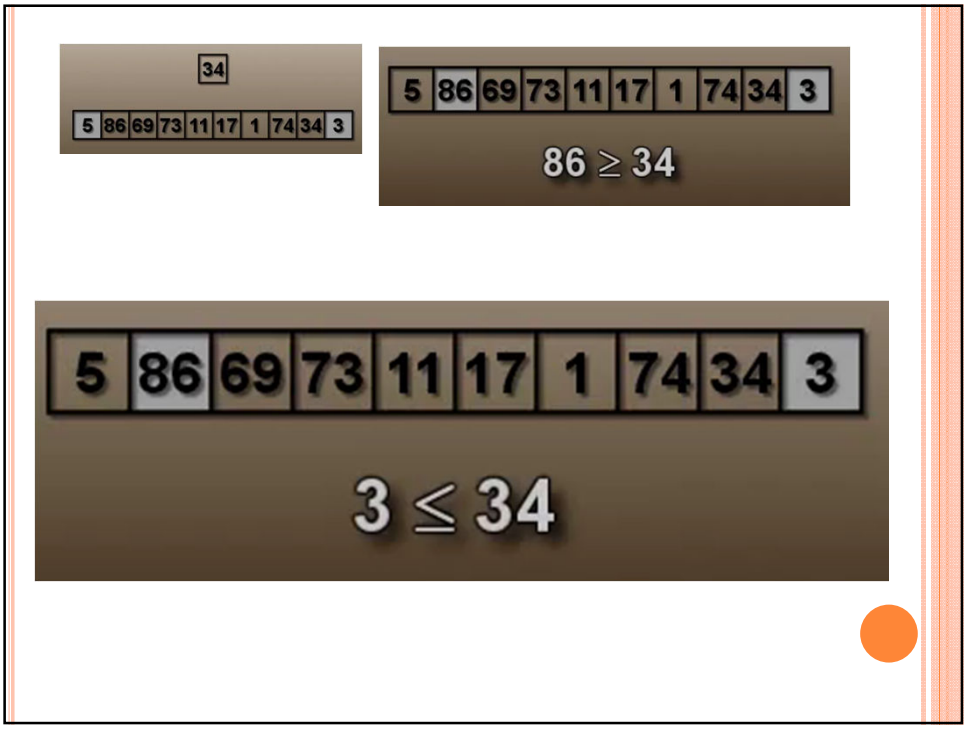
Portanto a parte mais difícil deste método é o procedimento **parte** que divide a tabela em 2 sub-tabelas dependendo do pivô.

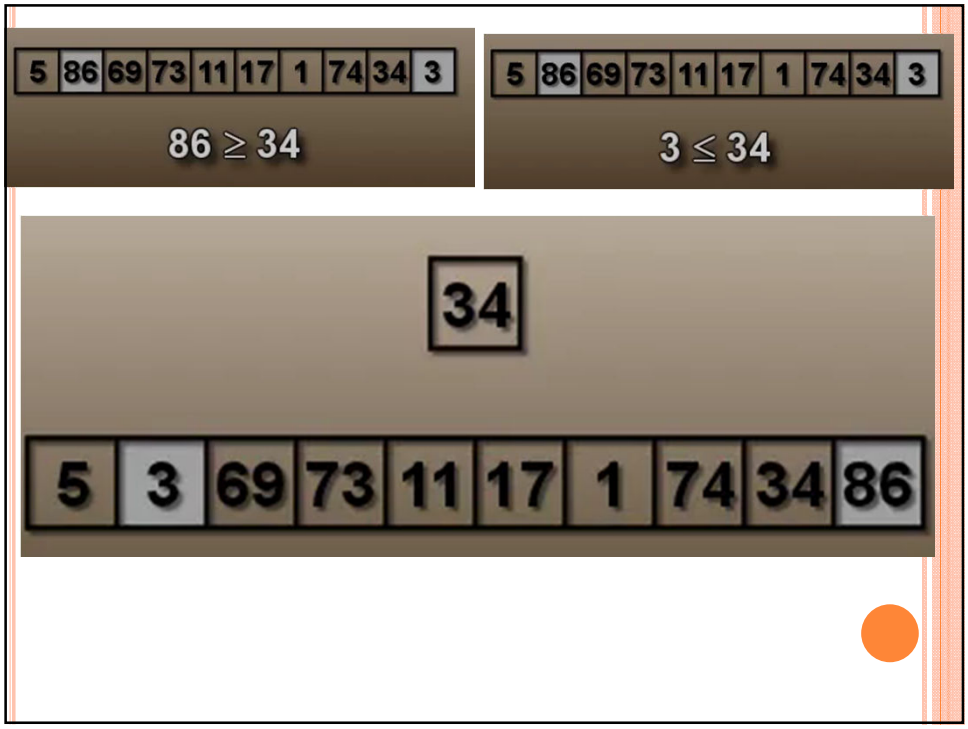
O método consiste em:

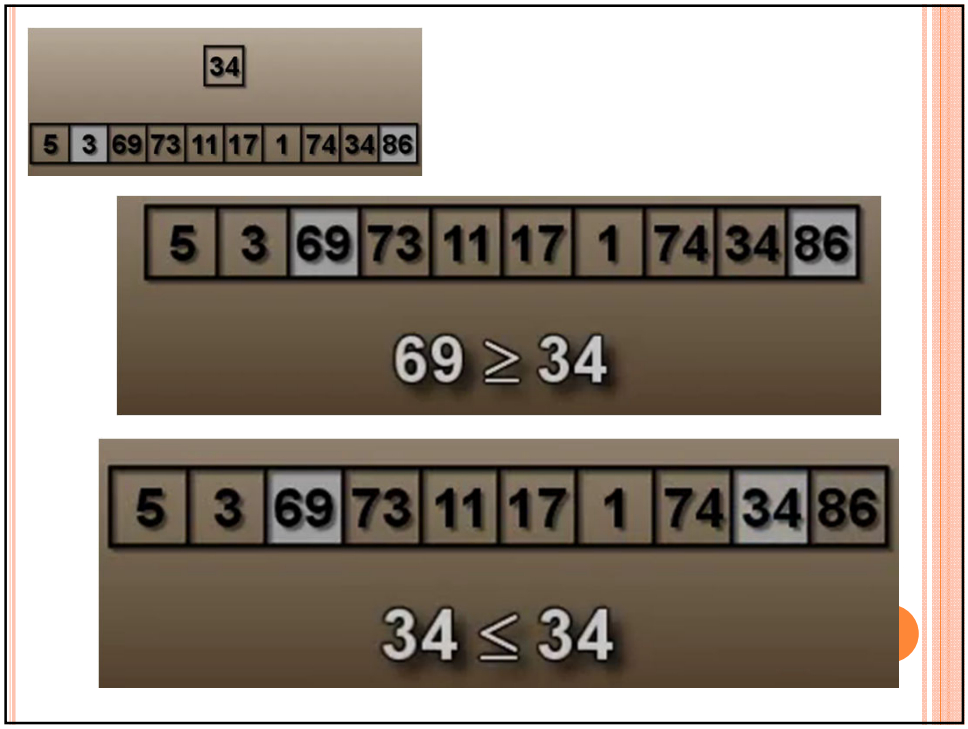
- Escolher um pivô inicial x;
  - Colocar todos itens com chave menor que a de x à esquerda de x, formando uma sequência S1;
  - Colocar todos itens com chave maior que a de x à direita de x, formando uma sequência S2;
- Isto feito, o mesmo processo é aplicado às sequências S1 e S2, que por sua vez produzirão novos segmentos;
- O processo deve ser aplicado sucessivamente às sequências enquanto elas tiverem `tamanho ≥ 1`.

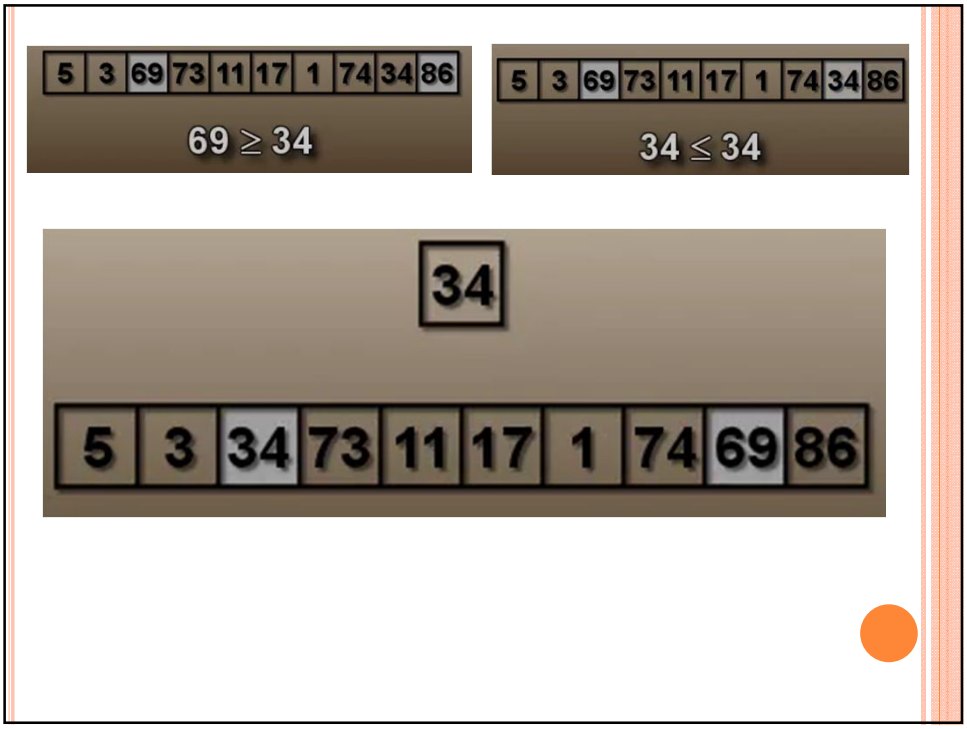

http://www.youtube.com/watch?v=y_G9BkAm6B8&feature=related

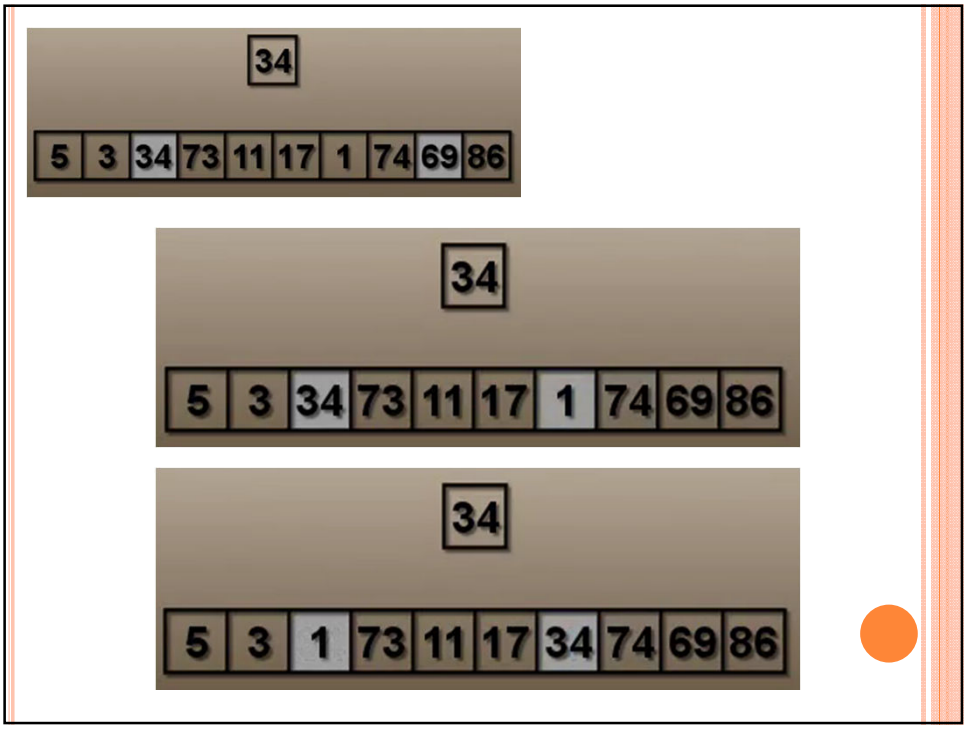

34
5 86 69 73 11 17 1 74 34 3
5 86 69 73 11 17 1 74 34 3
86 ≥ 34

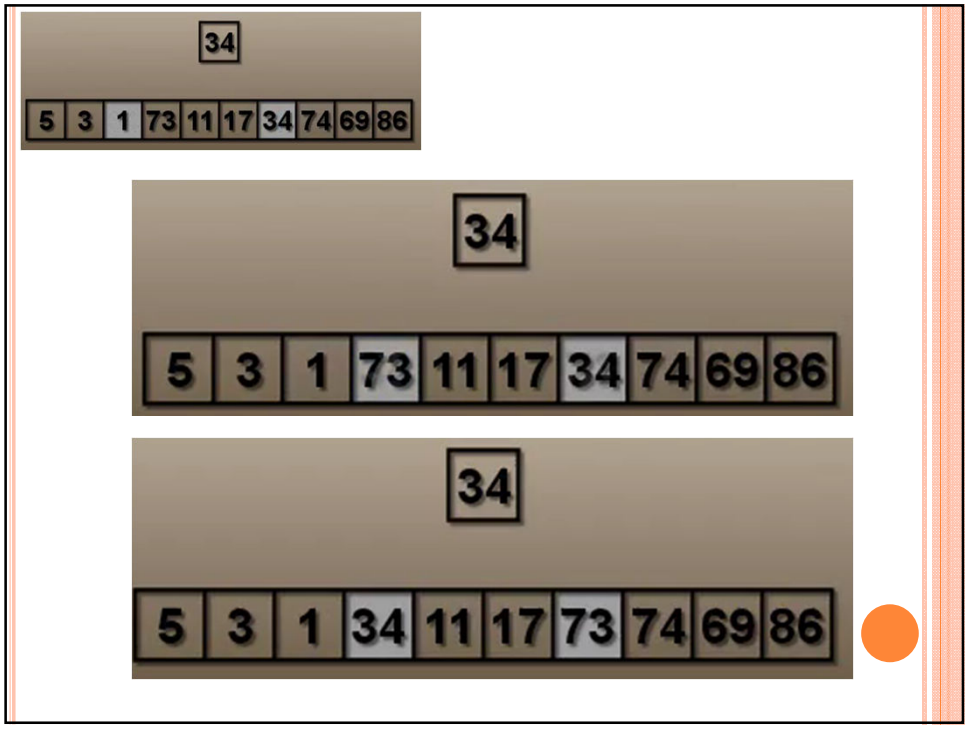

34
5 86 69 73 11 17 1 74 34 3
5 86 69 73 11 17 1 74 34 3
86 ≥ 34
5 86 69 73 11 17 1 74 34 3
3 ≤ 34

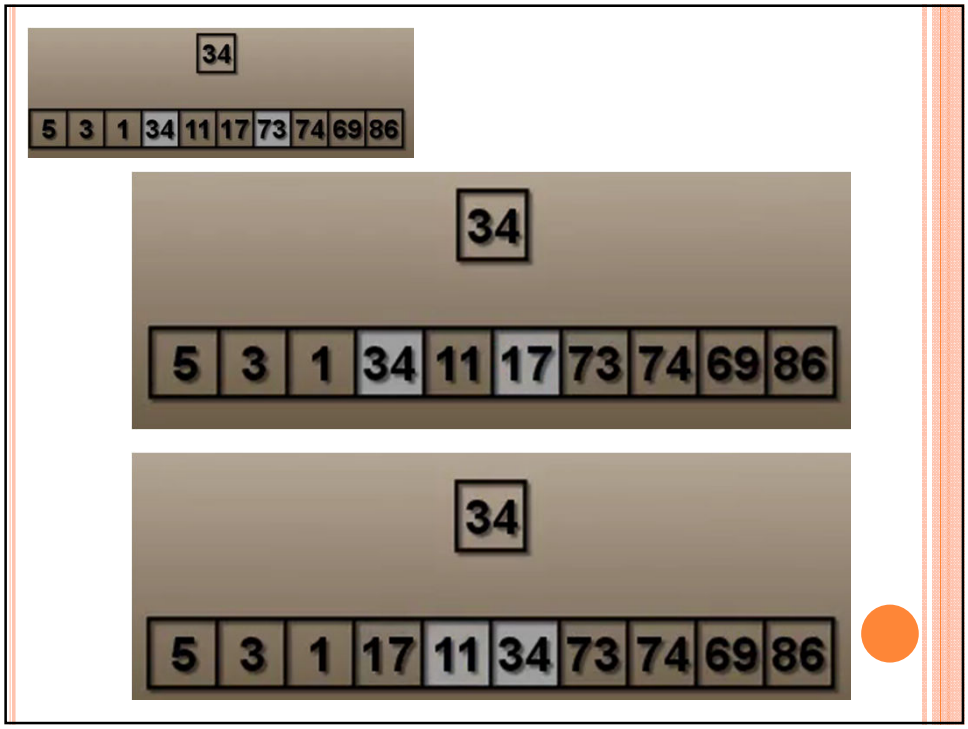

5 86 69 73 11 17 1 74 34 3
86 ≥ 34
5 86 69 73 11 17 1 74 34 3
3 ≤ 34
34
5 3 69 73 11 17 1 74 34 86


34
5 3 69 73 11 17 1 74 34 86
5 3 69 73 11 17 1 74 34 86
69 ≥ 34
5 3 69 73 11 17 1 74 34 86
34 ≤ 34

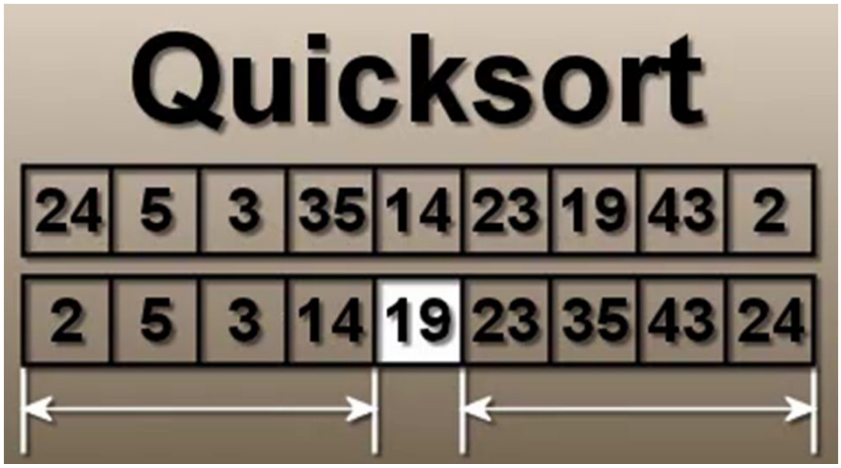

5 3 69 73 11 17 1 74 34 86
69 ≥ 34
5 3 69 73 11 17 1 74 34 86
34 ≤ 34
34
5 3 34 73 11 17 1 74 69 86


34
5 3 34 73 11 17 1 74 69 86
34
5 3 34 73 11 17 1 74 69 86
34
5 3 1 73 11 17 34 74 69 86

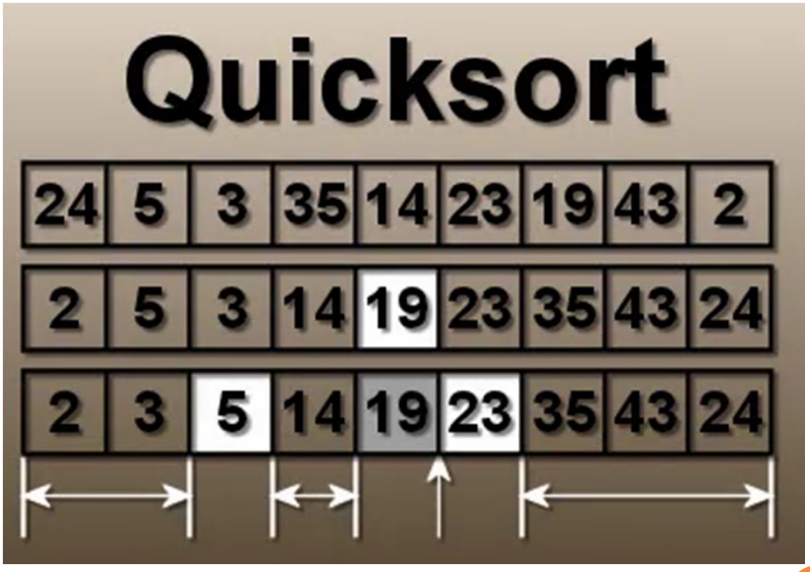

34
5 3 1 73 11 17 34 74 69 86
34
5 3 1 73 11 17 34 74 69 86
34
5 3 1 34 11 17 73 74 69 86


34
5 3 1 34 11 17 73 74 69 86
34
5 3 1 34 11 17 73 74 69 86
34
5 3 1 17 11 34 73 74 69 86


# Quicksort




http://www.youtube.com/watch?v=y_G9BkAm6B8&feature=related




http://www.youtube.com/watch?v=y_G9BkAm6B8&feature=related


http://www.youtube.com/watch?v=y_G9BkAm6B8&feature=related

# QuickSort Paralelo

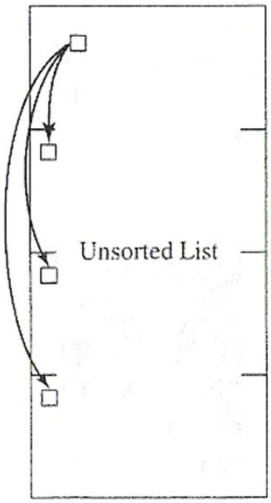
## Vantagens

- Considerado dos algoritmos mais rápidos
- Tem concorrência natural
- Não é necessário haver sincronização

## DESENVOLVIMENTO

1. Valores

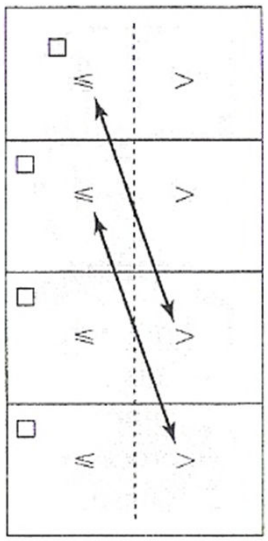
2.1.




## DESENVOLVIMENTO

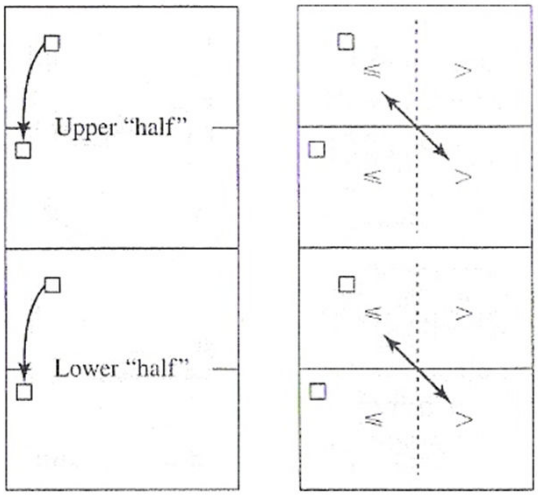
3.

4. T



## DESENVOLVIMENTO

5.




## DESENVOLVIMENTO

6.