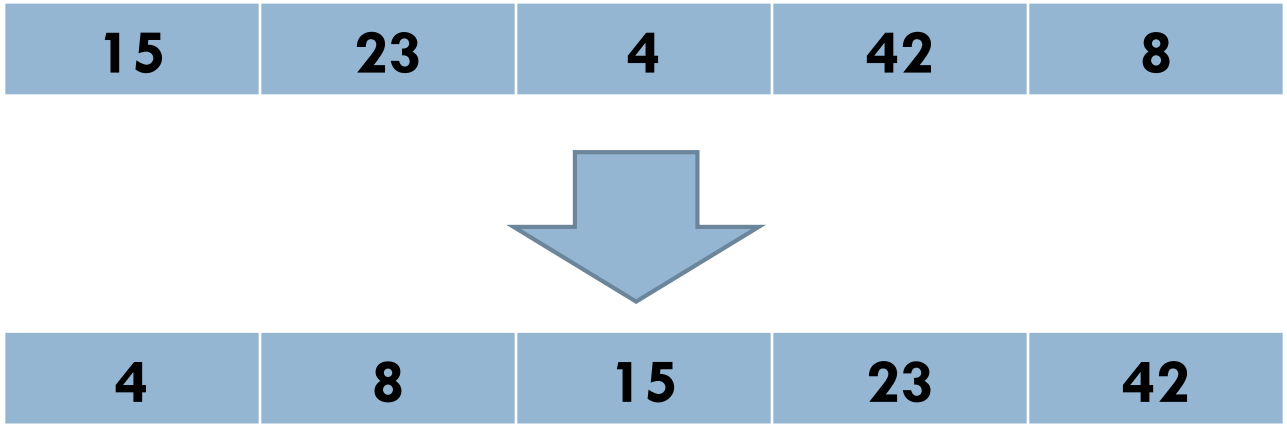
ORDENAÇÃO

Prof. Muriel Mazzetto

Estrutura de Dados

Ordenação

- Organizar os elementos em uma ordem específica:
  - Em um conjunto de *n* chaves {*k1*,*k2*,...,*kn*}, organizá-lo de forma que *k1*<*k2*<...<*kn*.
  - Exemplo:



Ordenação

- Métodos de ordenação mais conhecidos:
  - BubbleSort (por trocas);
  - InsertionSort (por inserção direta);
  - SelectionSort (seleção direta);
  - HeapSort (seleção em árvore);
  - MergeSort (por intercalação);
  - QuickSort (por trocas).

Simples
Muitas comparações
Poucas buscas
Poucos elementos

Complexos
Poucas comparações
Muitas buscas
Vários elementos

QuickSort

- Baseado no paradigma “Dividir e Conquistar”:
  - Divide o problema original em problemas menores e semelhantes.
  - Vetor é subdividido múltiplas vezes.
- Procedimento recursivo.
- A **complexidade** varia de acordo com os parâmetros:
  - Ordenação de entrada e escolha do pivô.

QuickSort

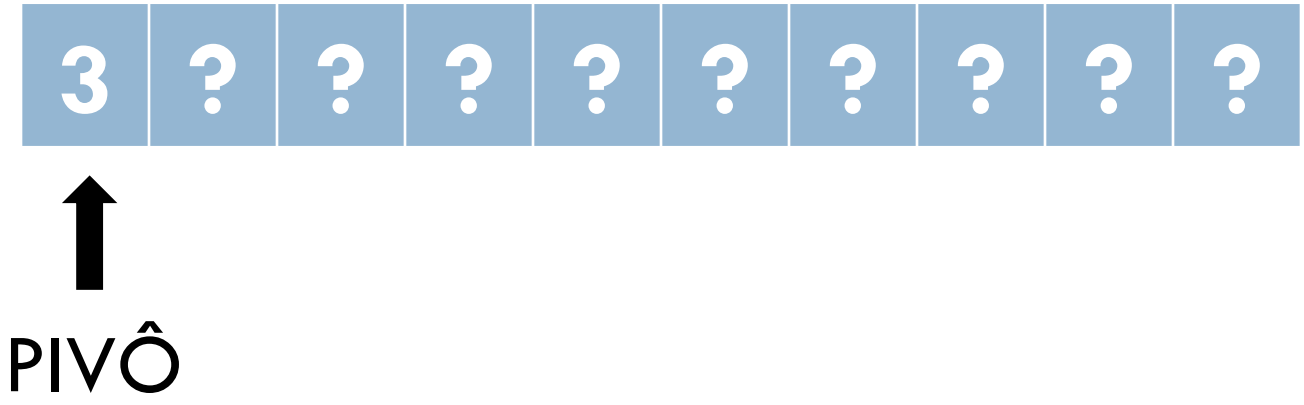
- Passos:
  - Dividir: escolher um elemento como pivô, dividindo o vetor de forma que os elementos à esquerda sejam menores que o pivô, e os elementos à direita sejam maiores.
  - Conquistar: realizar o procedimento de divisão para cada subvetor criado (recursão).
  - Combinar: após subdividir todos os elementos do vetor, os dados estarão ordenados.

QuickSort

- Selecionar o pivô: primeiro elemento.
- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.

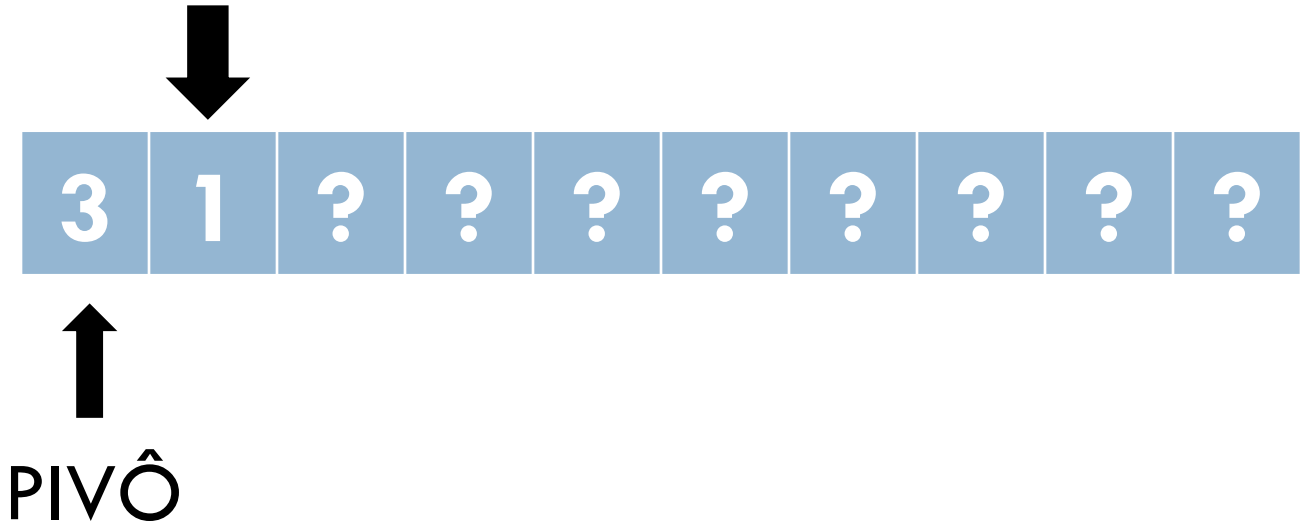
QuickSort

- Selecionar o pivô: primeiro elemento.
  - Existem outros métodos de seleção.
    - Meio do vetor, Aleatório, Média de posições, etc.



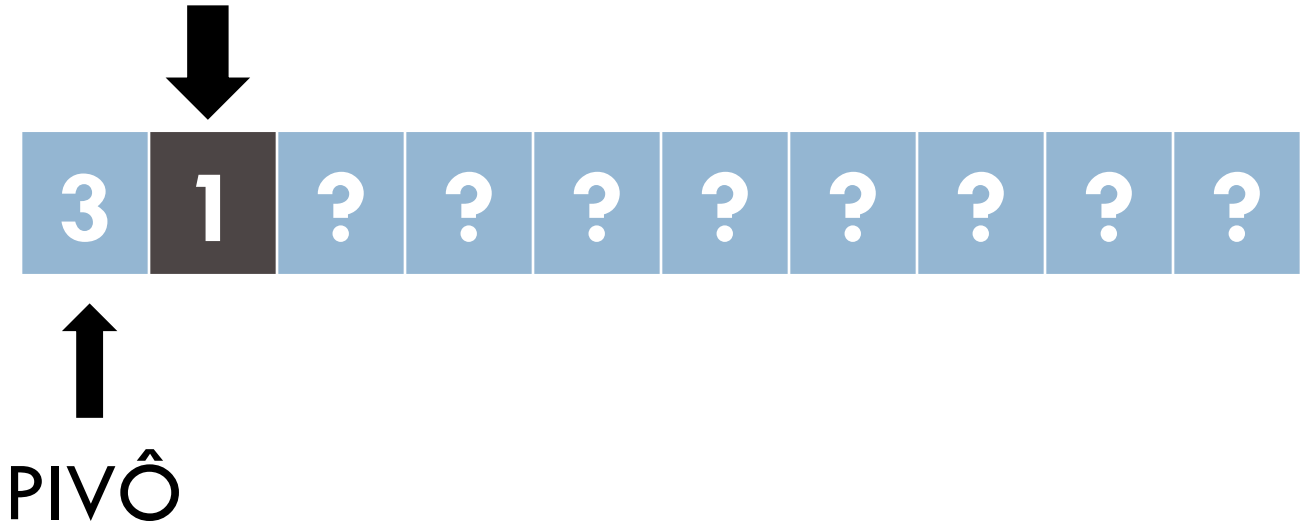
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.



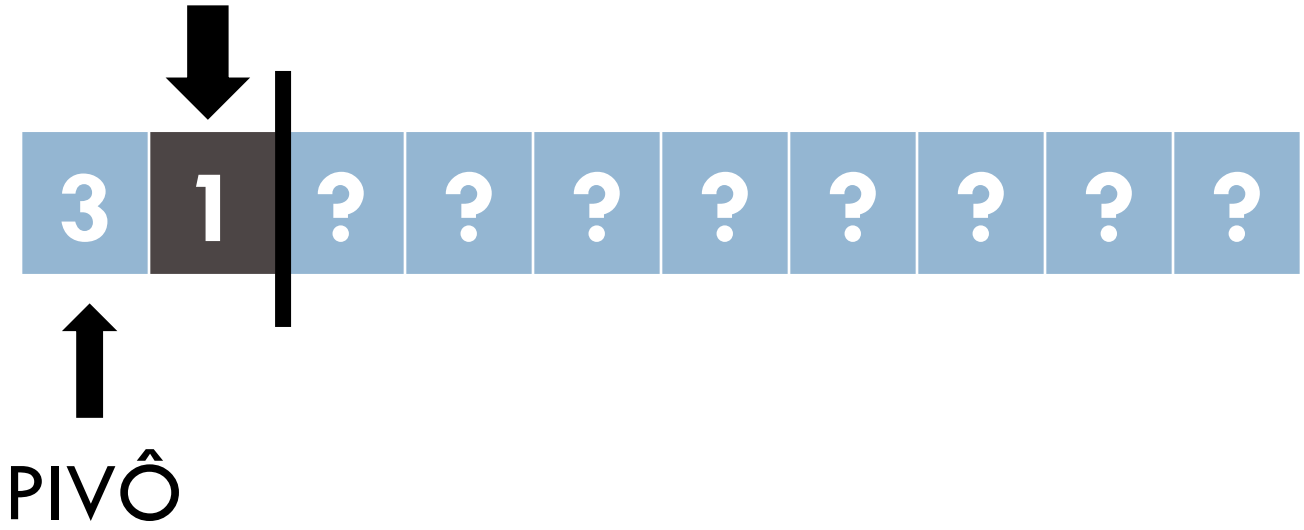
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**



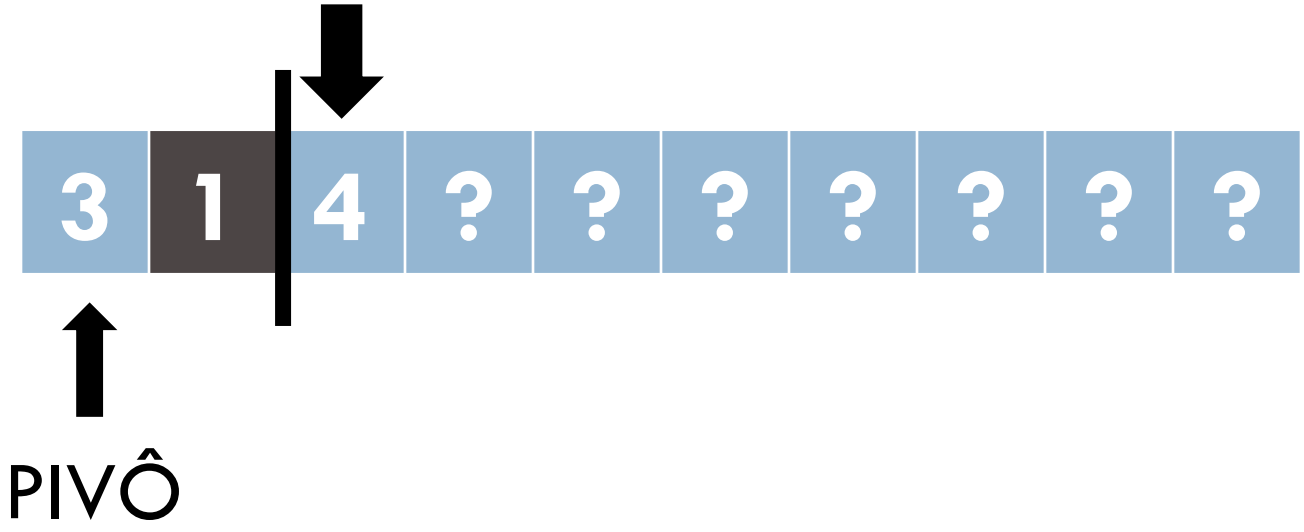
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.
  - **Marcar o índice que indica a separação de dir/esq.**



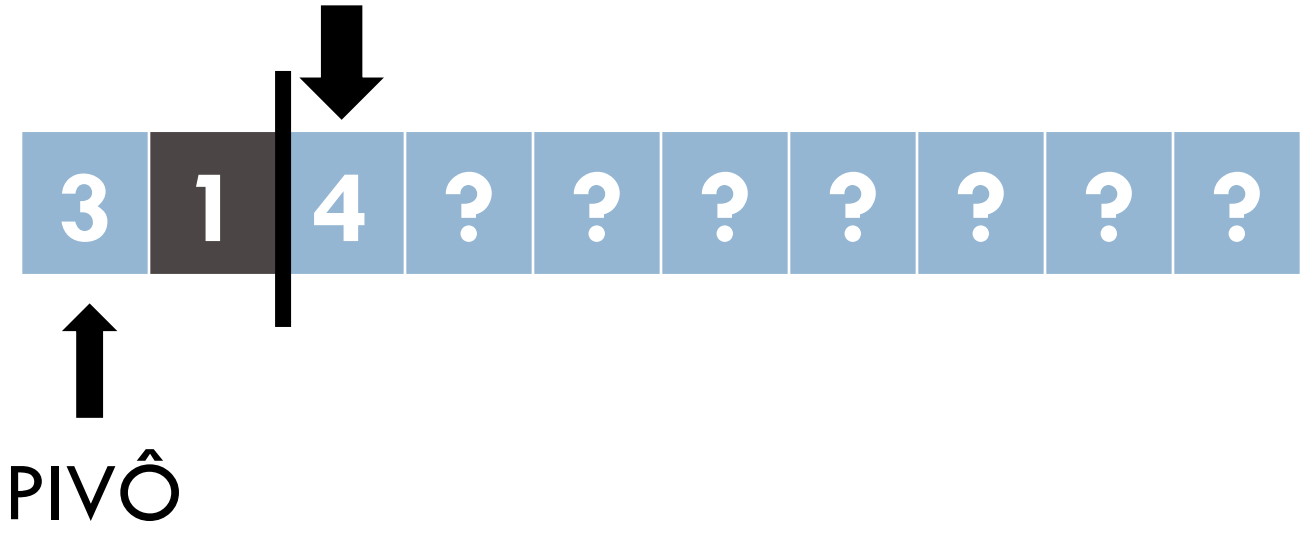
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.
  - Marcar o índice que indica a separação de dir/esq.



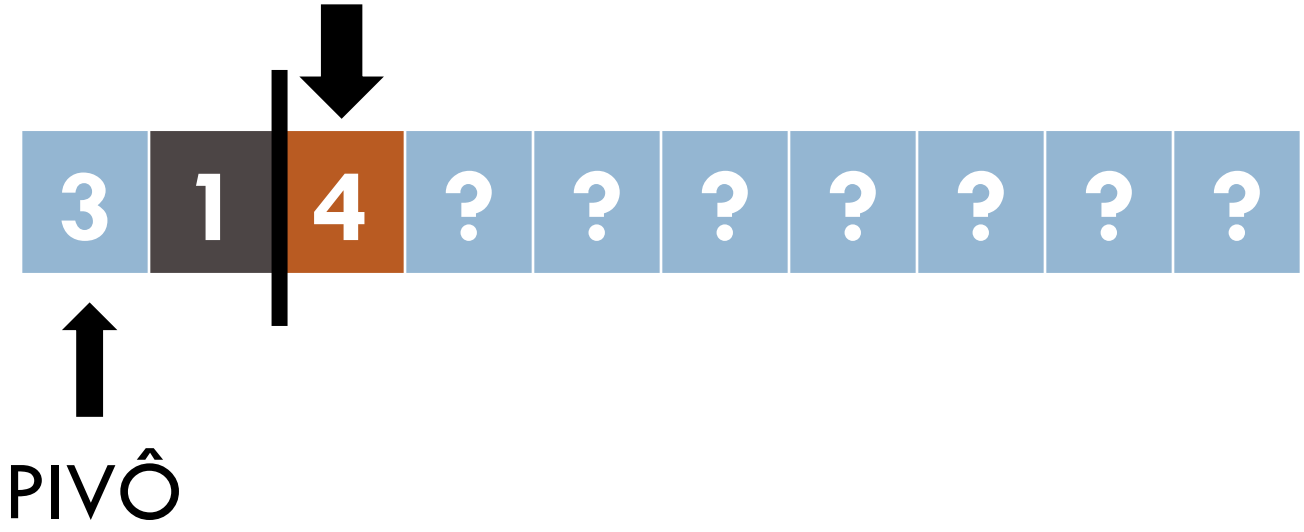
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.



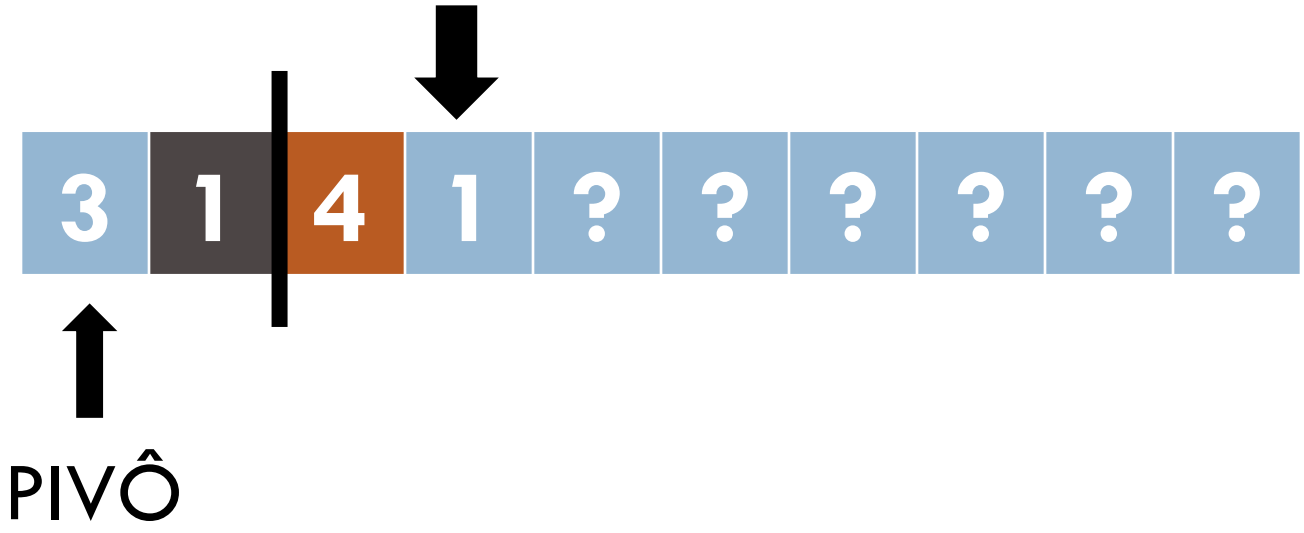
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.
  - **Marcar o índice que indica a separação de dir/esq.**



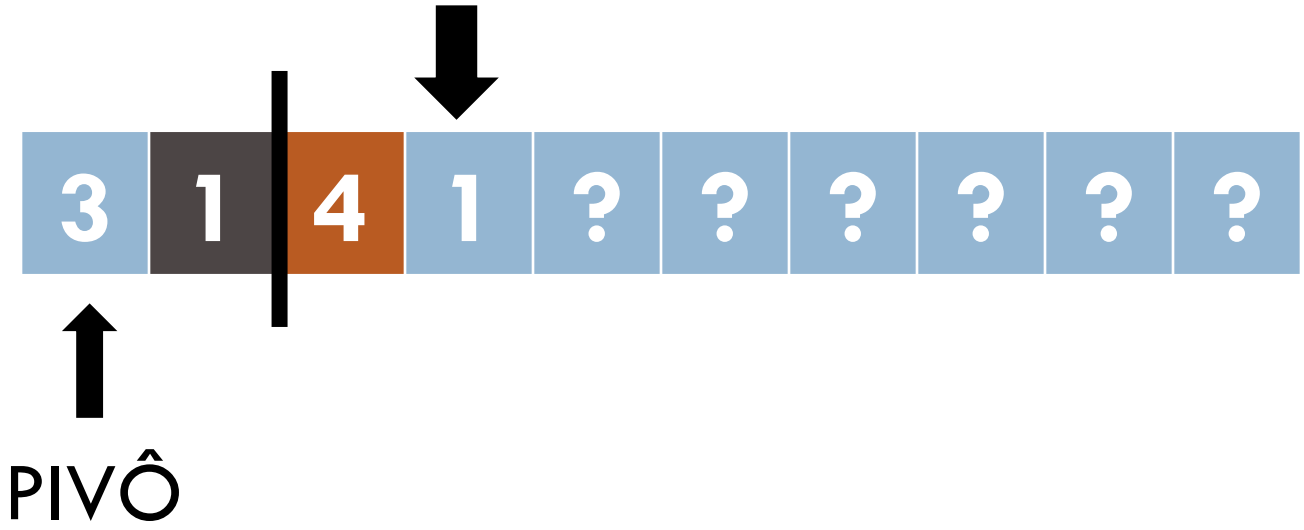
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.
  - Marcar o índice que indica a separação de dir/esq.



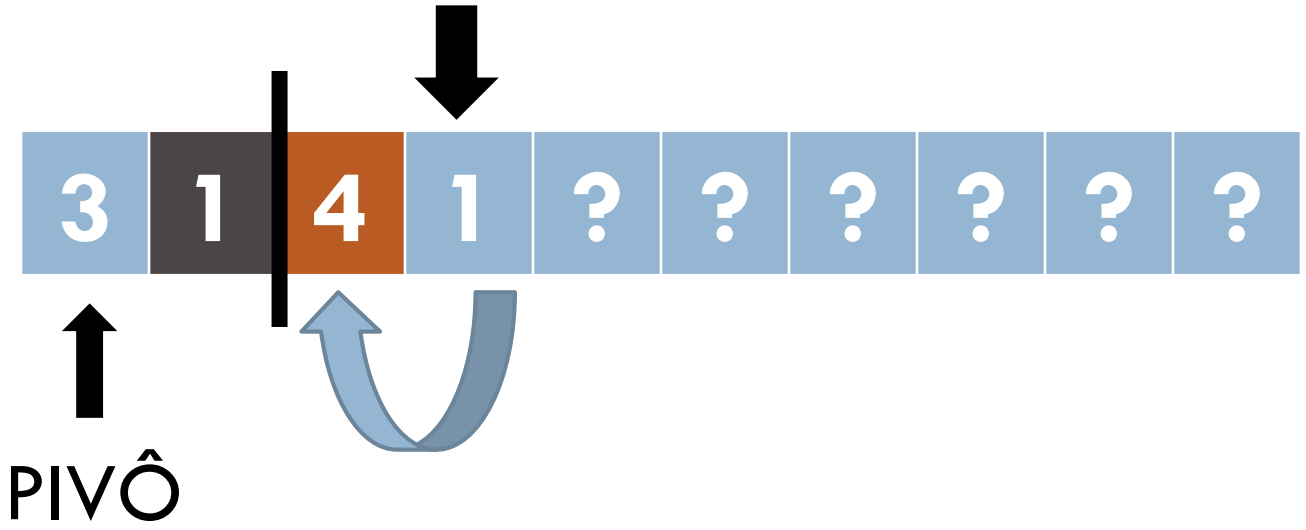
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.



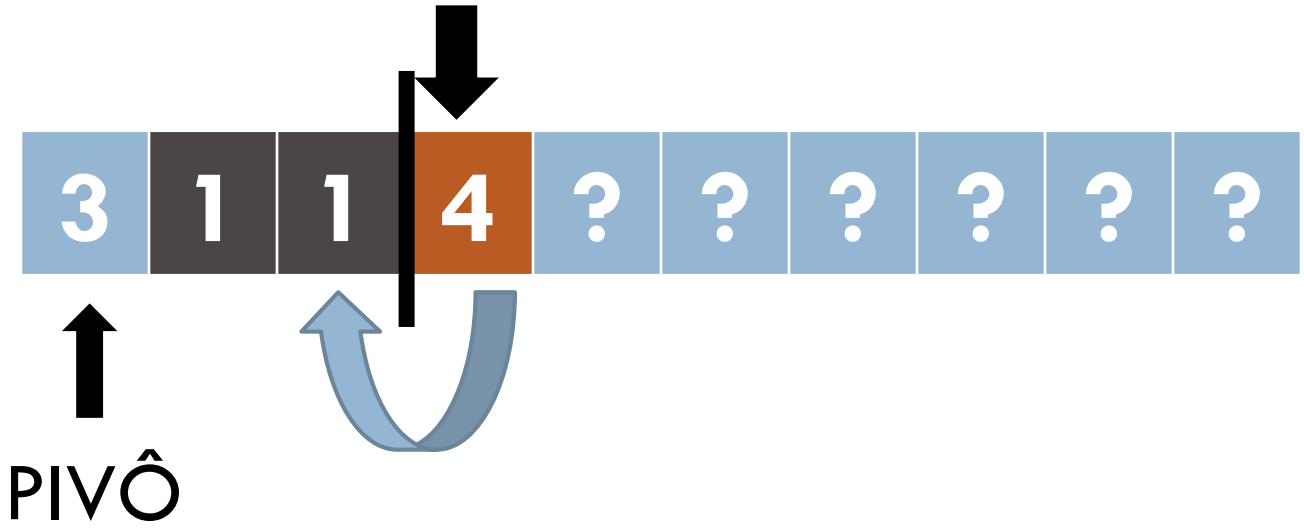
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.



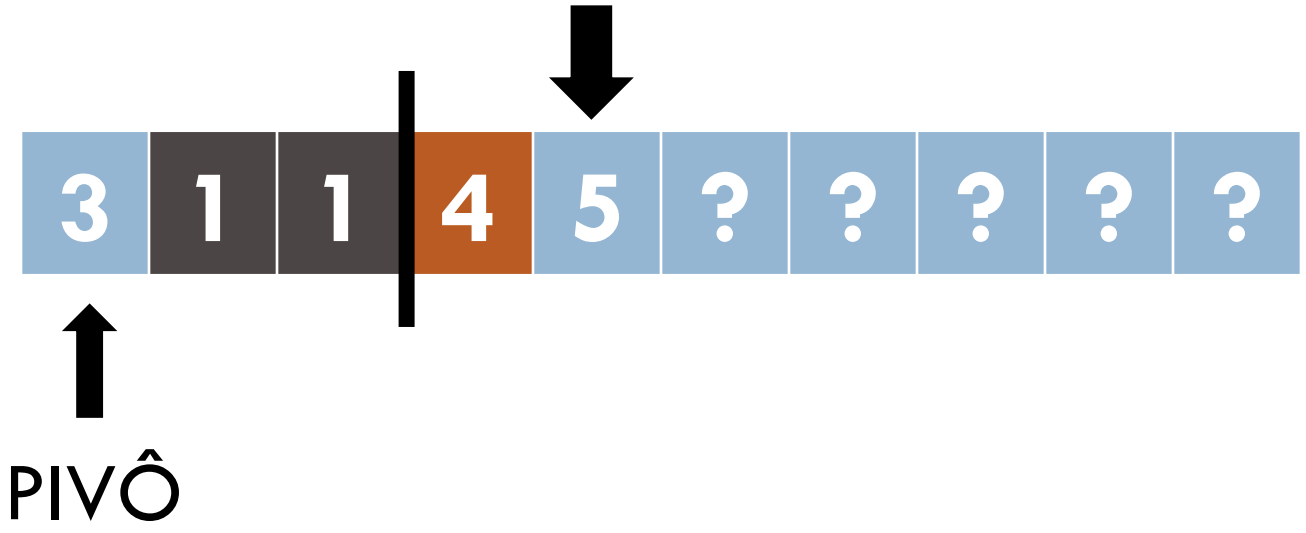
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.
  - **Marcar o índice que indica a separação de dir/esq.**



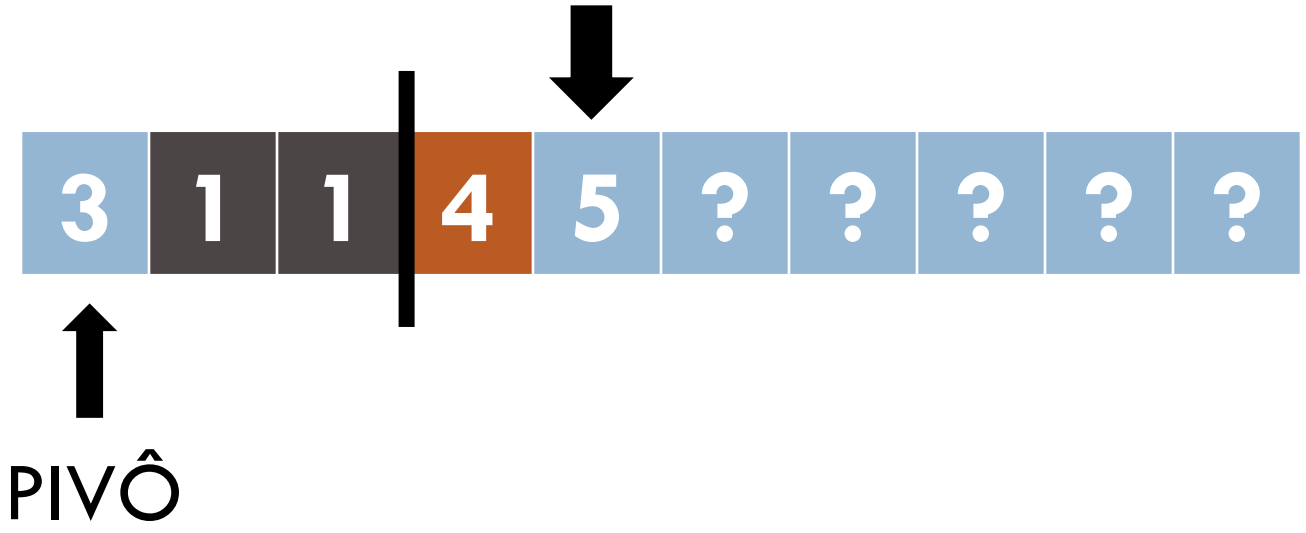
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.
  - Marcar o índice que indica a separação de dir/esq.



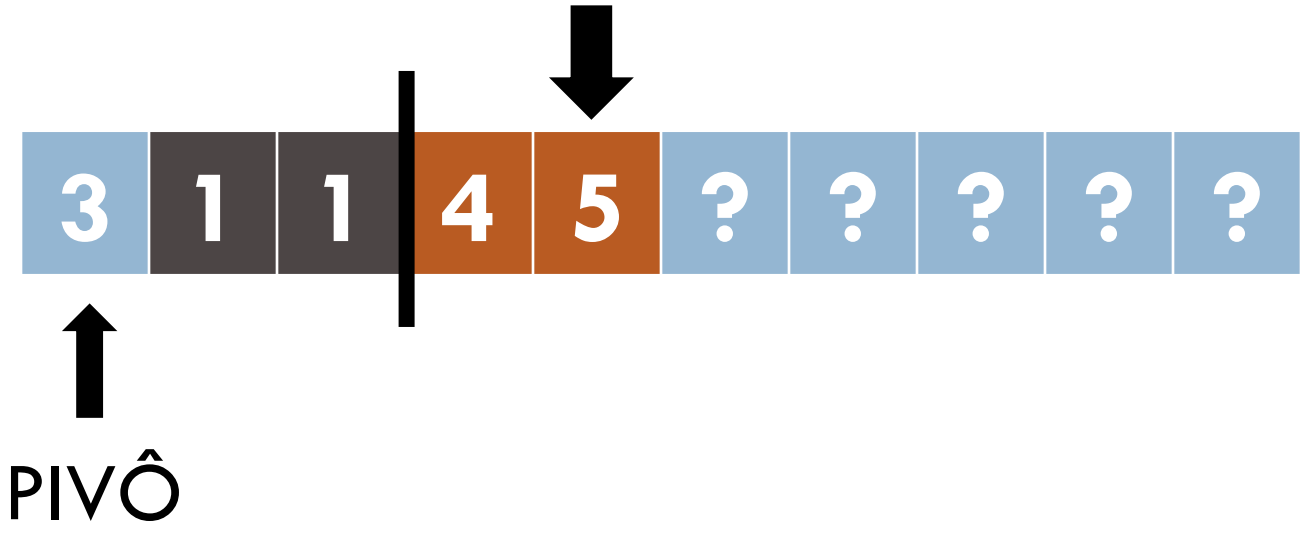
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.



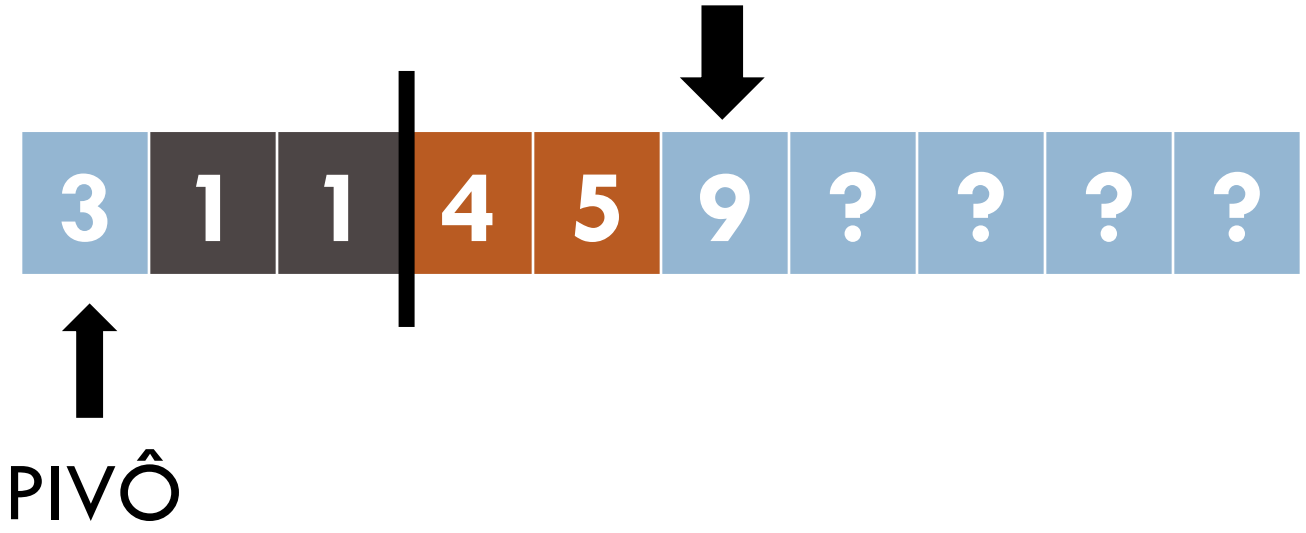
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.



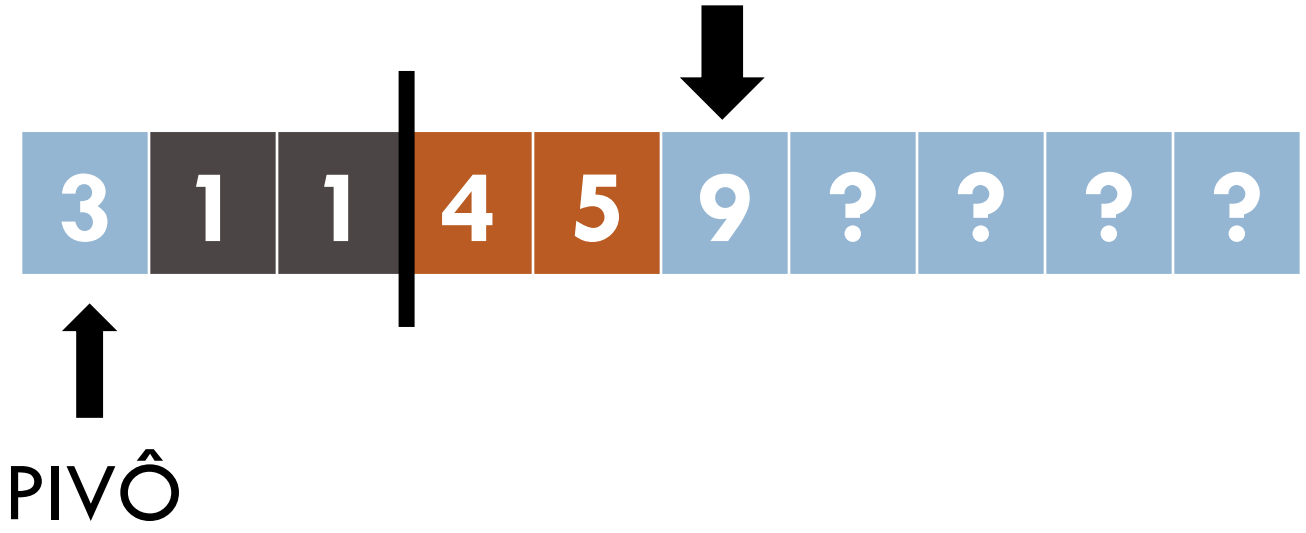
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.
  - Marcar o índice que indica a separação de dir/esq.



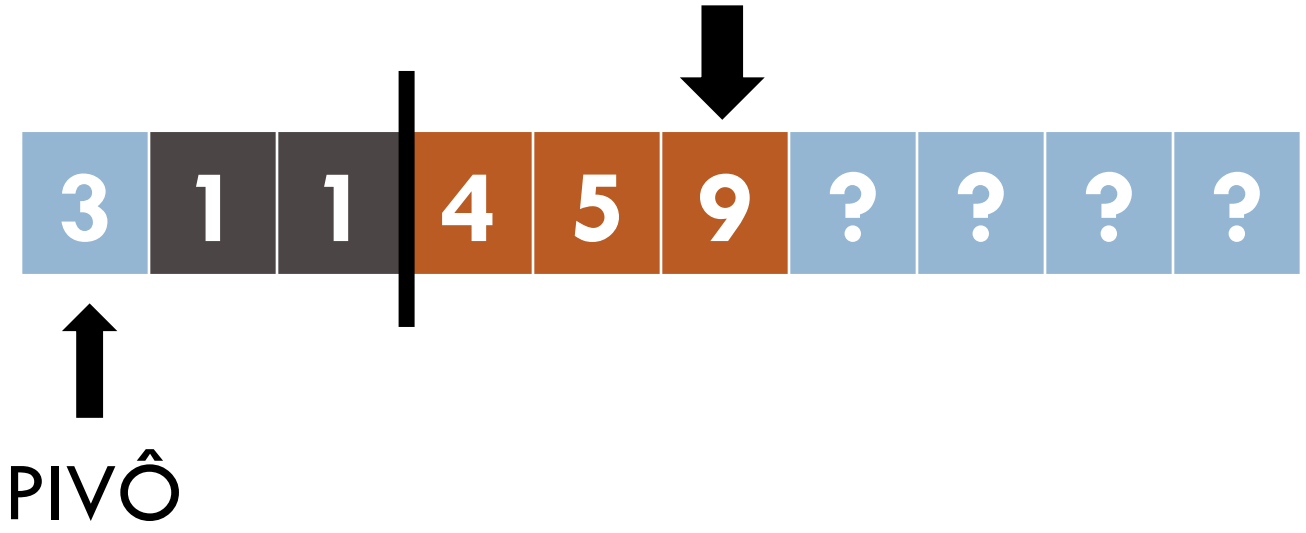
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.



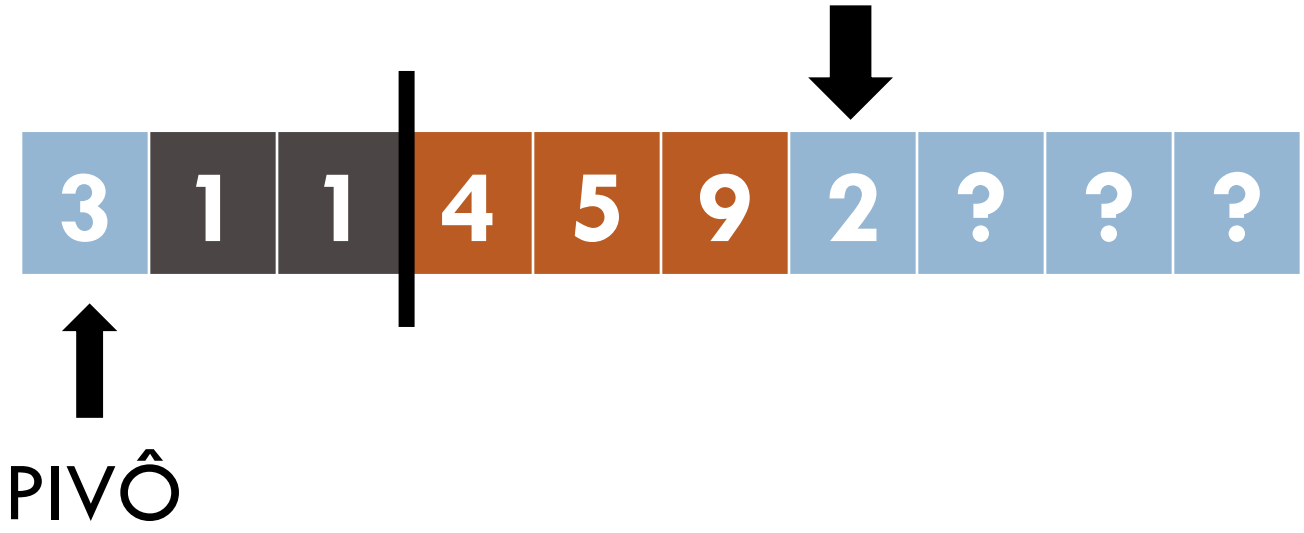
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.



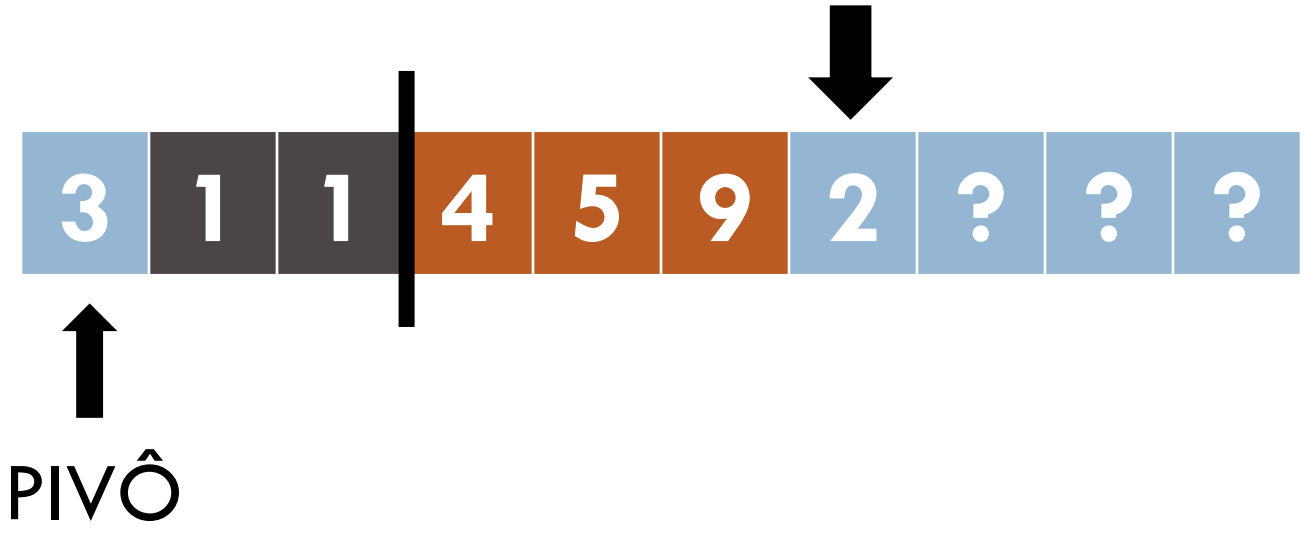
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.
  - Marcar o índice que indica a separação de dir/esq.



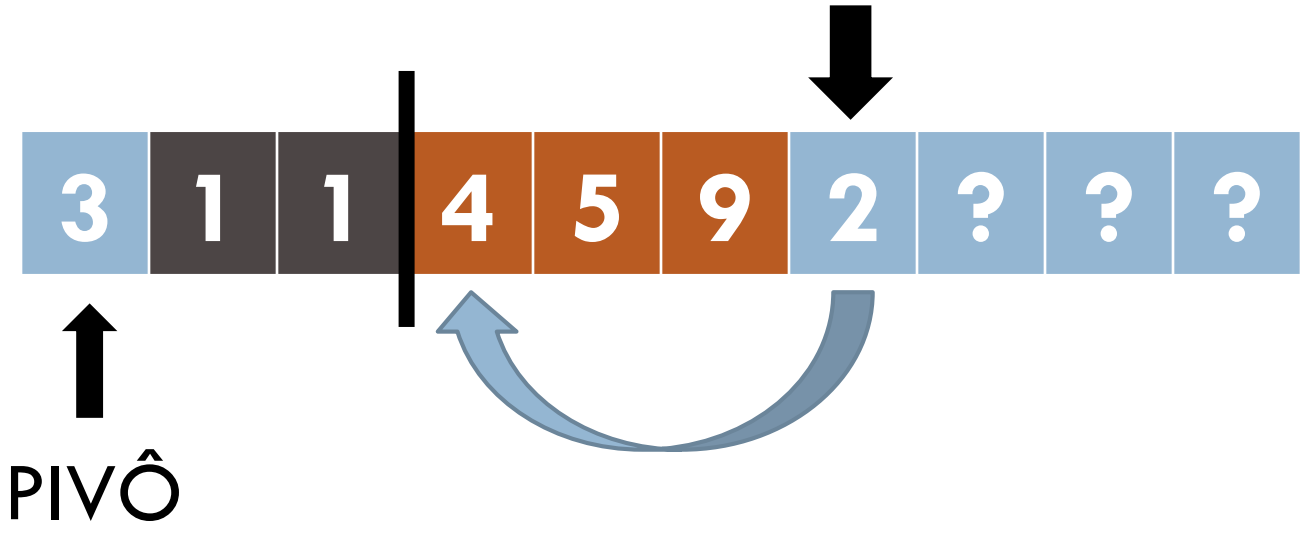
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.



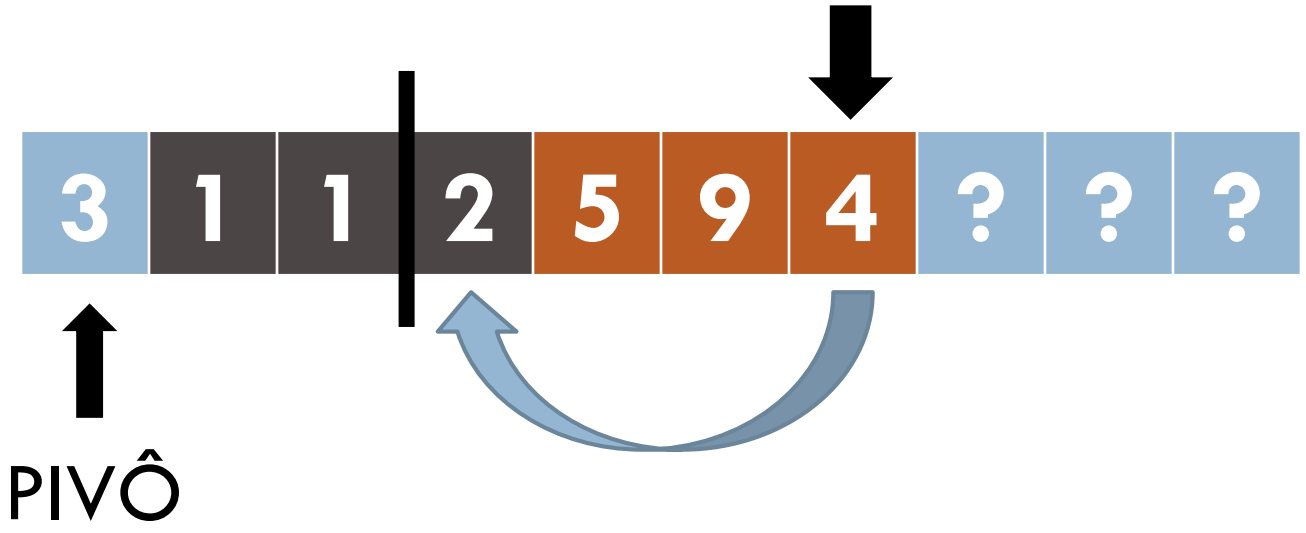
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.



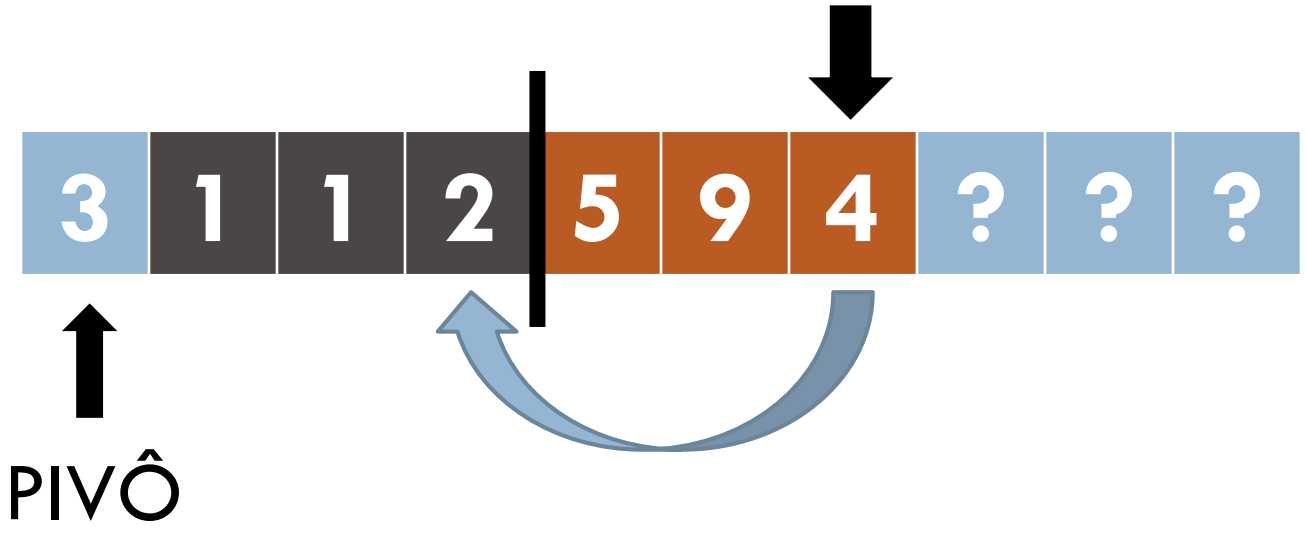
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.



QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.
  - **Marcar o índice que indica a separação de dir/esq.**



QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.
  - Marcar o índice que indica a separação de dir/esq.

QuickSort

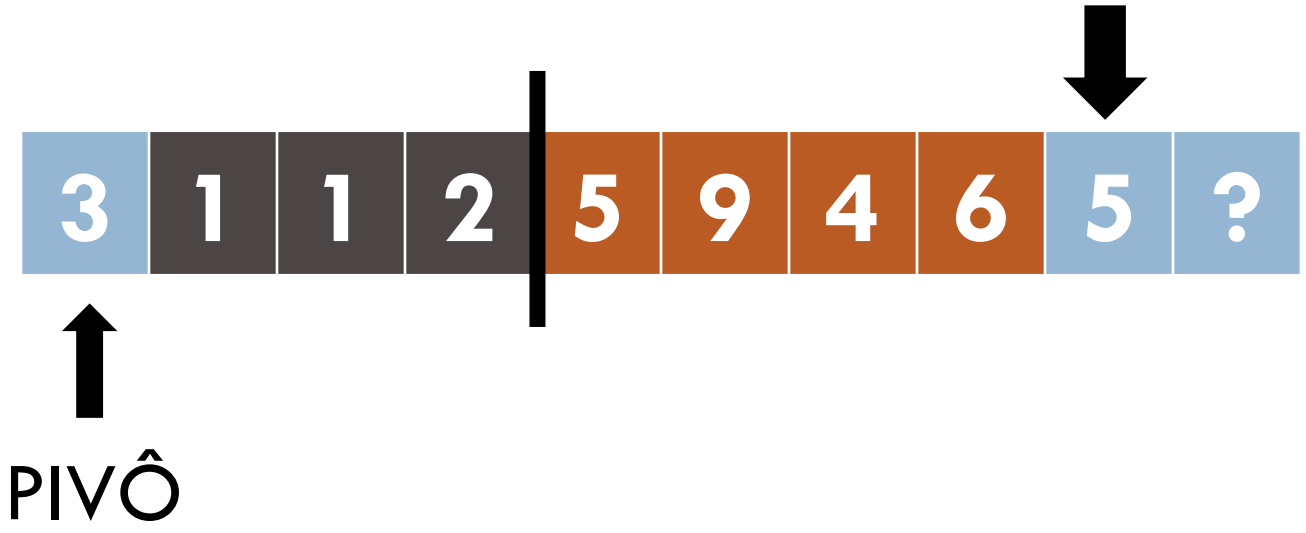
- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.

QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.

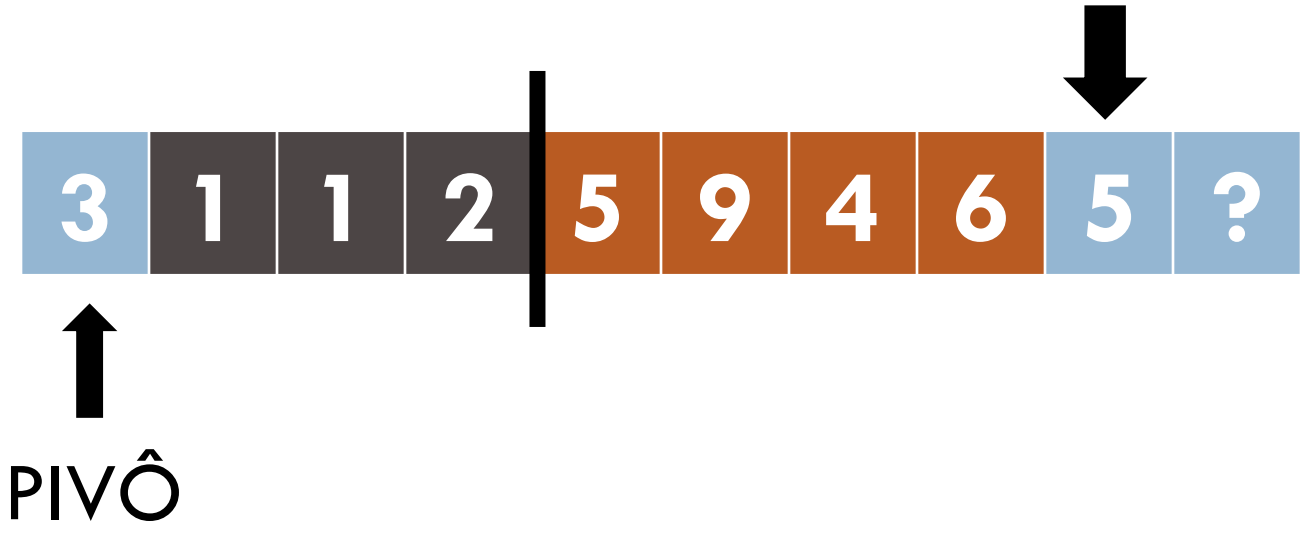
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.
  - Marcar o índice que indica a separação de dir/esq.



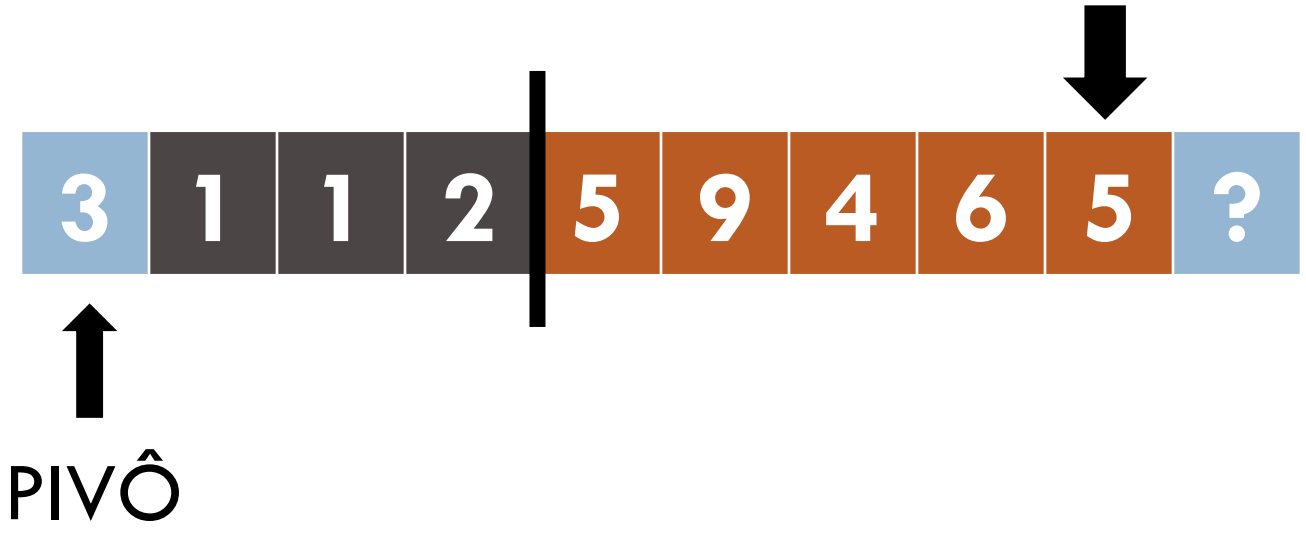
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.



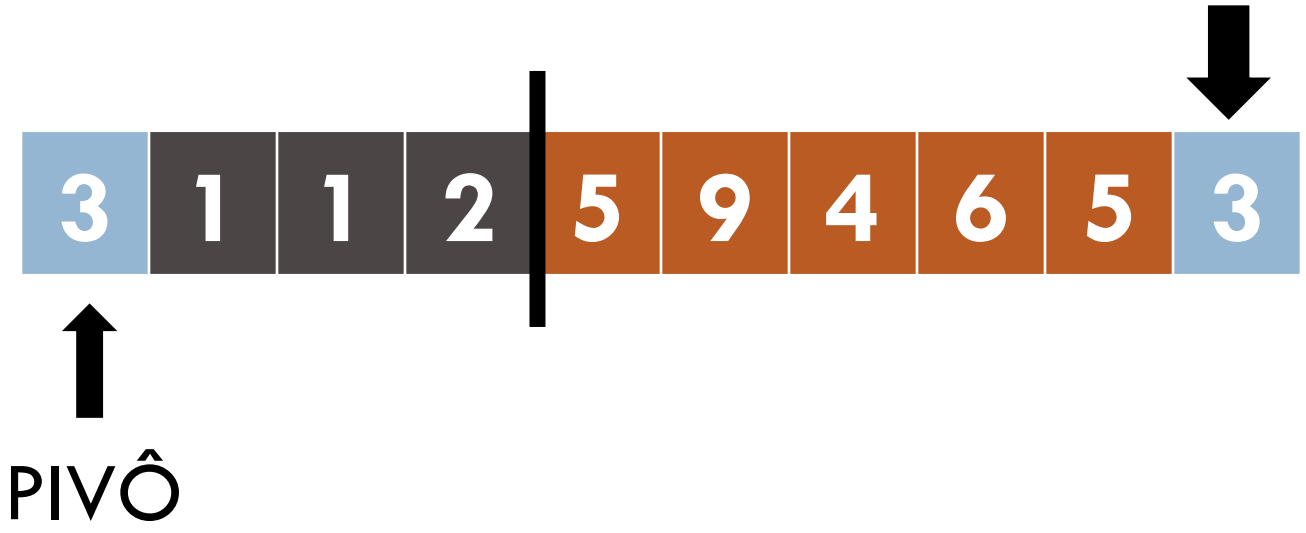
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.



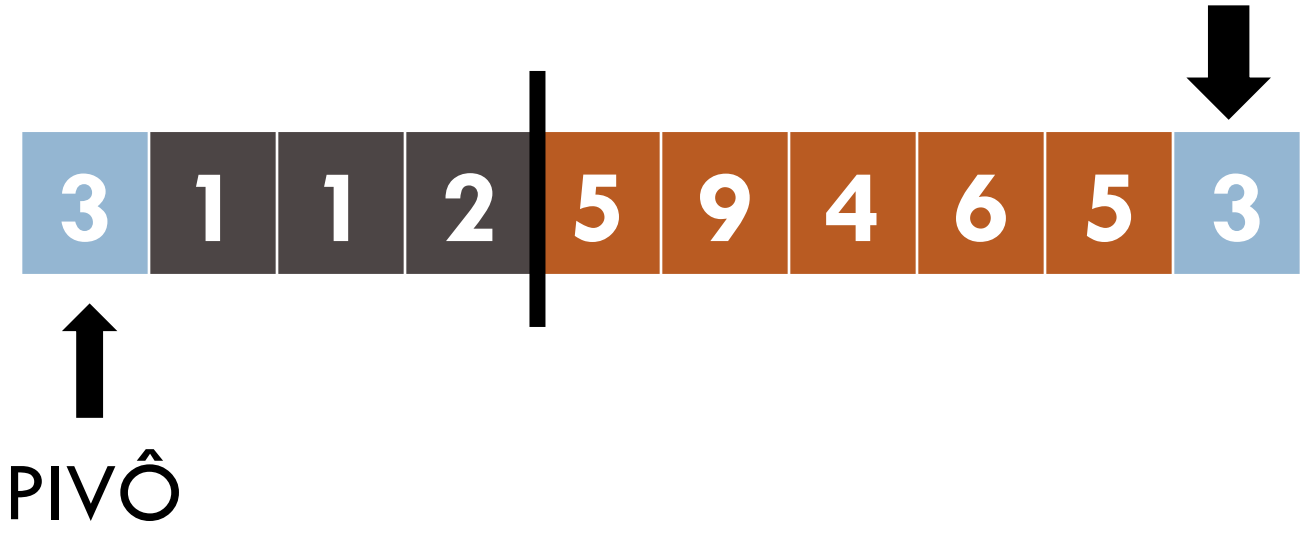
QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - Menores para a esquerda, maiores para a direita.
  - Marcar o índice que indica a separação de dir/esq.



QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
  - Marcar o índice que indica a separação de dir/esq.

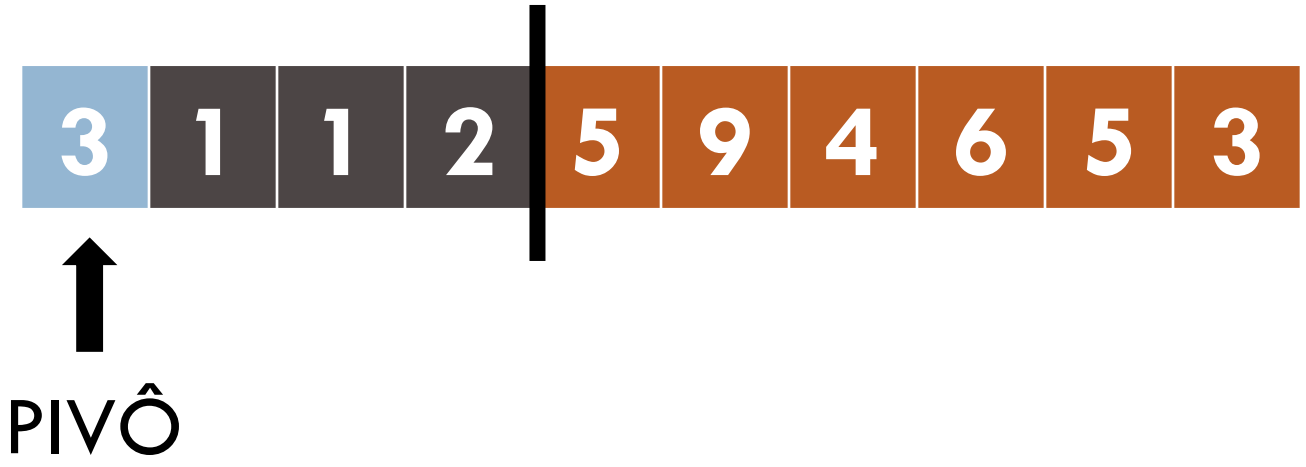


QuickSort

- Percorrer o vetor da esquerda para a direita, comparando chaves:
  - **Menores para a esquerda, maiores para a direita.**
    - **Manter iguais à direita para manter estabilidade.**
  - Marcar o índice que indica a separação de dir/esq.

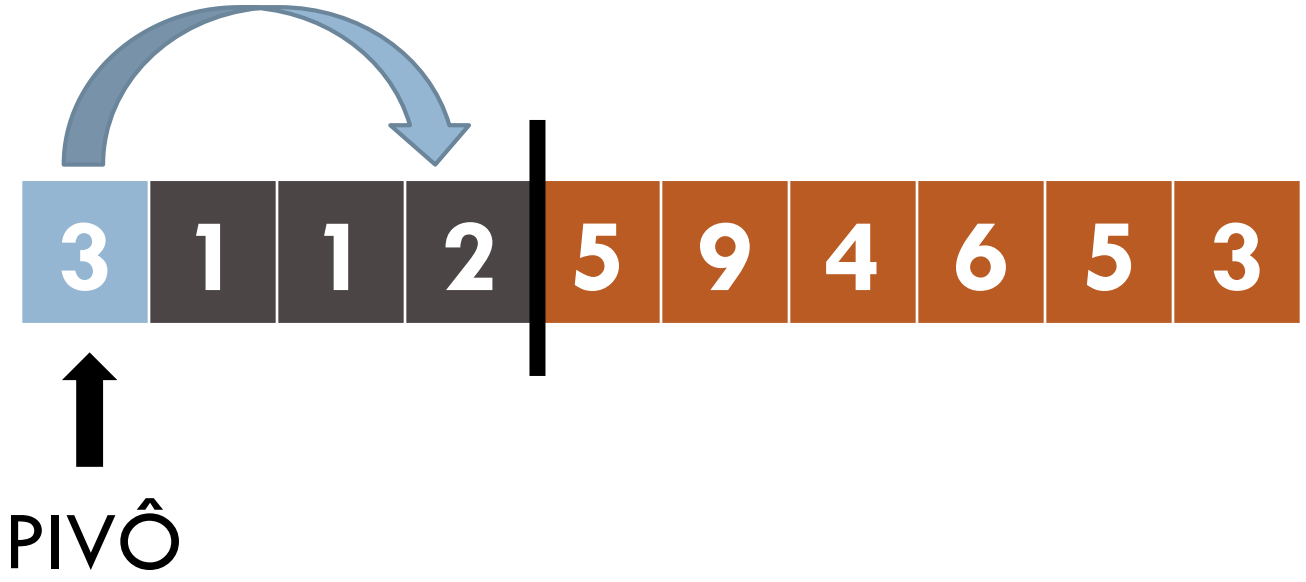
QuickSort

- Posicionar o pivô no local que marca a separação de direita e esquerda.



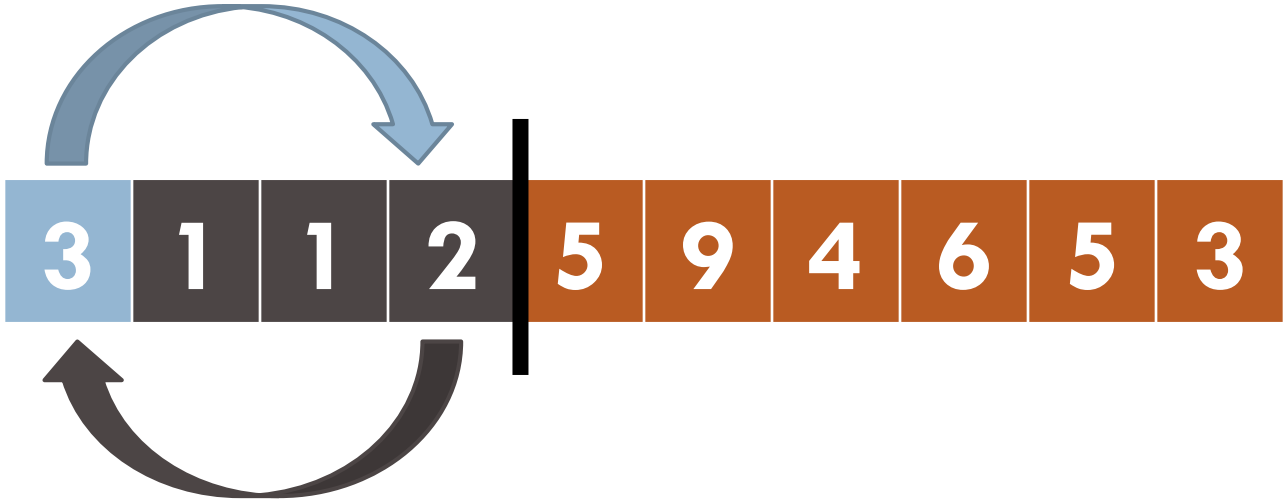
QuickSort

- Posicionar o pivô no local que marca a separação de direita e esquerda.



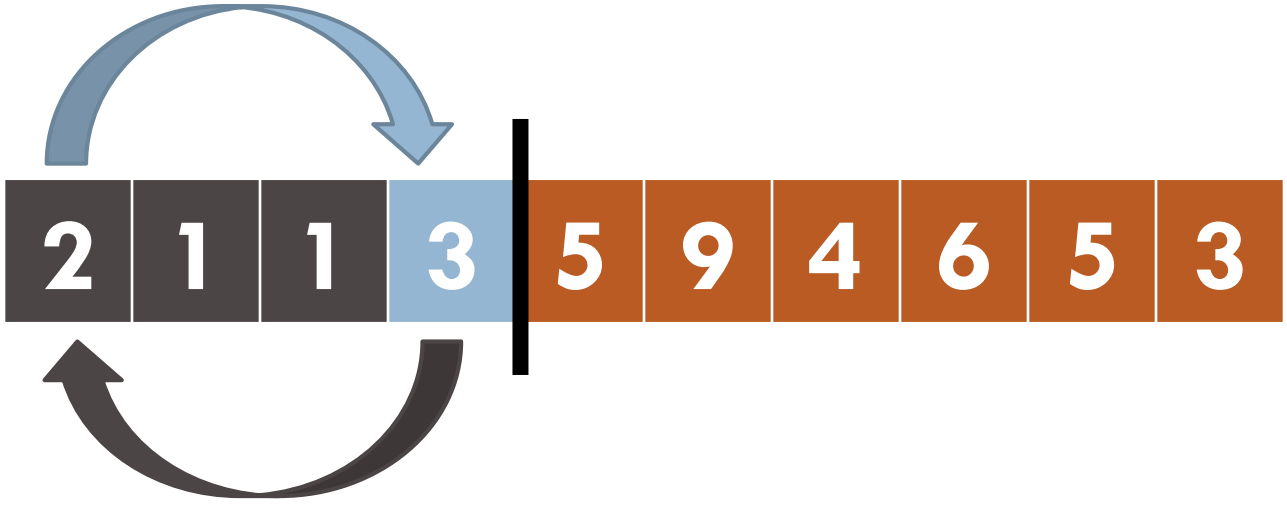
QuickSort

- Posicionar o pivô no local que marca a separação de direita e esquerda.



QuickSort

- Posicionar o pivô no local que marca a separação de direita e esquerda.
  - Manter menores para a esquerda e maiores para a direita.

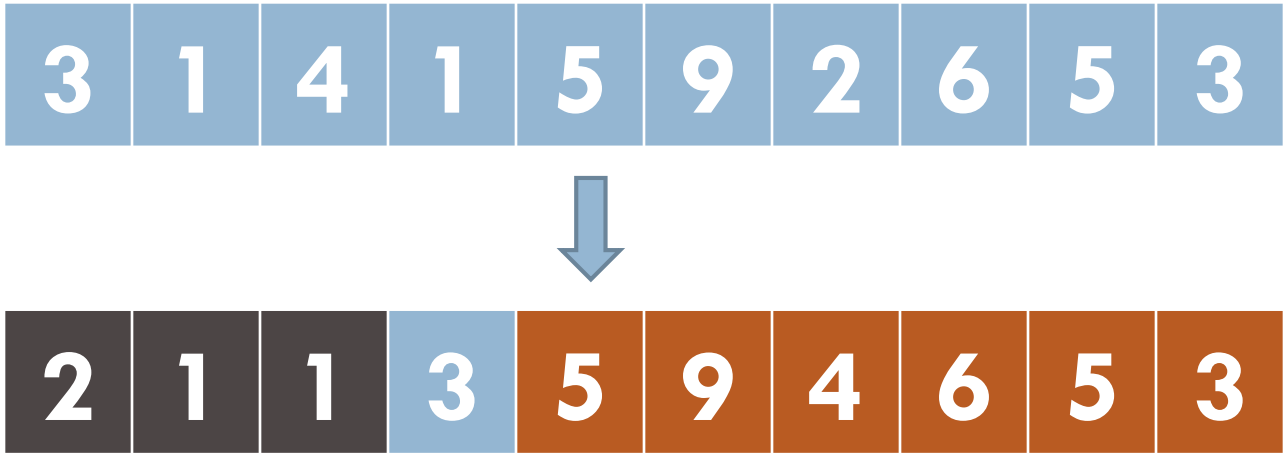


QuickSort

- Posicionar o pivô no local que marca a separação de direita e esquerda.
  - Manter menores para a esquerda e maiores para a direita.

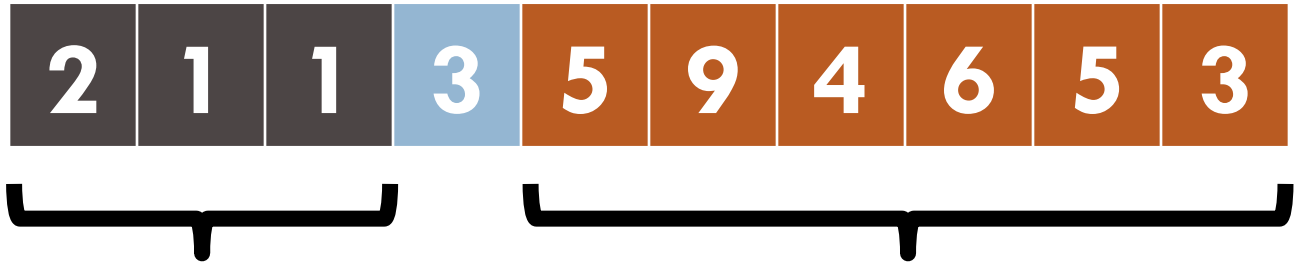
QuickSort

- Posicionar o pivô no local que marca a separação de direita e esquerda.
  - Manter menores para a esquerda e maiores para a direita.



QuickSort

- Realizar o mesmo procedimento para cada subvetor criado (esquerda e direita do pivô atual).
  - O pivô atual está fixado na sua posição final, e não será mais alterado.



QuickSort

- Exemplo alternativo:
  - Comparar valores das extremidades.

6 5 3 1 8 7 2 4

QuickSort

- Subdivide e organiza recursivamente, até não existir mais partições possíveis.
- Complexidade: O(n^2) no pior caso.

  O($nlogn$) no melhor e médio.
- Formas de seleção do pivô influenciam no tempo de execução.
- Pode ser instável.
- Não é adaptável.
- Vídeo no Youtube

QuickSort

- Útil apenas quando se possuem vários elementos.
- Melhoria: em partições pequenas (5 a 20 elementos), utiliza-se uma das funções quadráticas (Selection, Insertion).
- Pior caso: vetor ordenado.
  - Sempre escolhe o pior pivô (subdivisão ineficiente).
- Melhor pivô é o que consegue subdividir o vetor na metade.

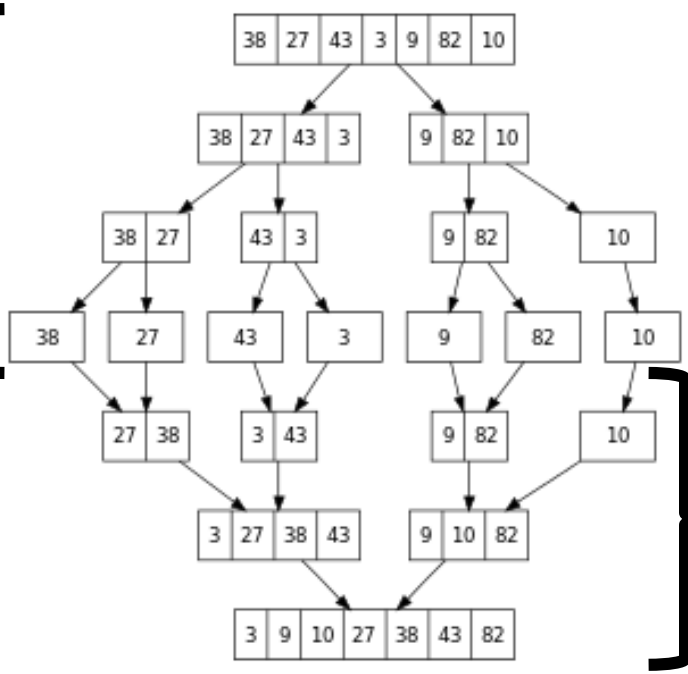
MergeSort

- Baseado no paradigma “Dividir e Conquistar”:
- Procedimento recursivo.
- Utiliza a divisão e intercalação de elementos.

MergeSort

- Baseado no paradigma “Dividir e Conquistar”:
- Procedimento recursivo.
- Utiliza a divisão e intercalação de elementos.

DIVIDIR



INTERCALAR

MergeSort

- Utiliza dois procedimentos:

MergeSort

- Utiliza dois procedimentos:
  - MERGE (intercalar): cria dois vetores, cada um correspondente a uma metade do vetor original. Intercala os valores no vetor original.

MergeSort

- Utiliza dois procedimentos:
  - MERGE (intercalar): cria dois vetores, cada um correspondente a uma metade do vetor original. Intercala os valores no vetor original.
  - MERGE-SORT (ordenação): divide o vetor ao meio, recursivamente, até que o os subvetores sejam elementos unitários. Ao fim da chamada recursiva intercala os subvetores utilizando o MERGE (informa índices de início, meio e fim).

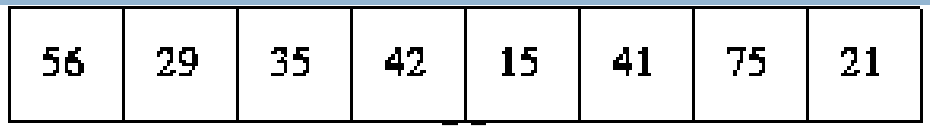
MergeSort

- Utiliza dois procedimentos:
  - MERGE (intercalar): cria dois vetores, cada um correspondente a uma metade do vetor original. Intercala os valores no vetor original.
  - MERGE-SORT (ordenação): divide o vetor ao meio, recursivamente, até que o os subvetores sejam elementos unitários. Ao fim da chamada recursiva intercala os subvetores utilizando o MERGE (informa índices de início, meio e fim).
- Após retornar do último processo recursivo, o vetor está ordenado.

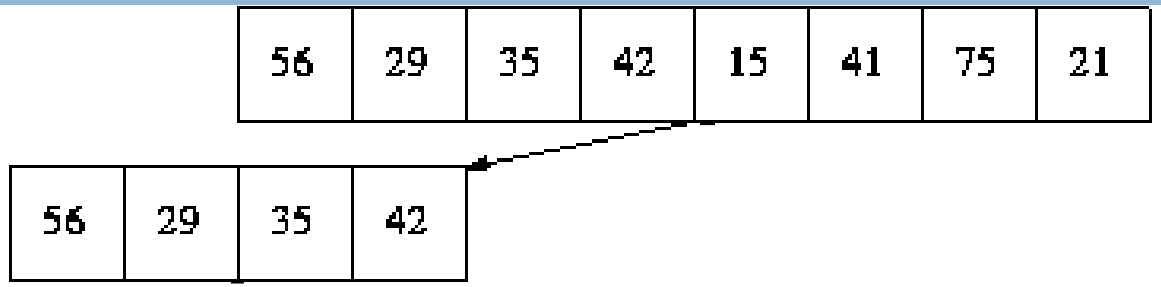
MergeSort

- Utiliza dois procedimentos:
  - MERGE (intercalar): cria dois vetores, cada um correspondente a uma metade do vetor original. Intercala os valores no vetor original.
  - MERGE-SORT (ordenação): divide o vetor ao meio, recursivamente, até que o os subvetores sejam elementos unitários. Ao fim da chamada recursiva intercala os subvetores utilizando o MERGE (informa índices de início, meio e fim).
- Após retornar do último processo recursivo, o vetor está ordenado.

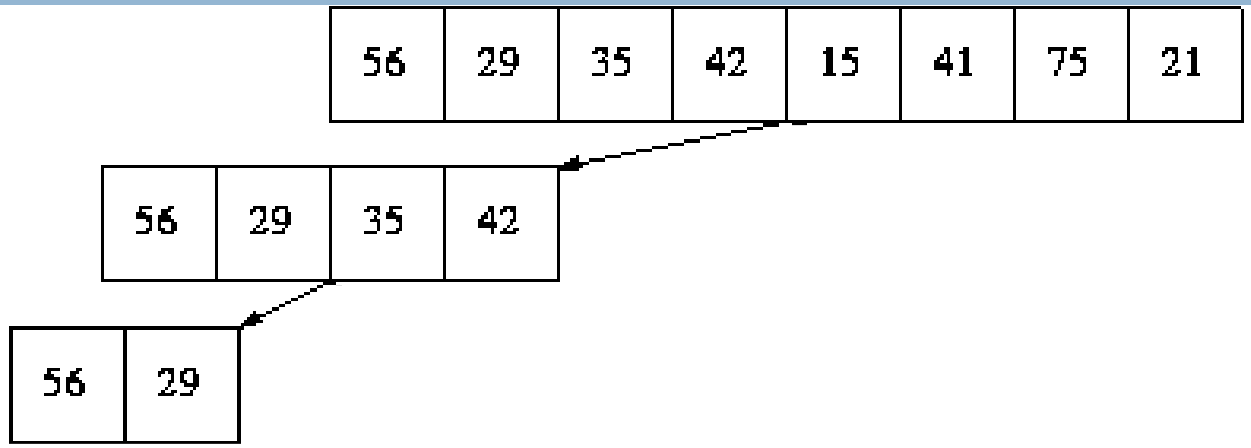
MergeSort



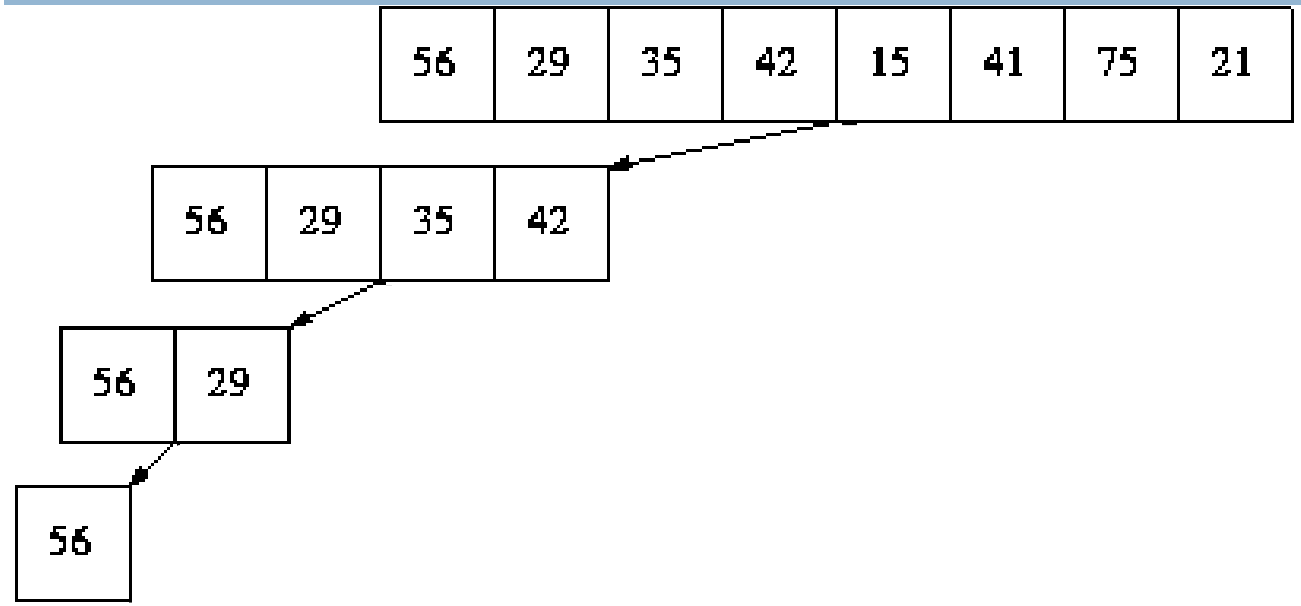
MergeSort



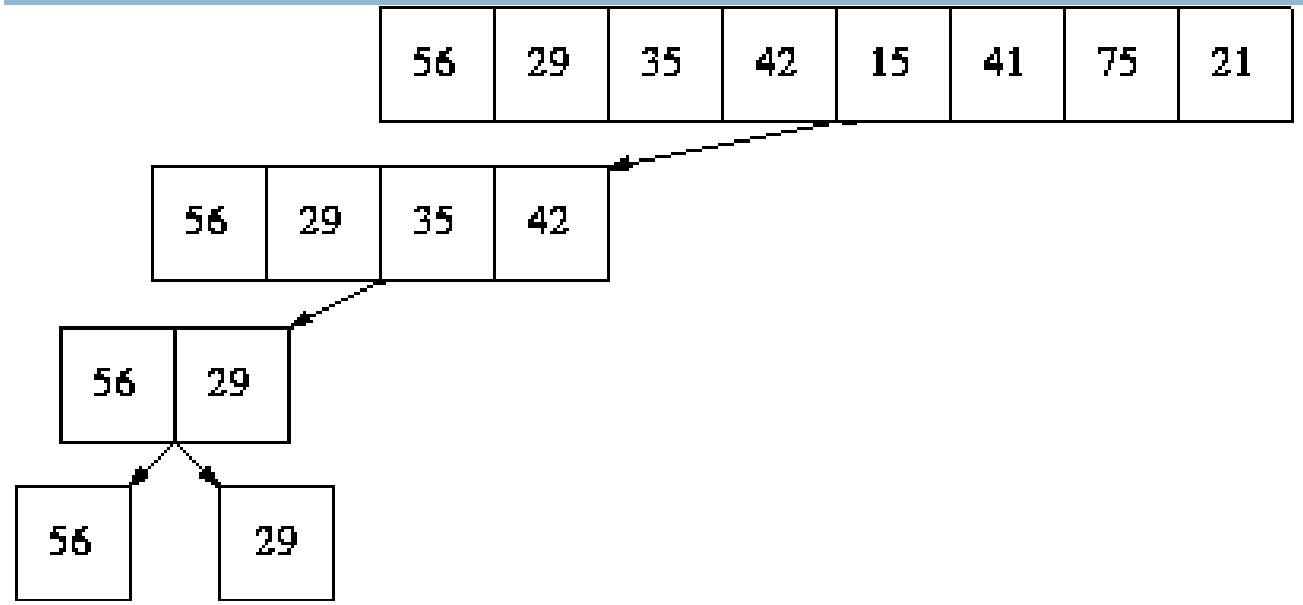
MergeSort



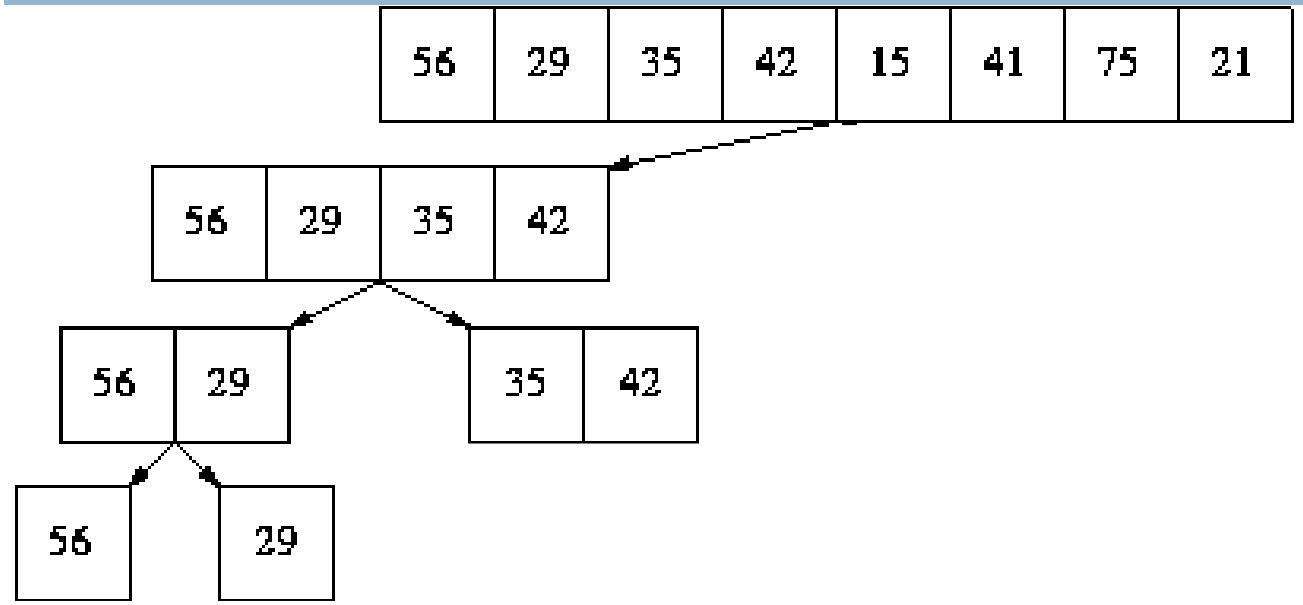
MergeSort



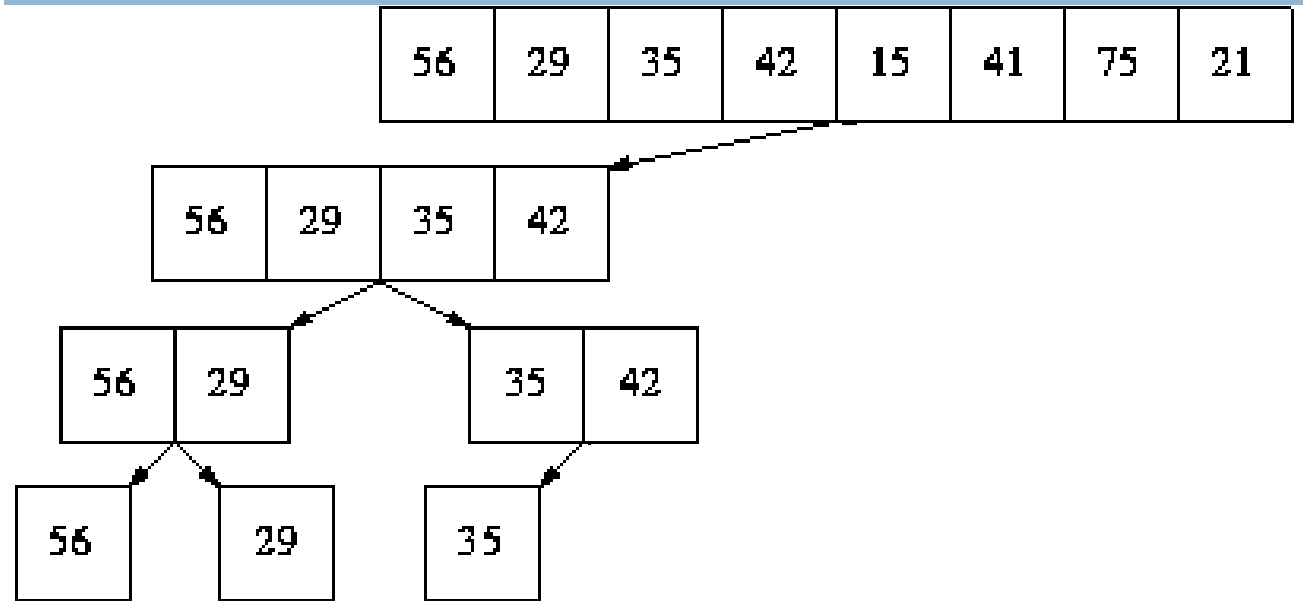
MergeSort



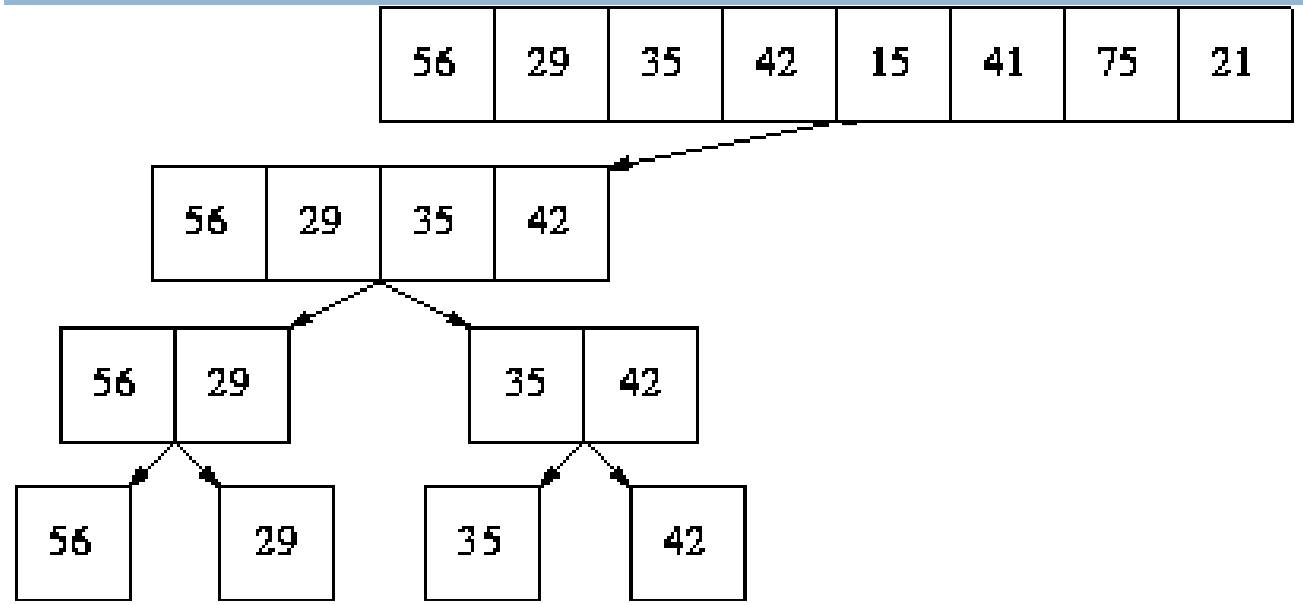
MergeSort



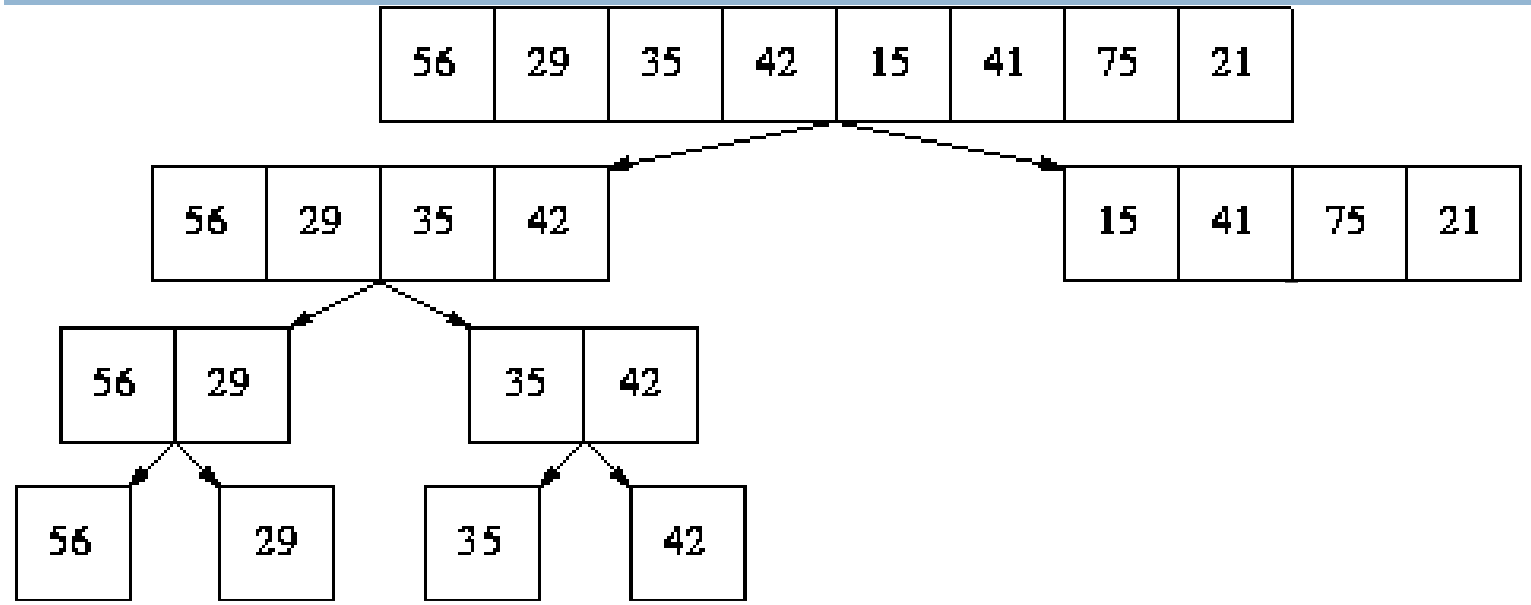
MergeSort



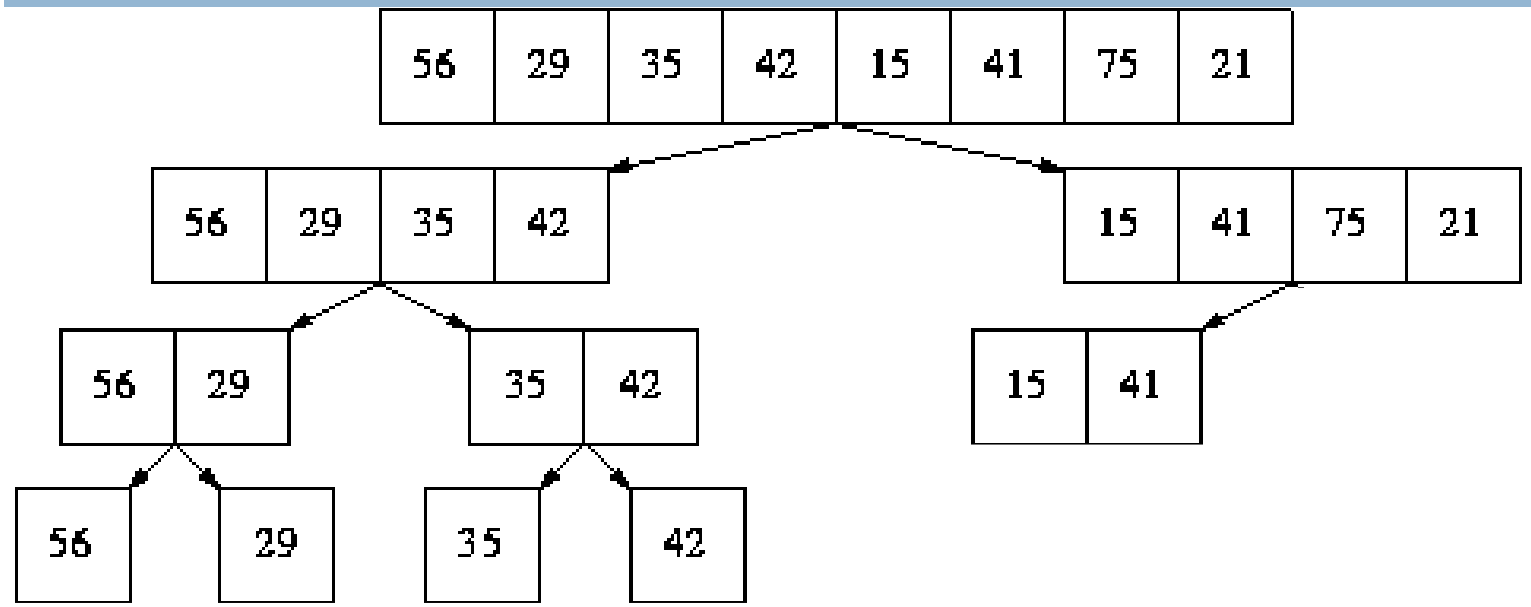
MergeSort



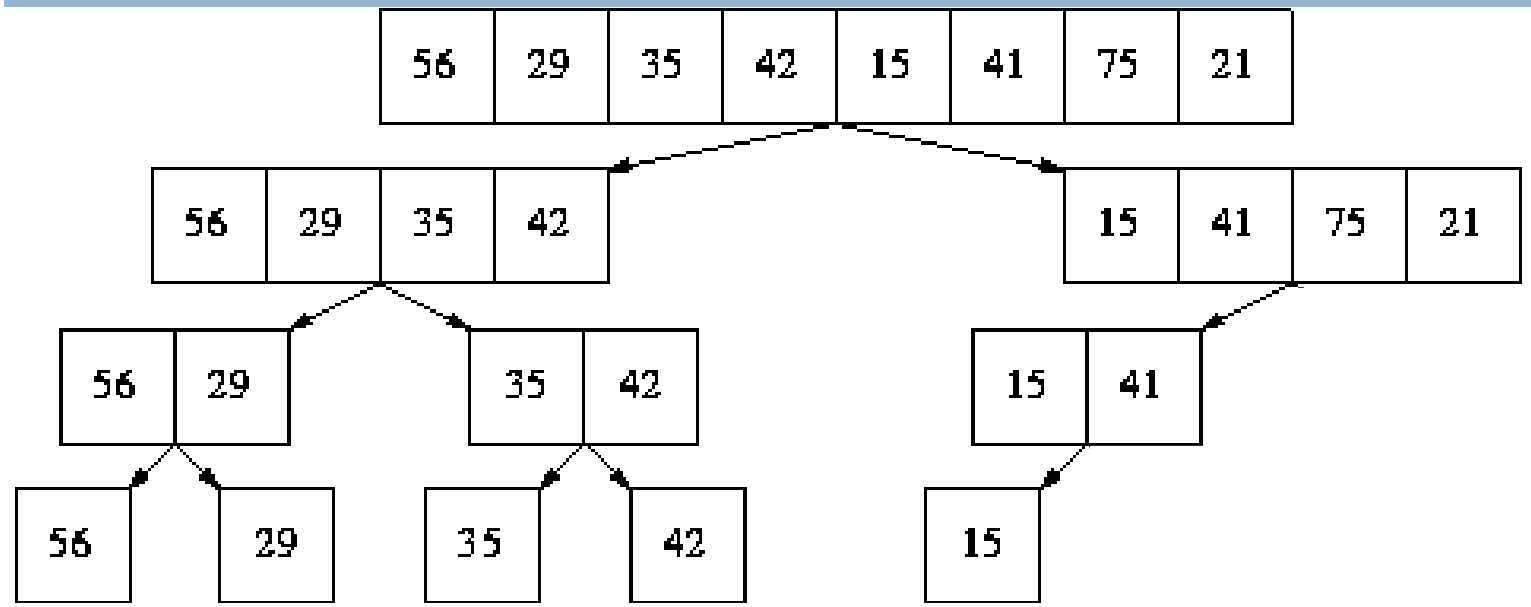
MergeSort



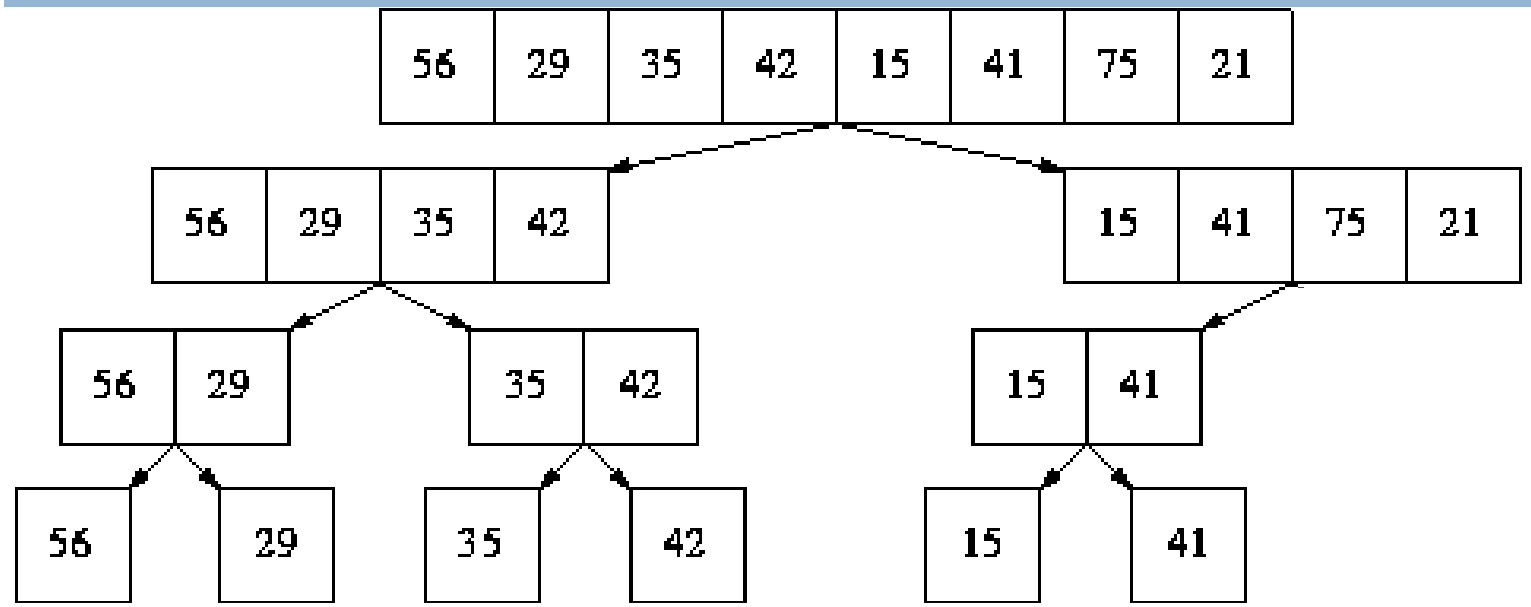
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
56
29
35
42

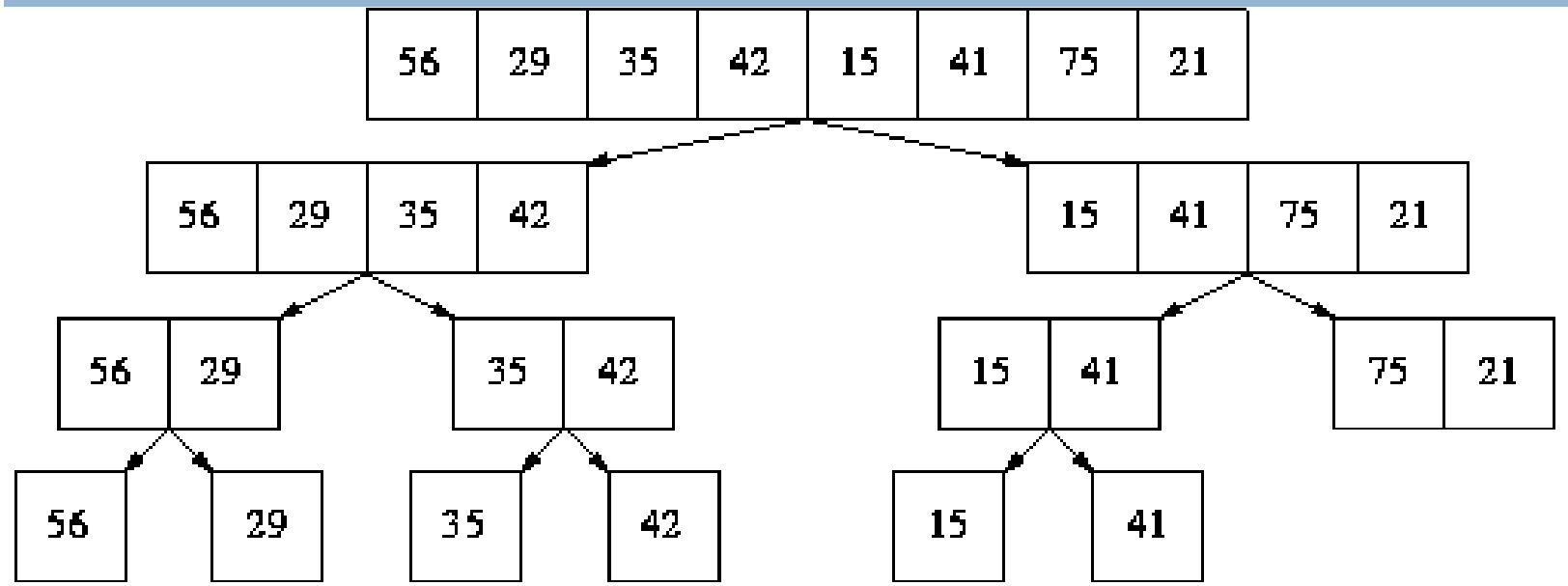
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
56
29
35
42
15

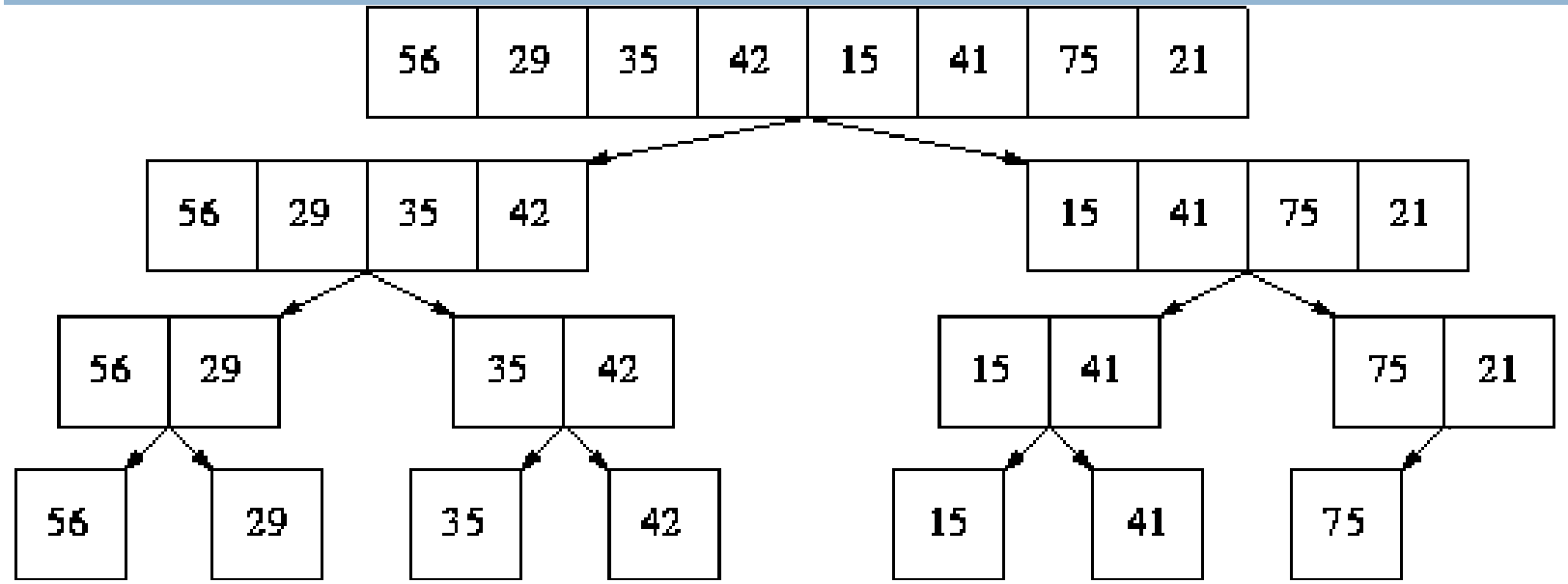
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
56
29
35
42
15
41

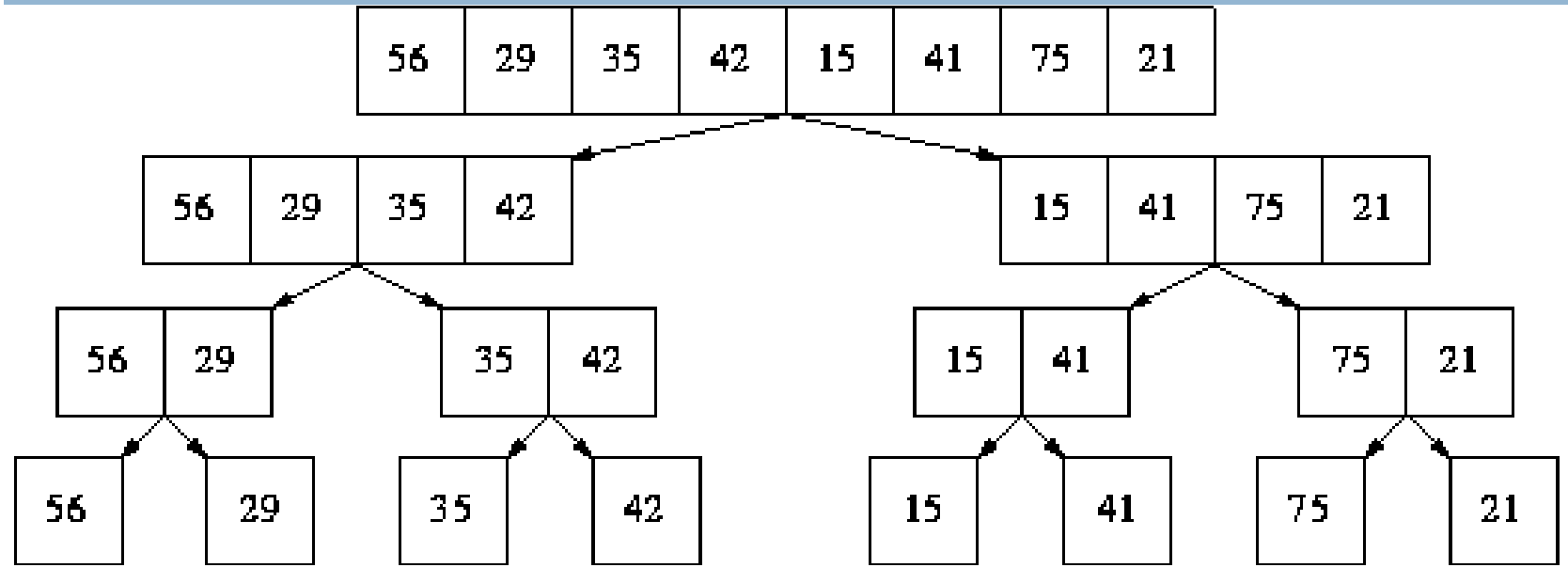
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41

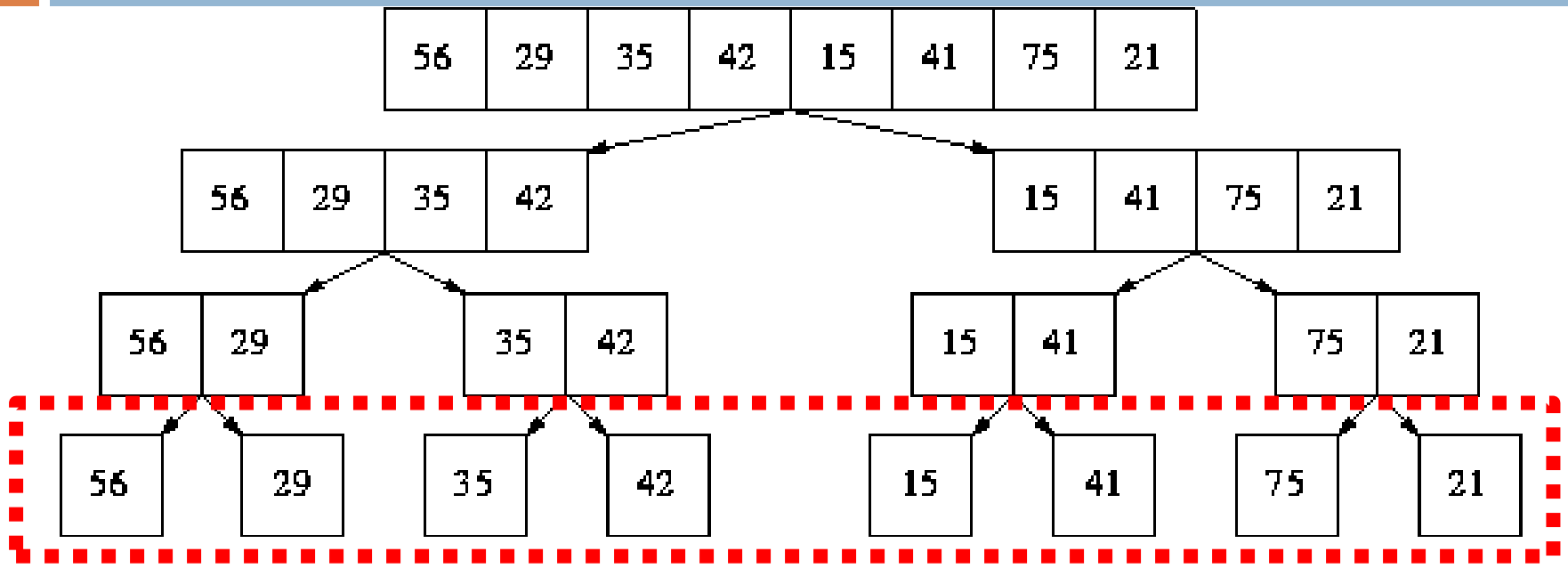
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75

MergeSort

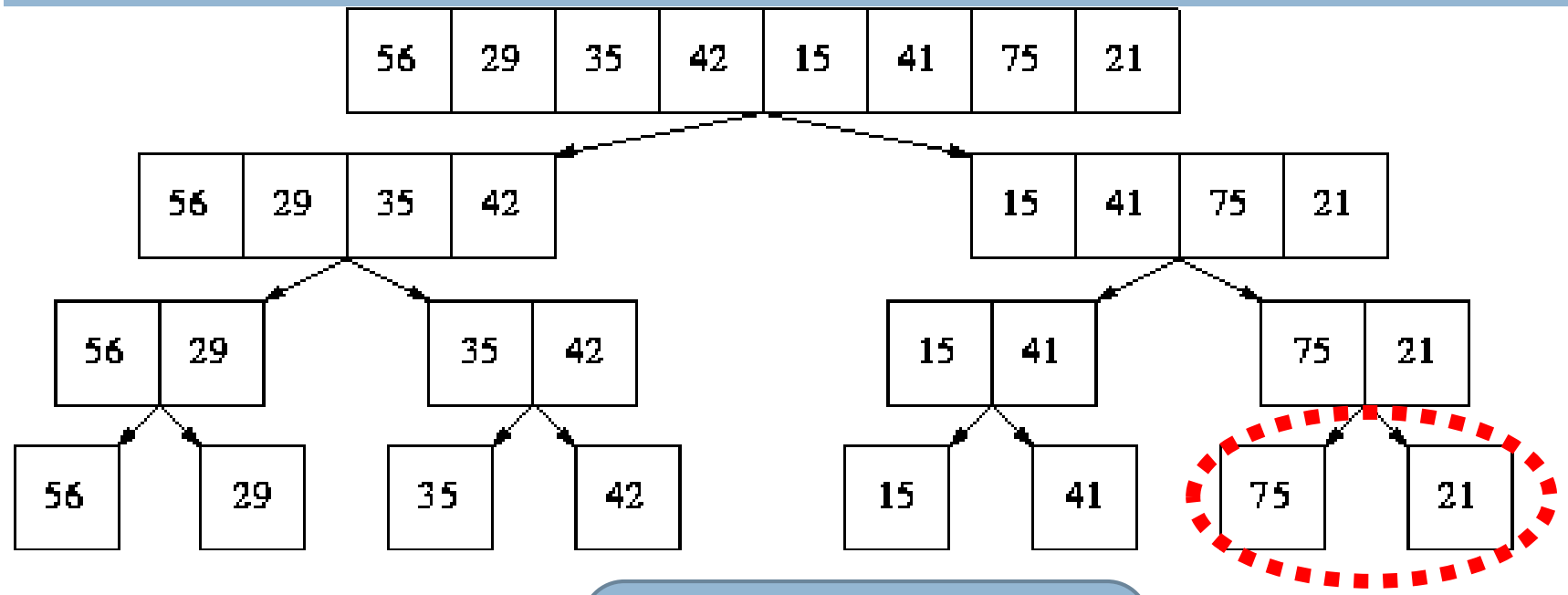

56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21

MergeSort



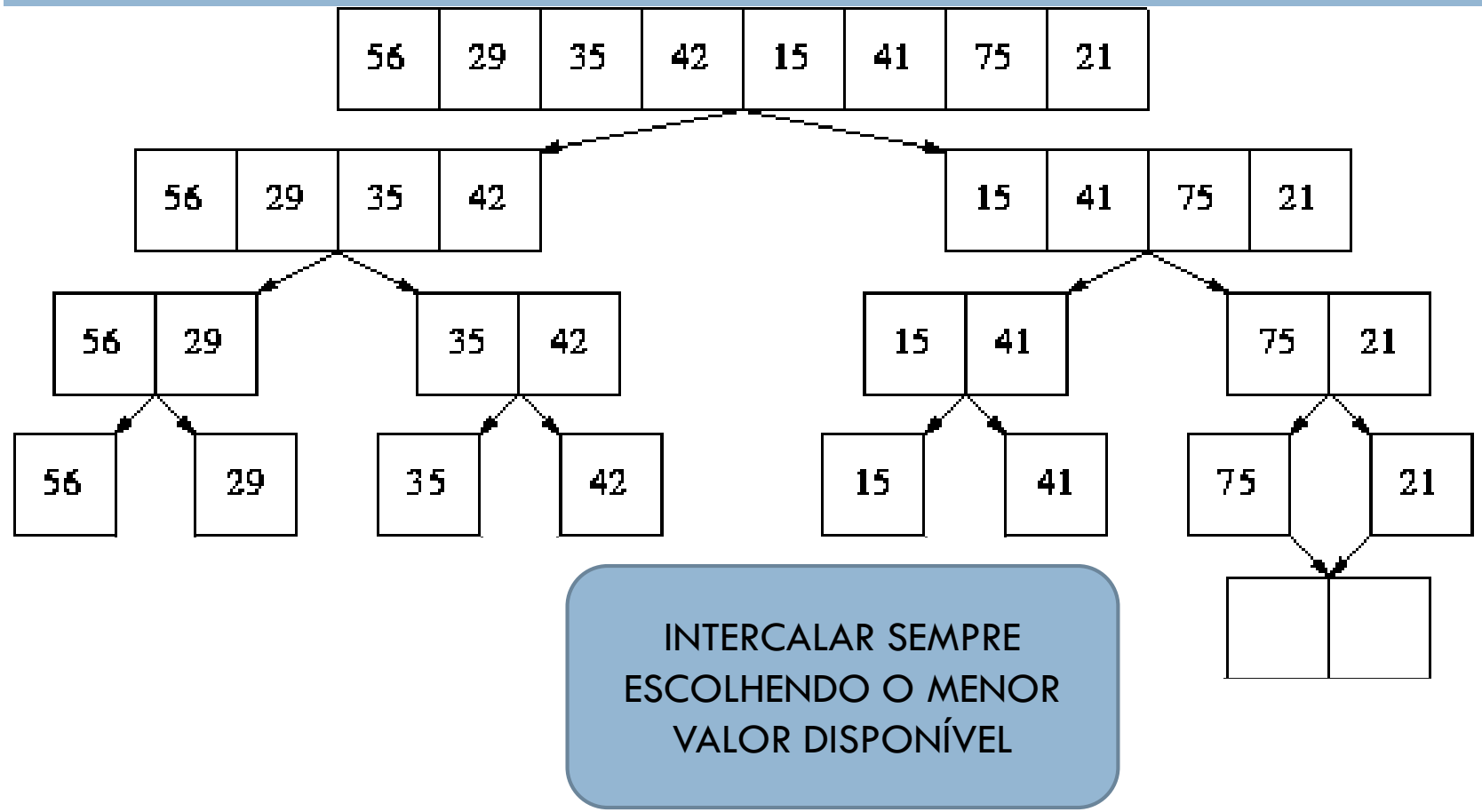
INTERCALAR VETORES

MergeSort

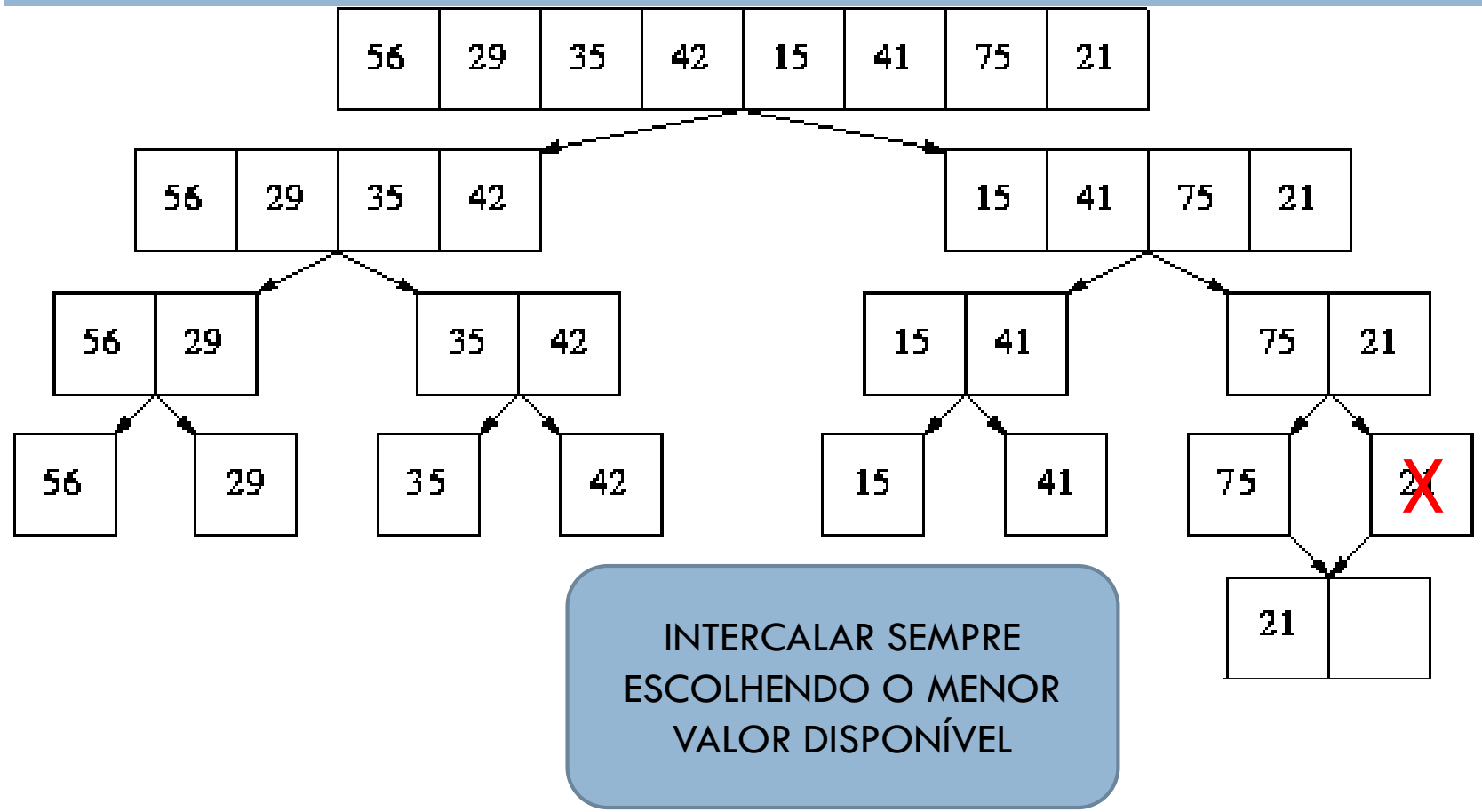


INTERCALAR SEMPRE ESCOLHENDO O MENOR VALOR DISPONÍVEL

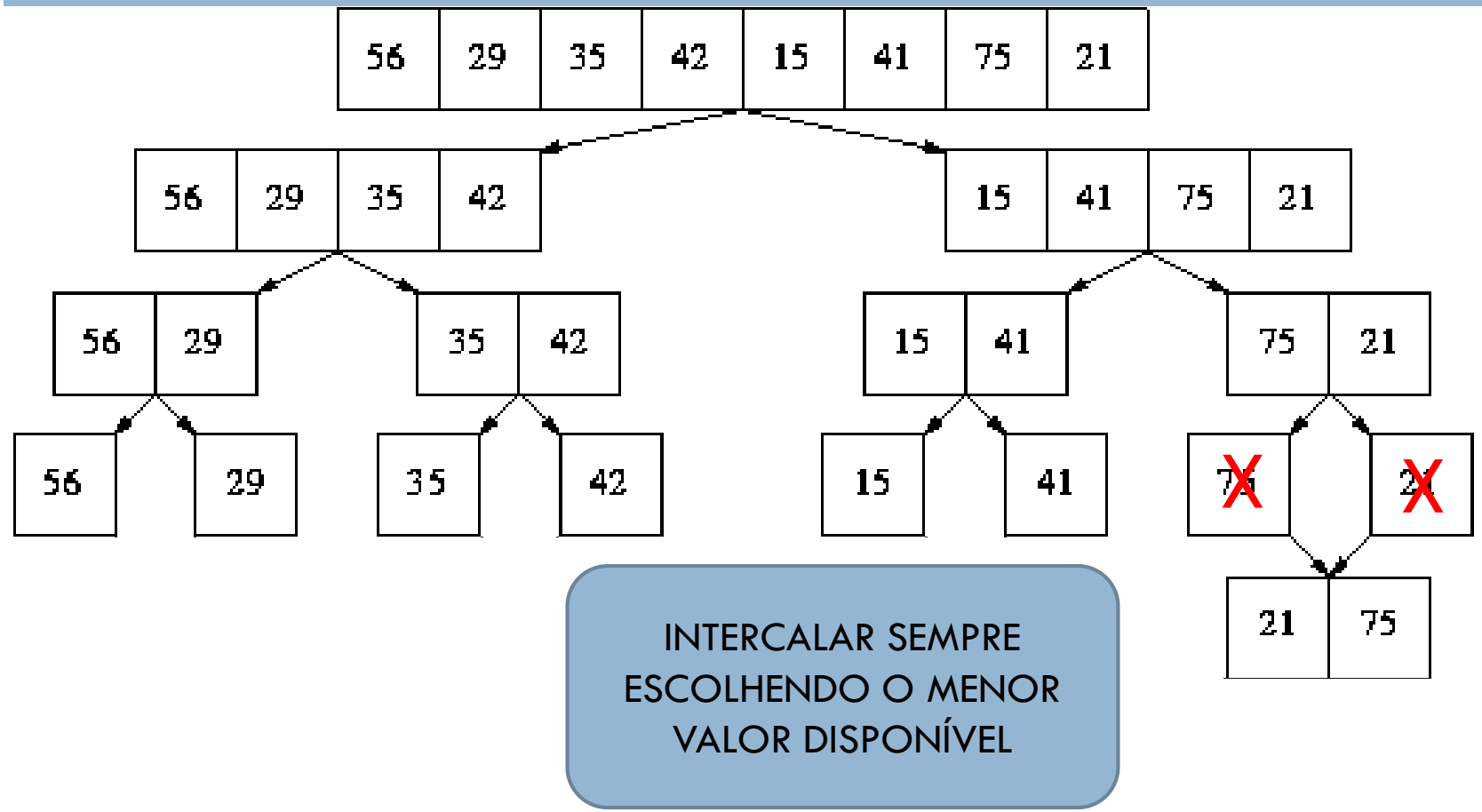
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21
INTERCALAR SEMPRE
ESCOLHENDO O MENOR
VALOR DISPONÍVEL

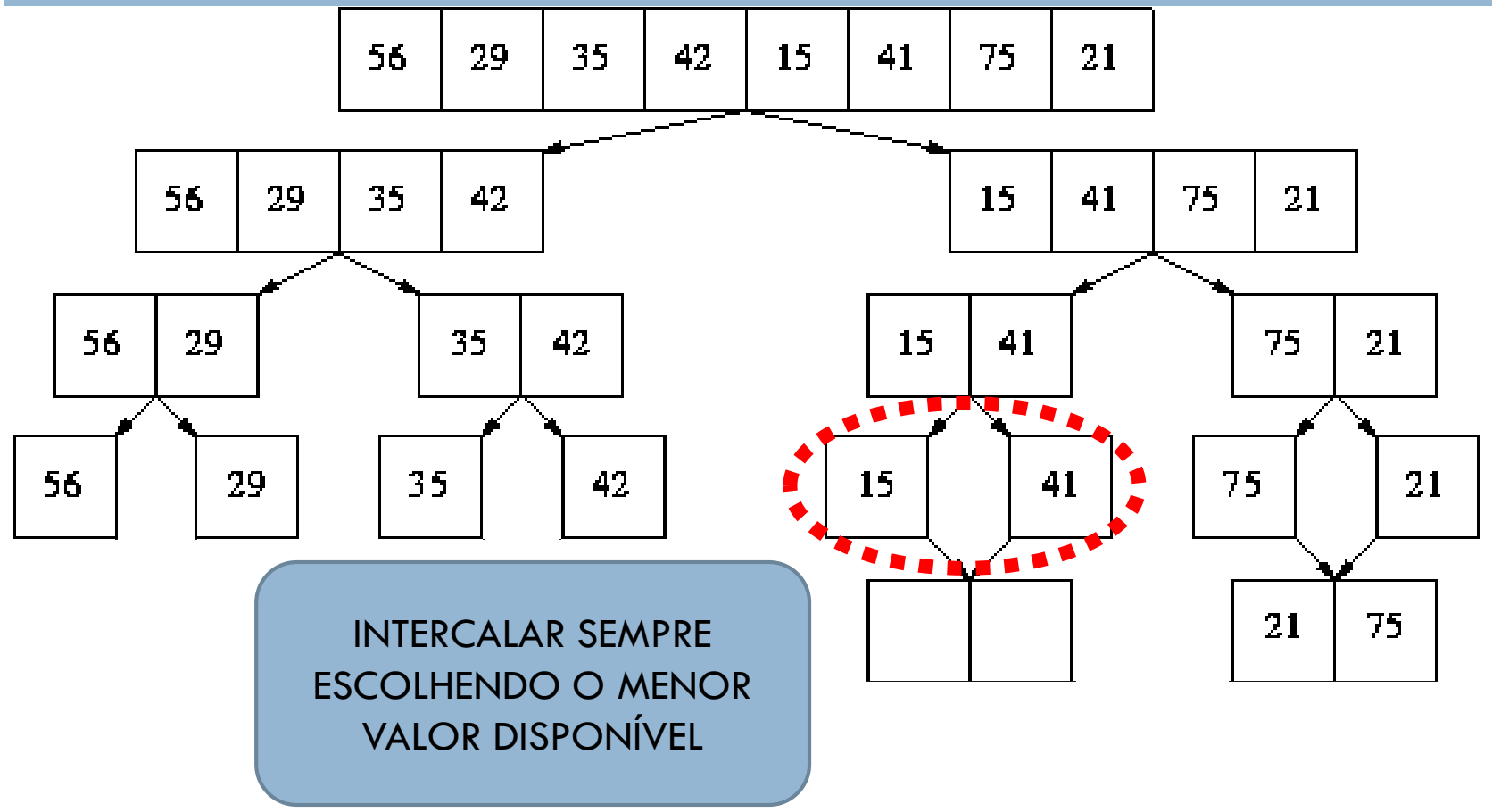
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21
21
INTERCALAR SEMPRE
ESCOLHENDO O MENOR
VALOR DISPONÍVEL

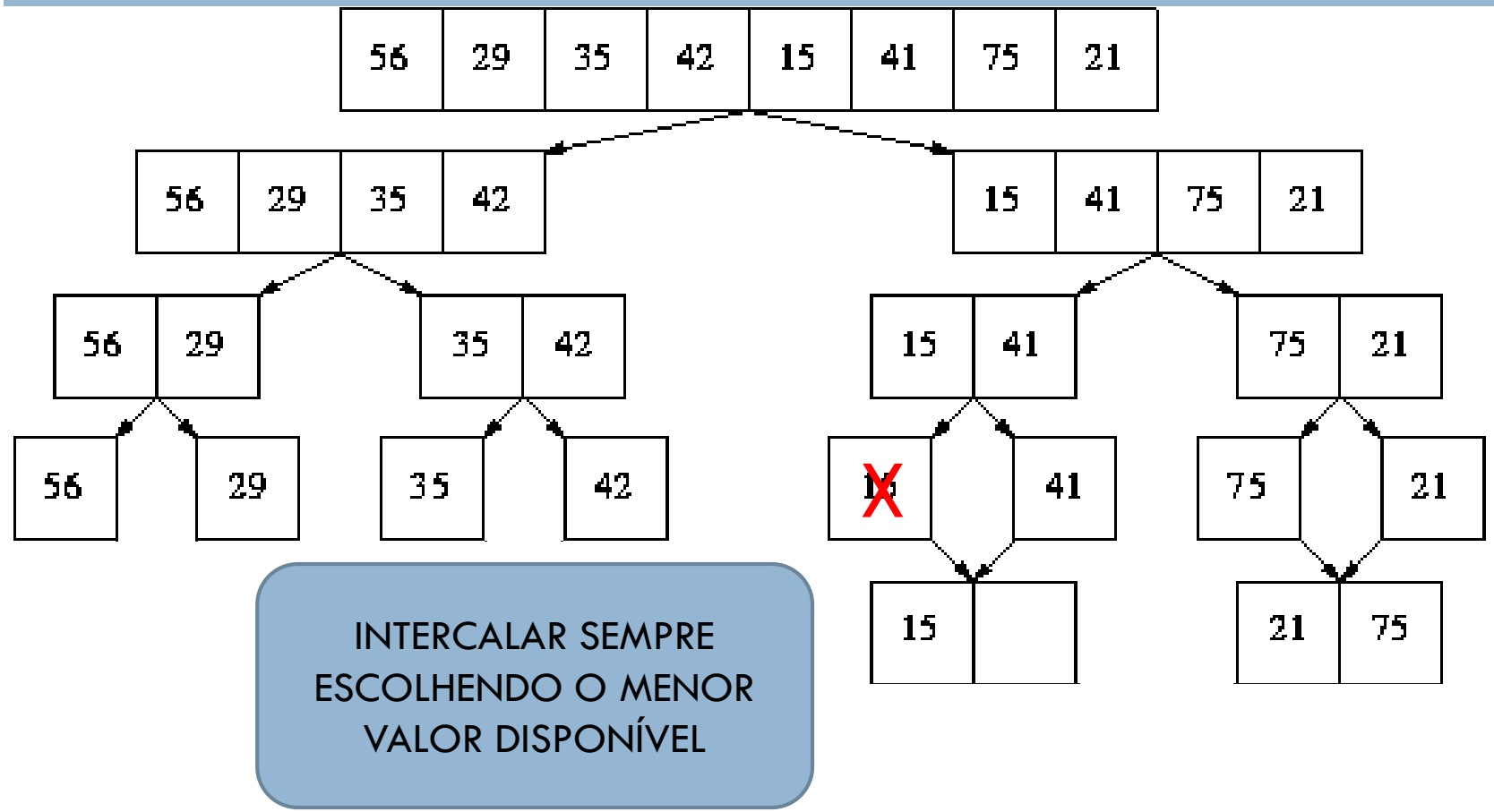
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21
INTERCALAR SEMPRE
ESCOLHENDO O MENOR
VALOR DISPONÍVEL

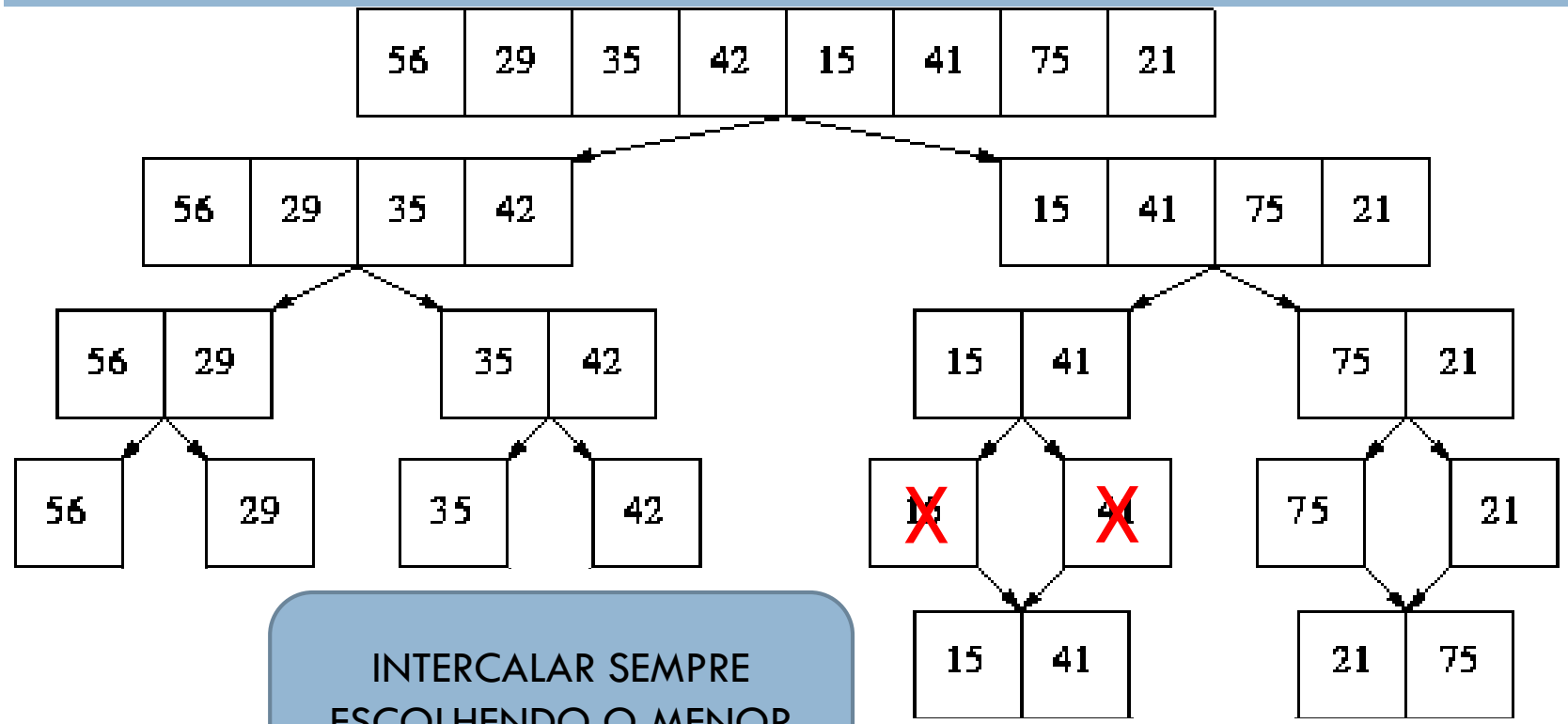
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21
21 75
INTERCALAR SEMPRE
ESCOLHENDO O MENOR
VALOR DISPONÍVEL

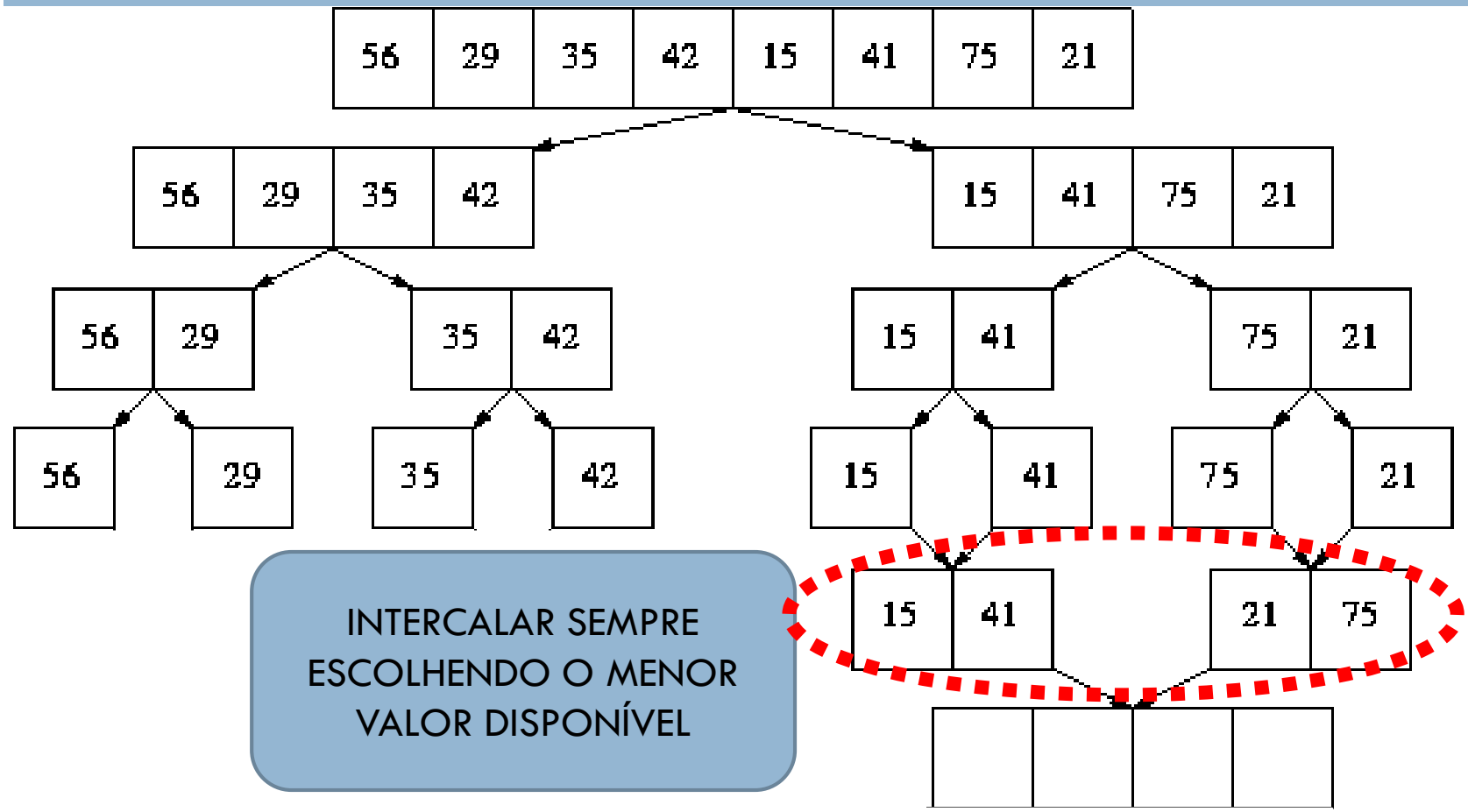
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21
15
21 75
INTERCALAR SEMPRE ESCOLHENDO O MENOR VALOR DISPONÍVEL

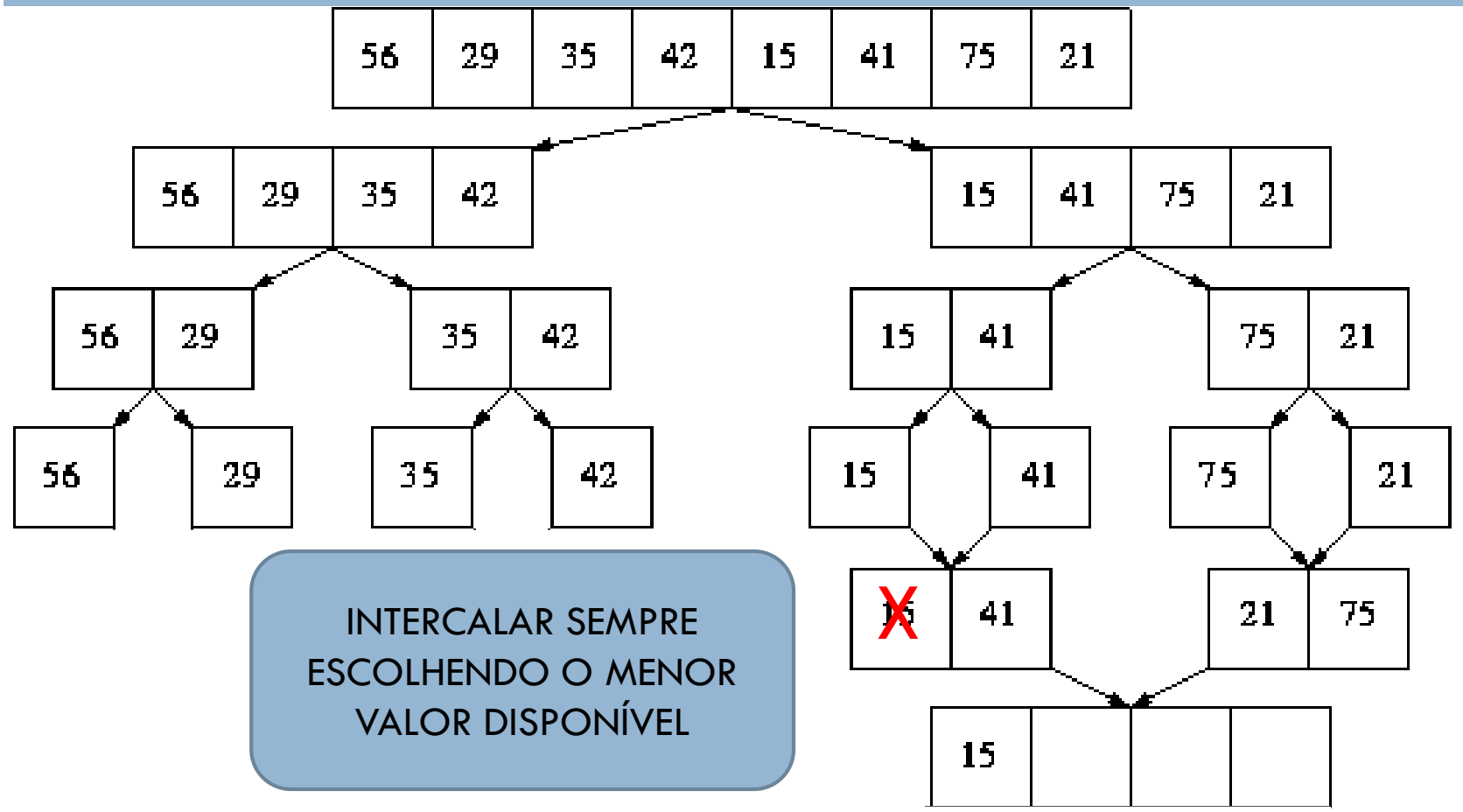
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21
15 41
21 75
INTERCALAR SEMPRE ESCOLHENDO O MENOR VALOR DISPONÍVEL

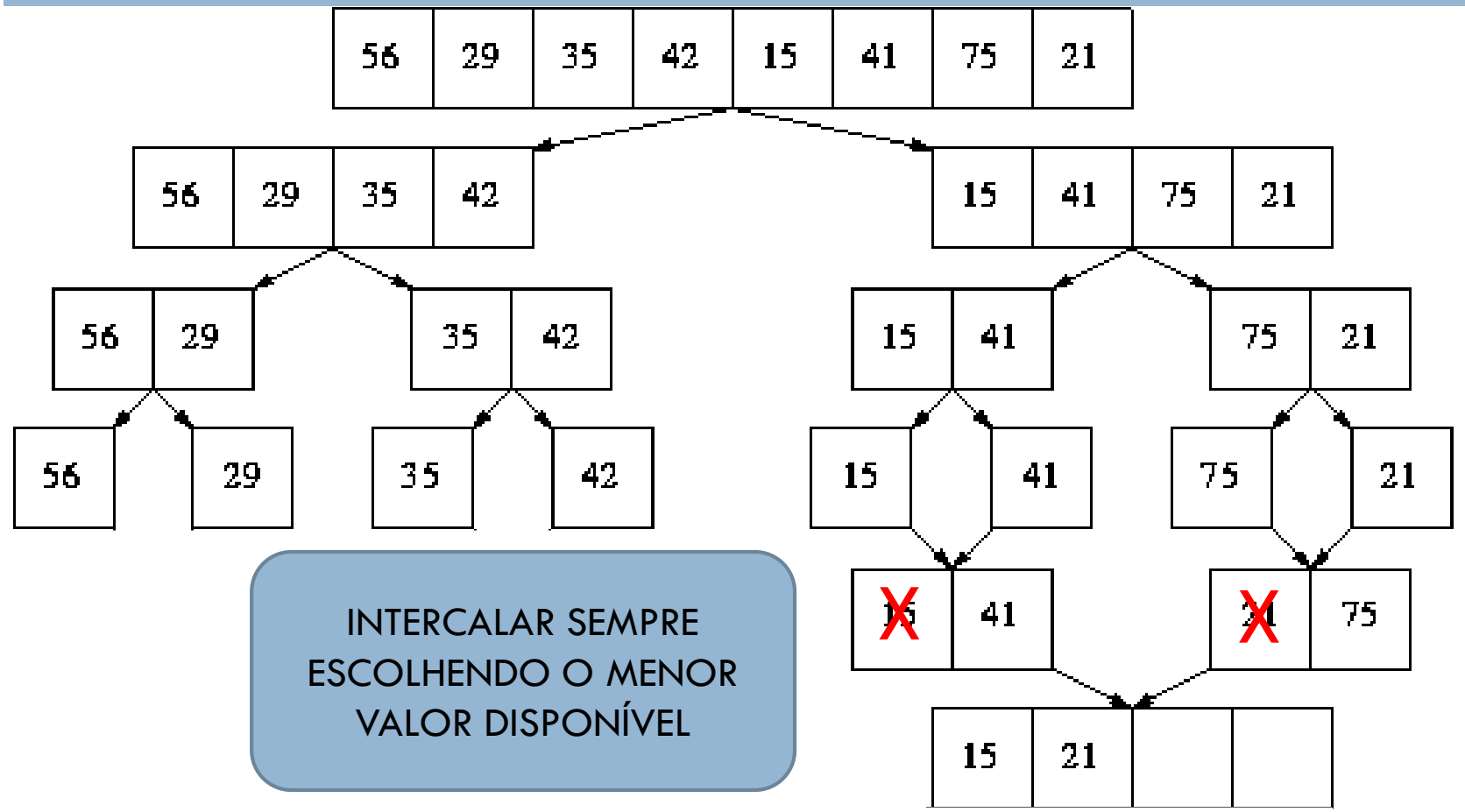
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21
15 41
21 75
INTERCALAR SEMPRE ESCOLHENDO O MENOR VALOR DISPONÍVEL

MergeSort

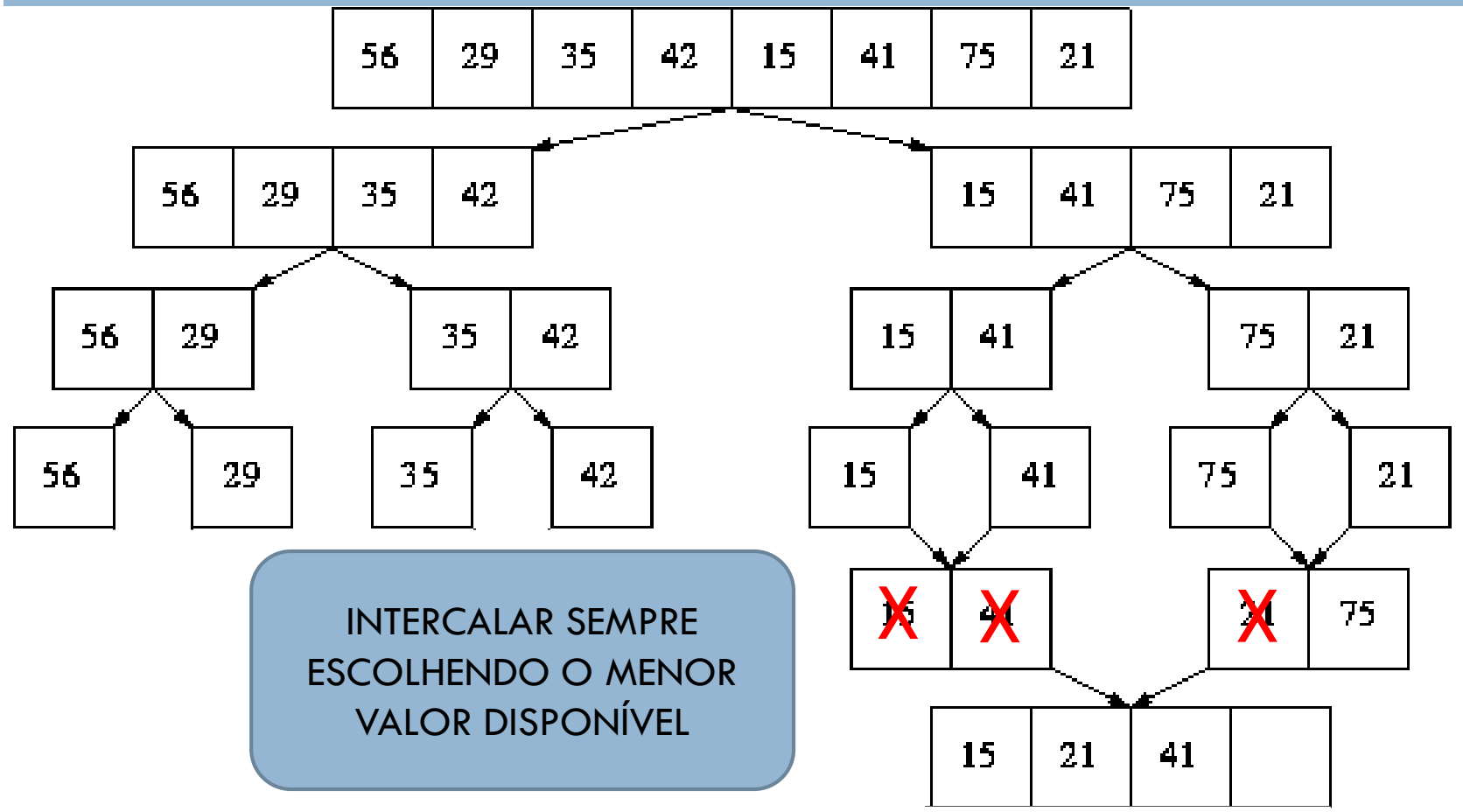

56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21
15
41
21
75
15
INTERCALAR SEMPRE
ESCOLHENDO O MENOR
VALOR DISPONÍVEL

MergeSort



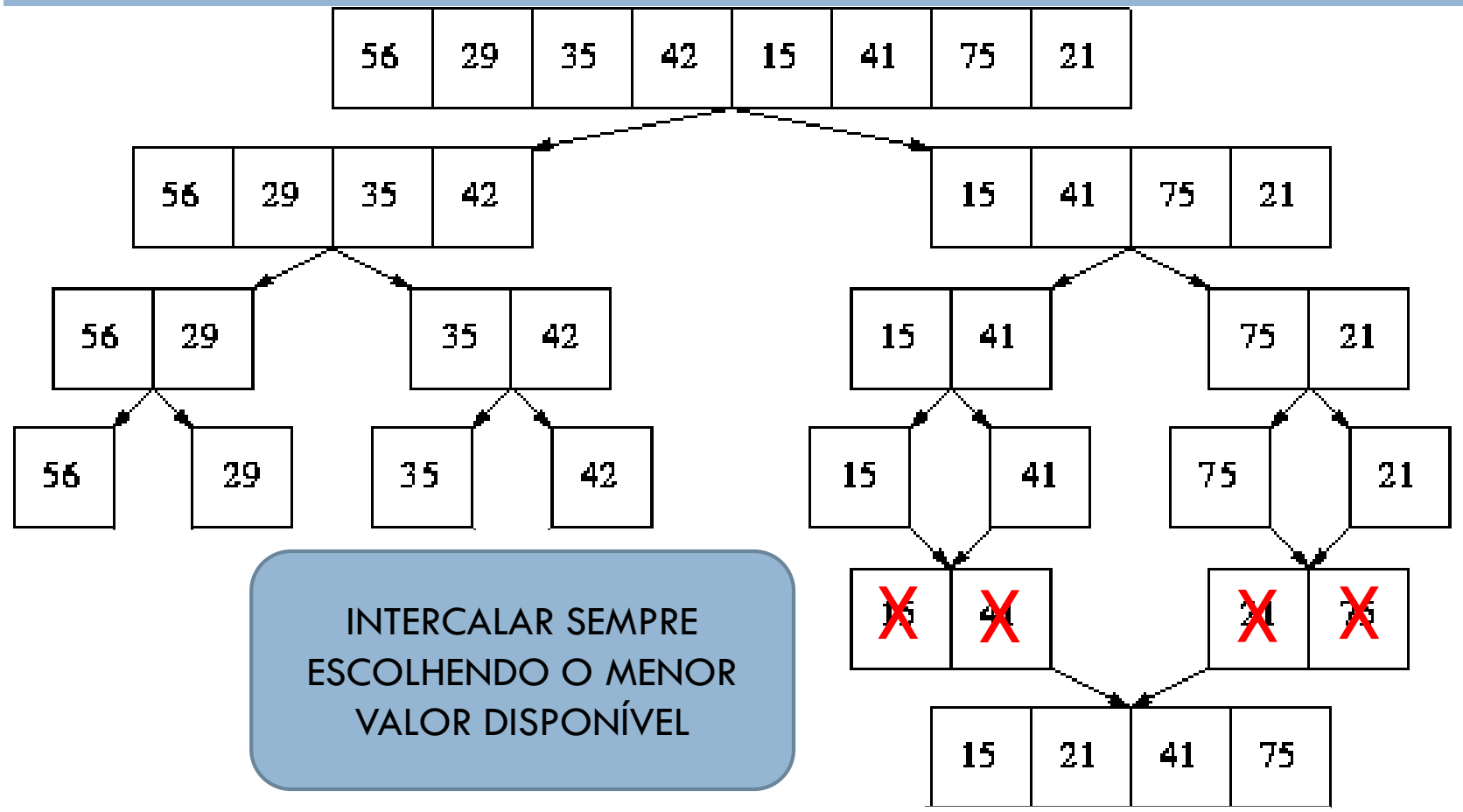
INTERCALAR SEMPRE ESCOLHENDO O MENOR VALOR DISPONÍVEL

MergeSort

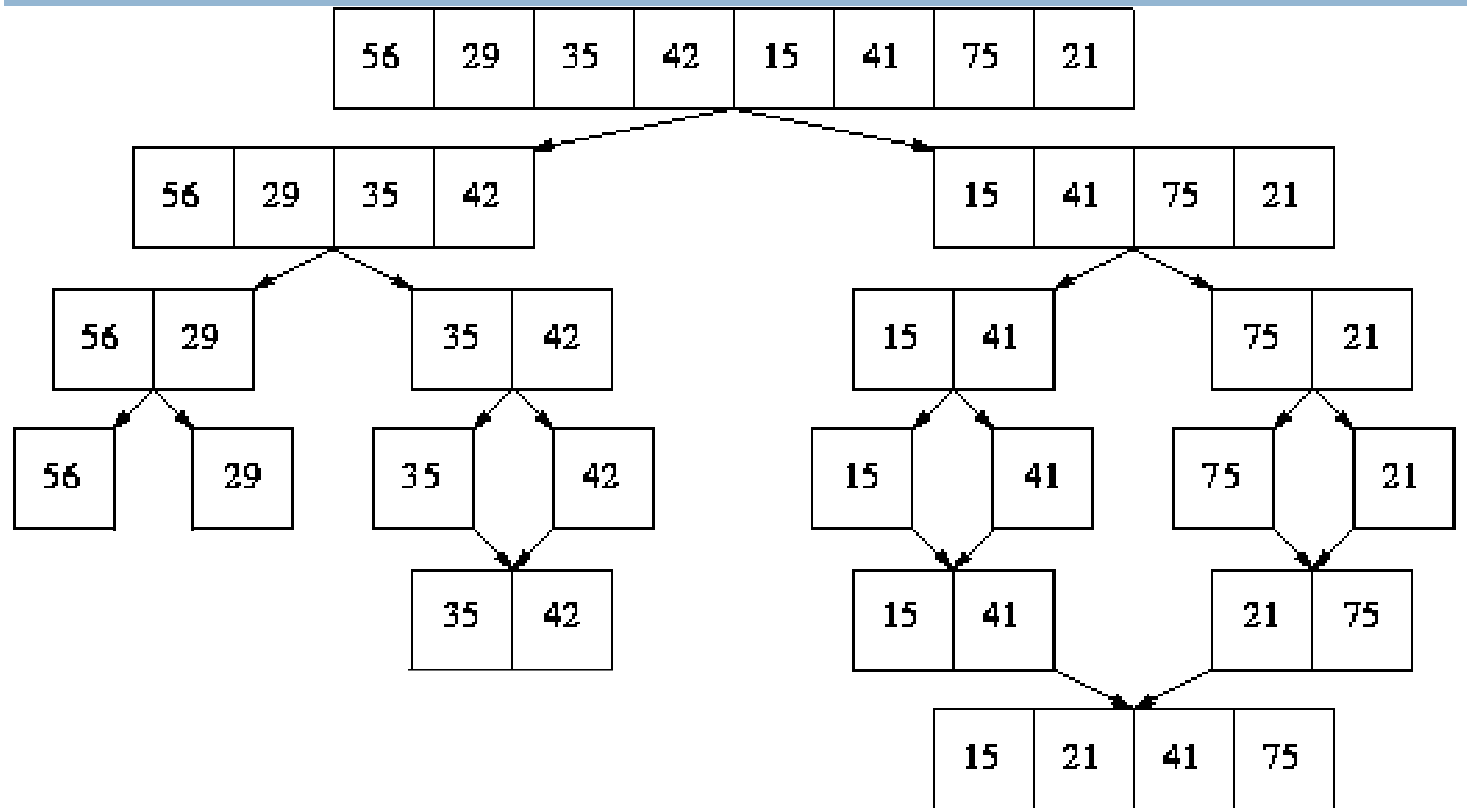


INTERCALAR SEMPRE ESCOLHENDO O MENOR VALOR DISPONÍVEL

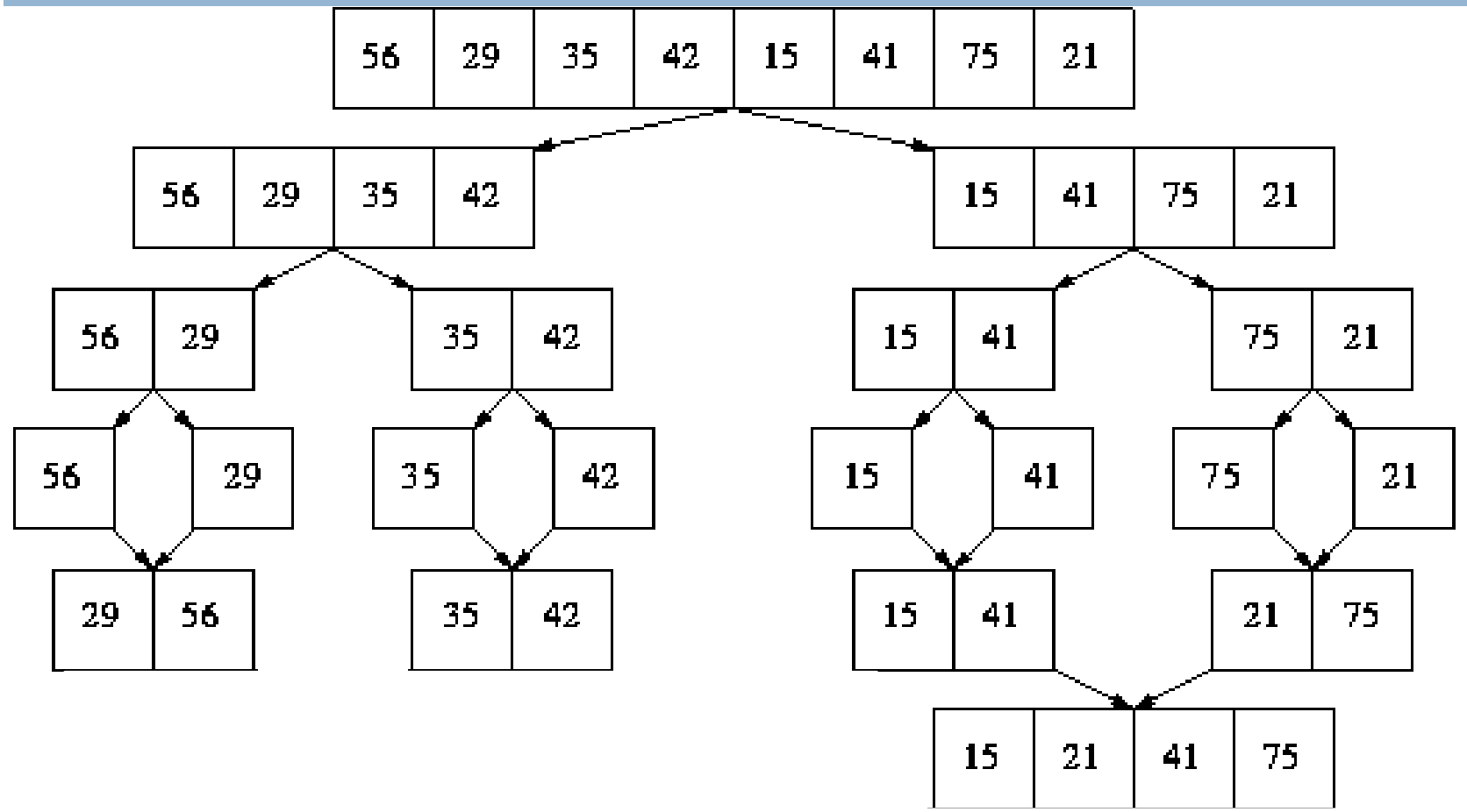
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21
15 21 41 75
INTERCALAR SEMPRE
ESCOLHENDO O MENOR
VALOR DISPONÍVEL

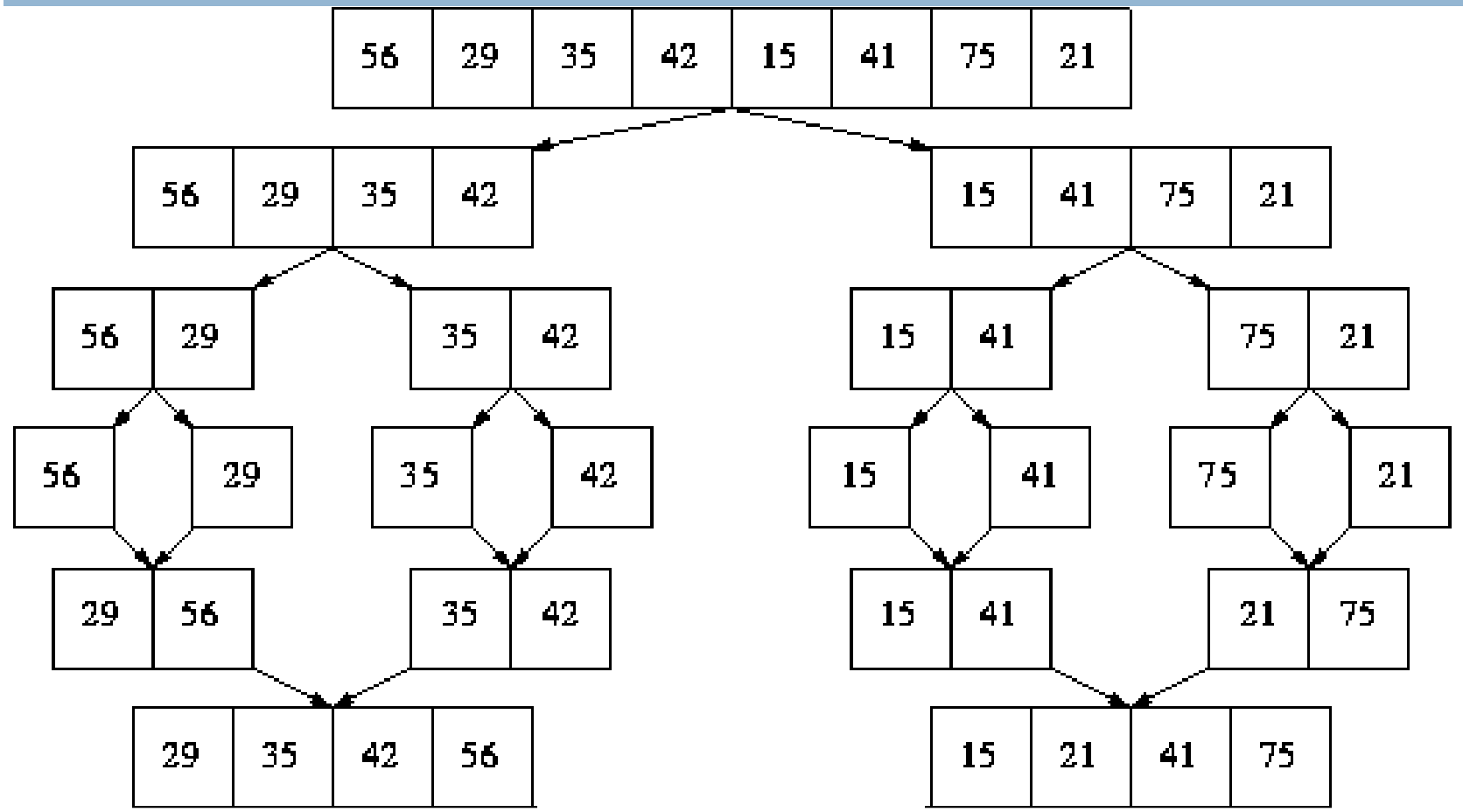
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21
35 42
15 41
21 75
15 21 41 75

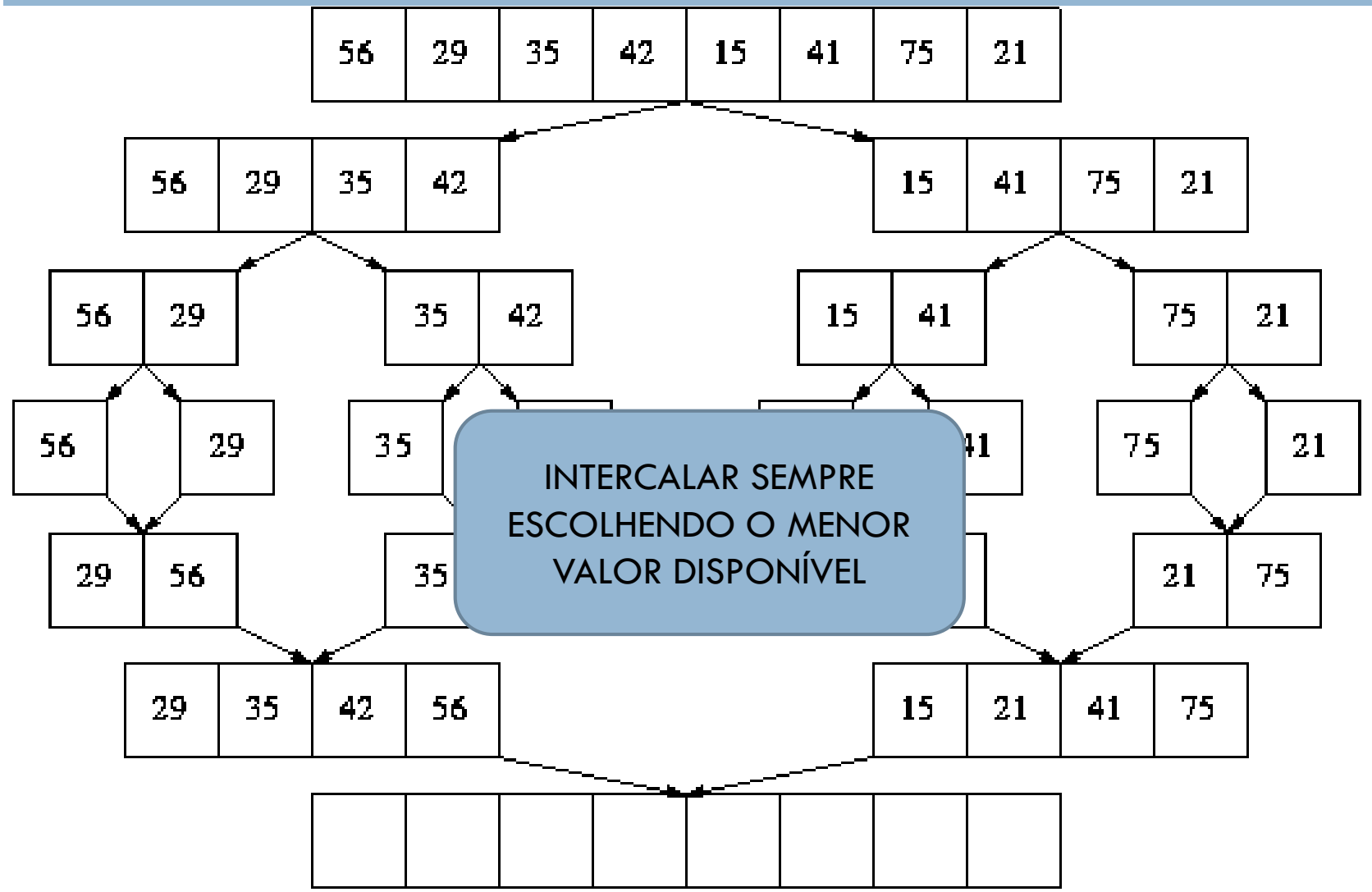
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21
29 56
35 42
15 41
21 75
15 21 41 75

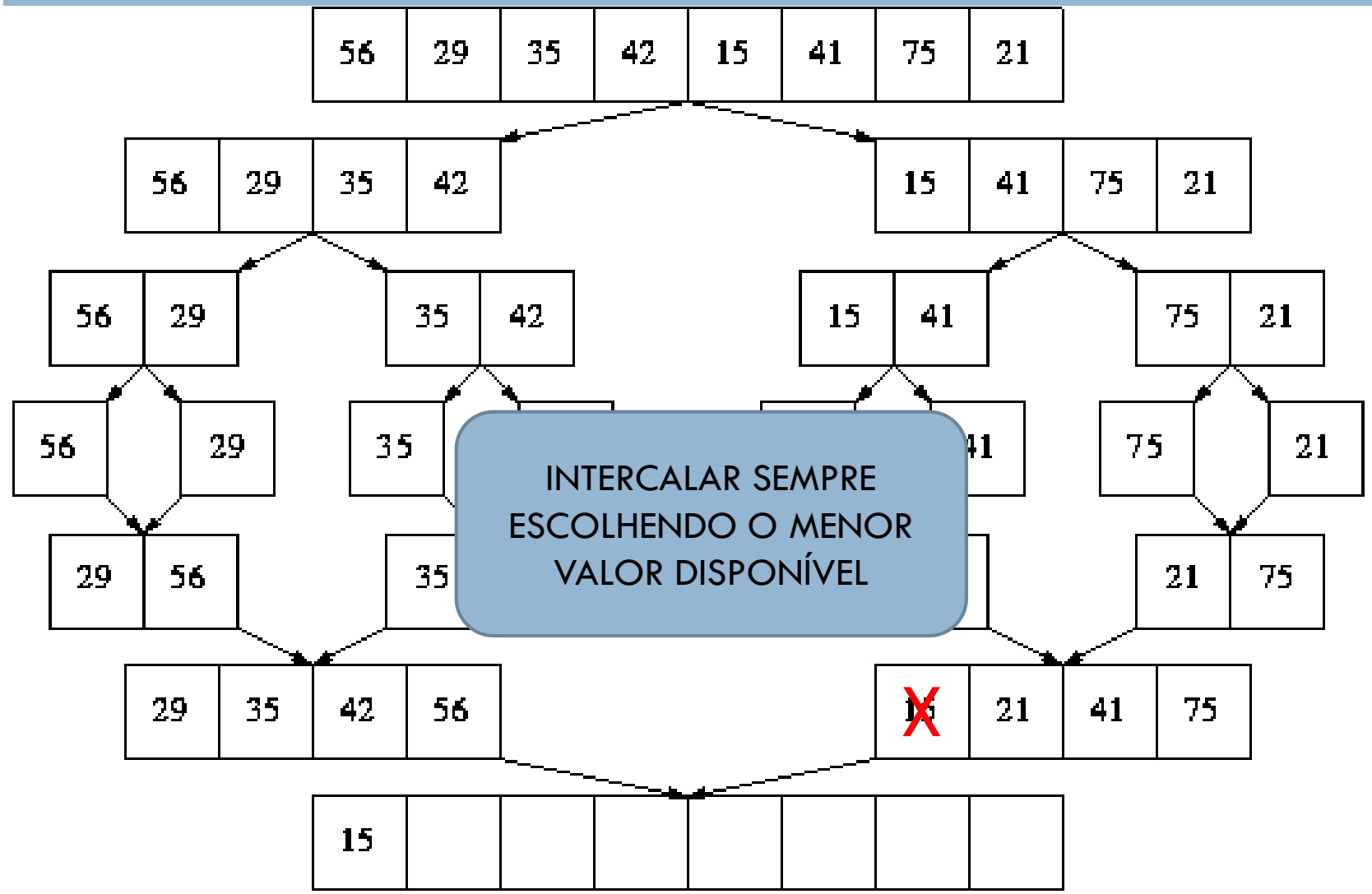
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
35
42
15
41
75
21
29 56
35 42
15 41
21 75
29 35 42 56
15 21 41 75

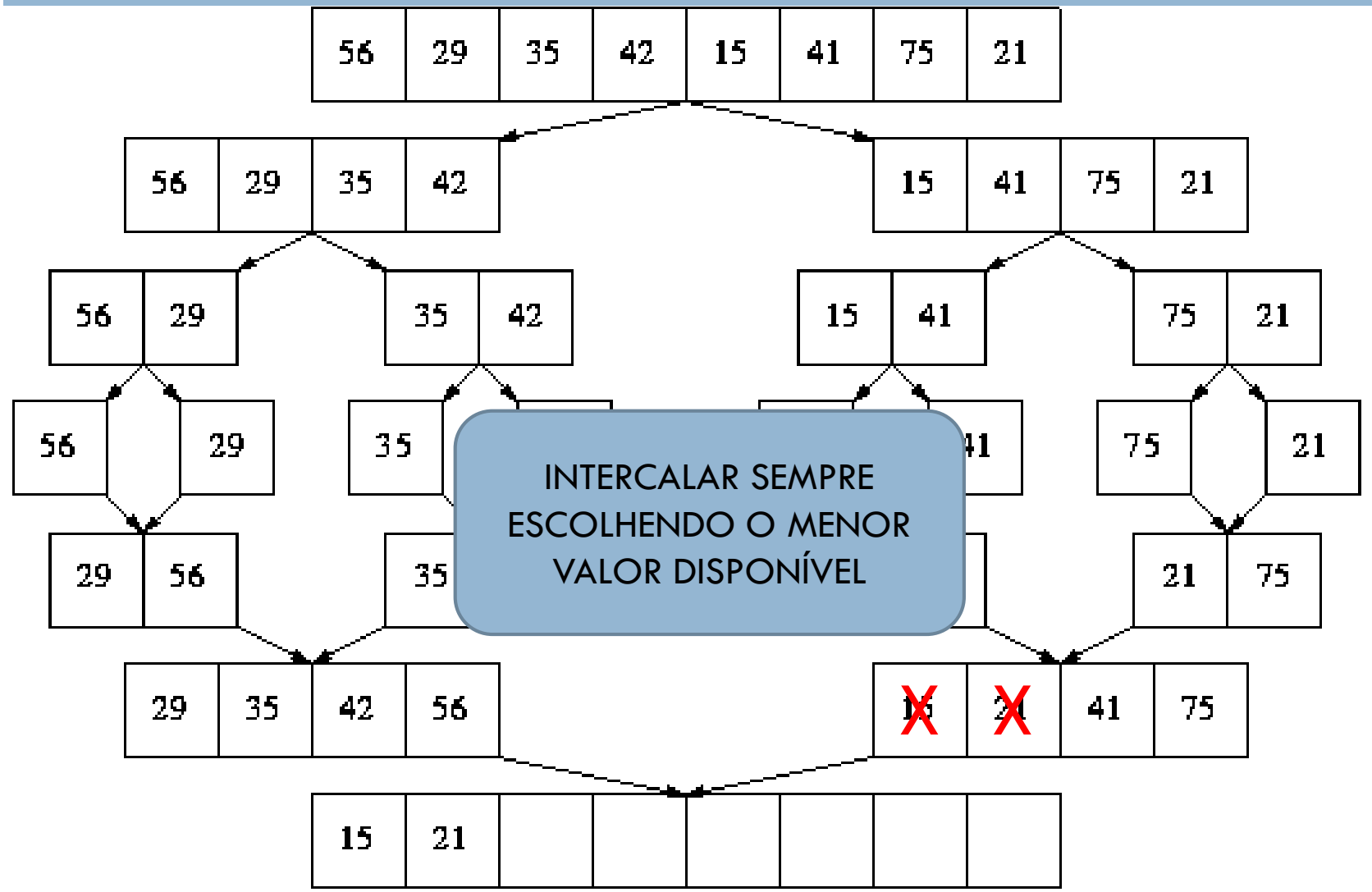
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
75
21
29 56
21 75
29 35 42 56
15 21 41 75
INTERCALAR SEMPRE ESCOLHENDO O MENOR VALOR DISPONÍVEL

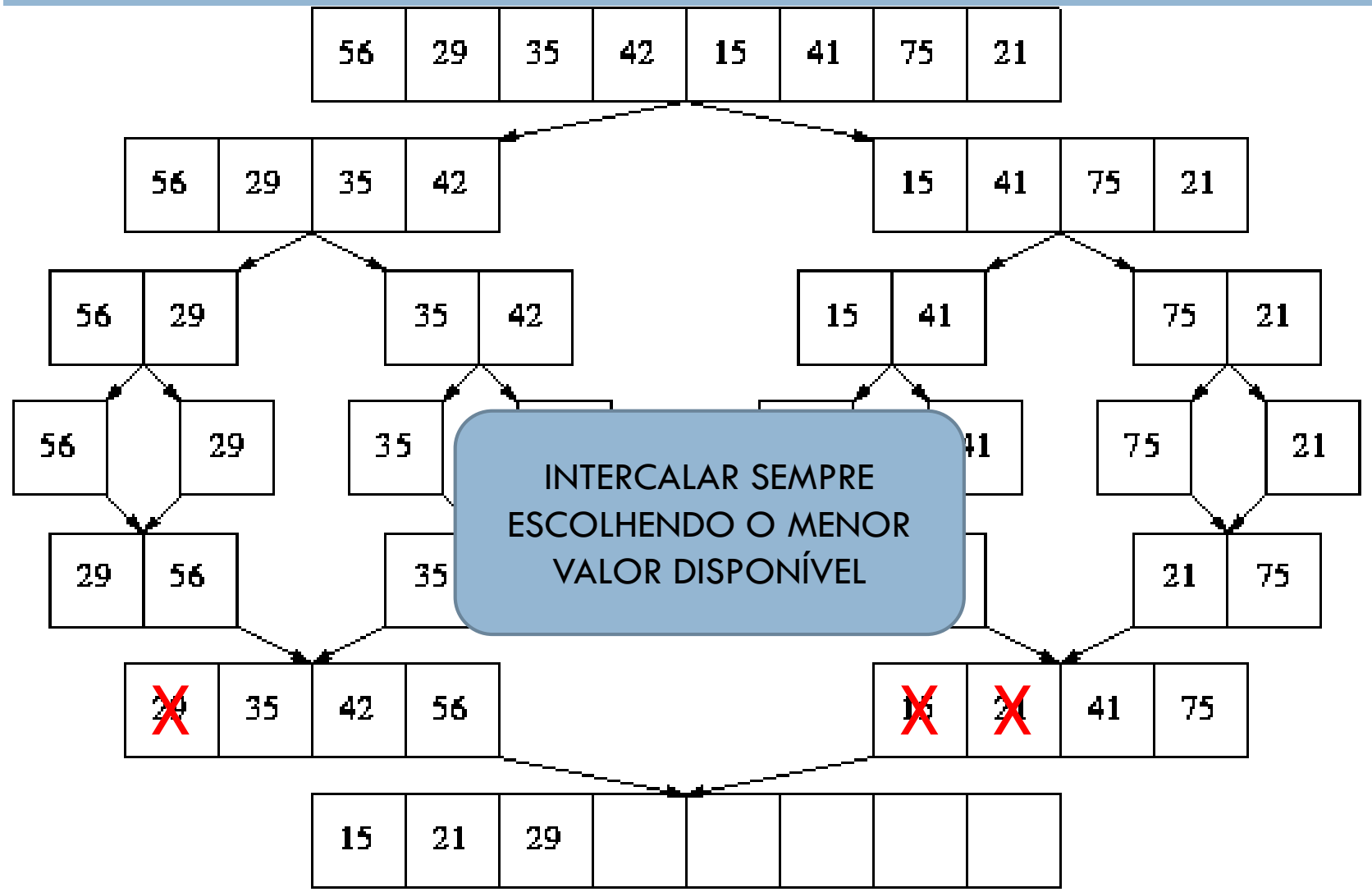
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
75
21
29 56
21 75
29 35 42 56
21 41 75
15
INTERCALAR SEMPRE
ESCOLHENDO O MENOR
VALOR DISPONÍVEL

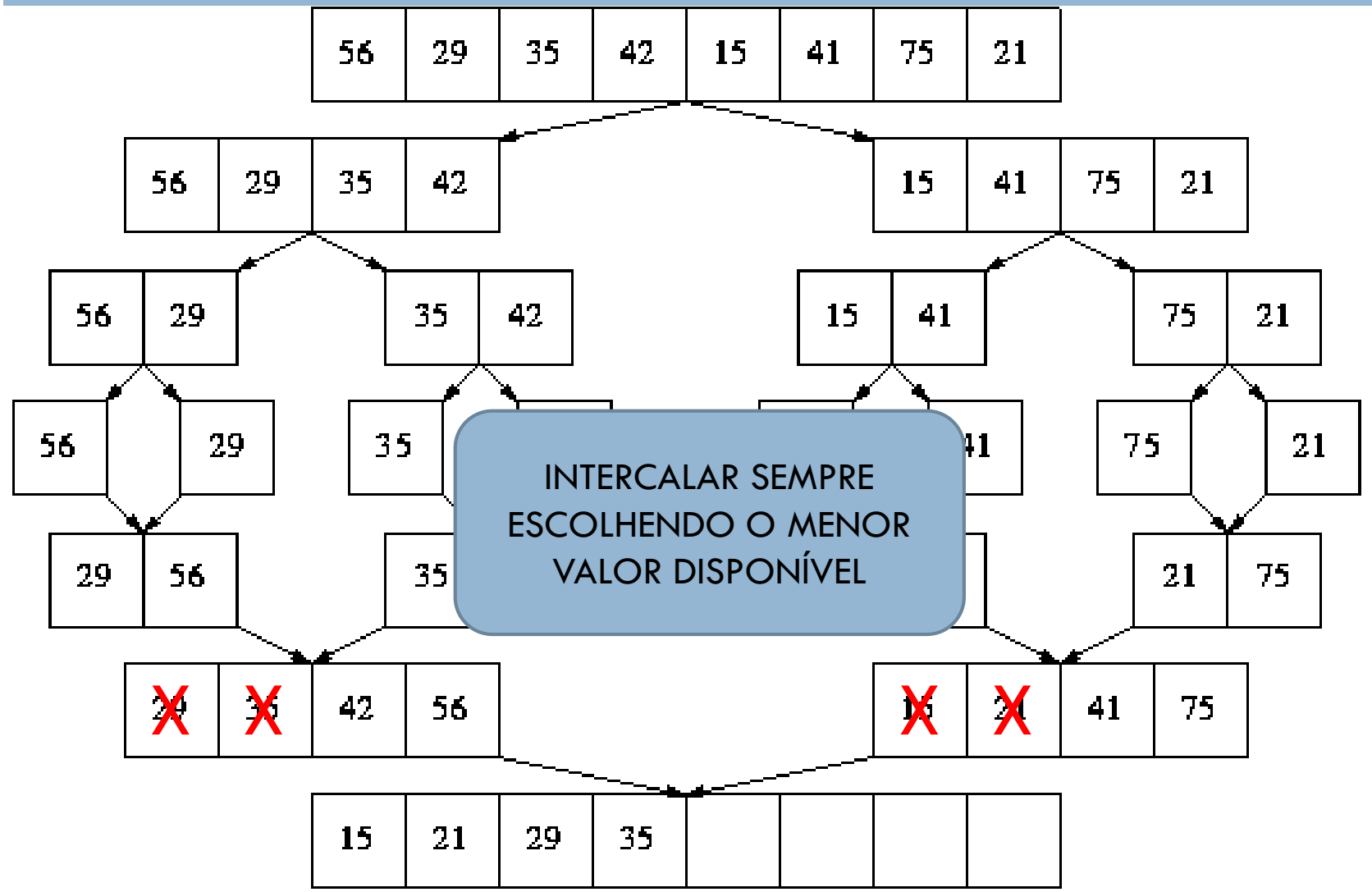
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
75
21
29 56
21 75
29 35 42 56
41 75
15 21
INTERCALAR SEMPRE
ESCOLHENDO O MENOR
VALOR DISPONÍVEL

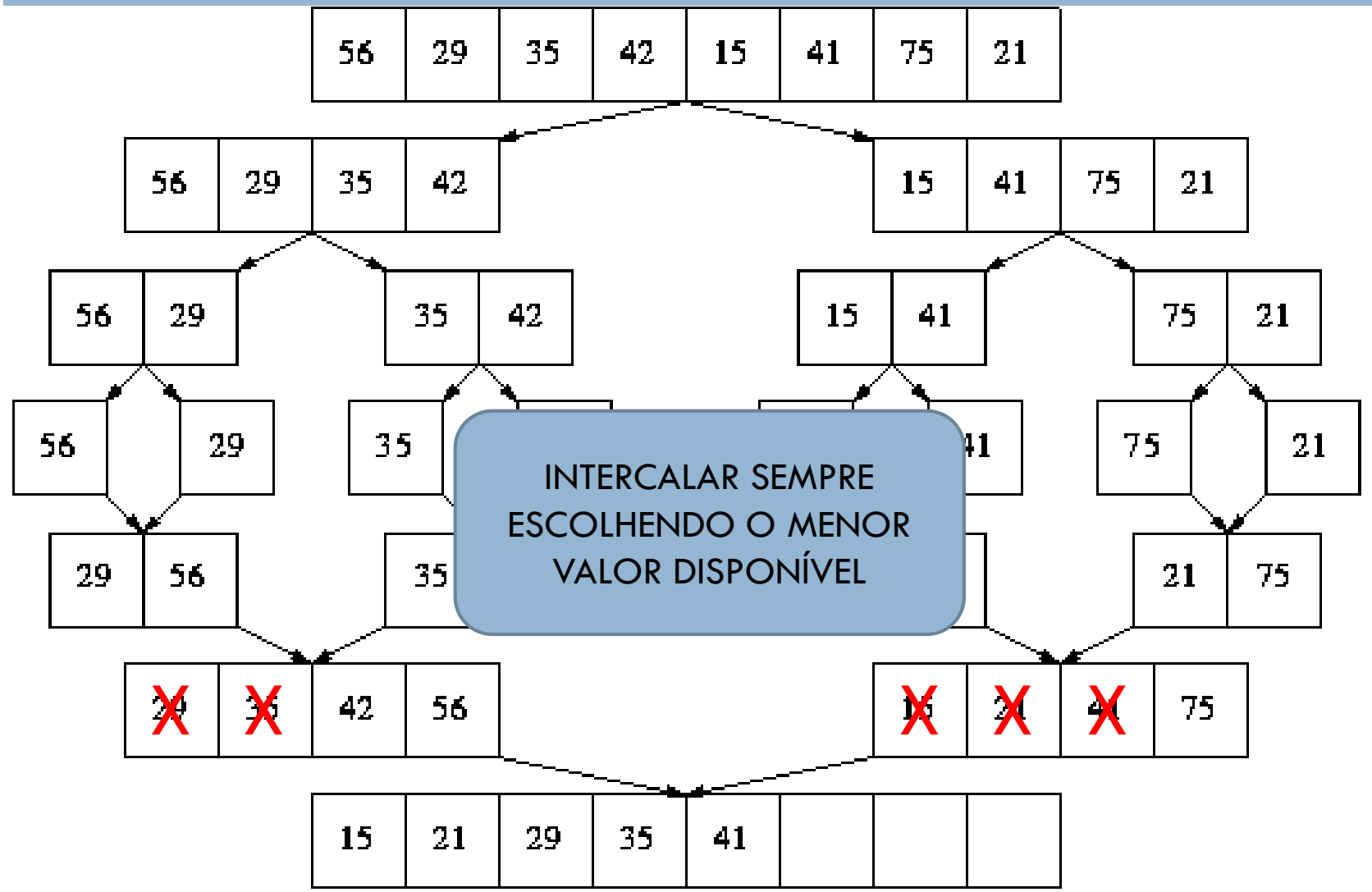
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
75
21
29 56
21 75
INTERCALAR SEMPRE ESCOLHENDO O MENOR VALOR DISPONÍVEL
35 42 56
41 75
15 21 29

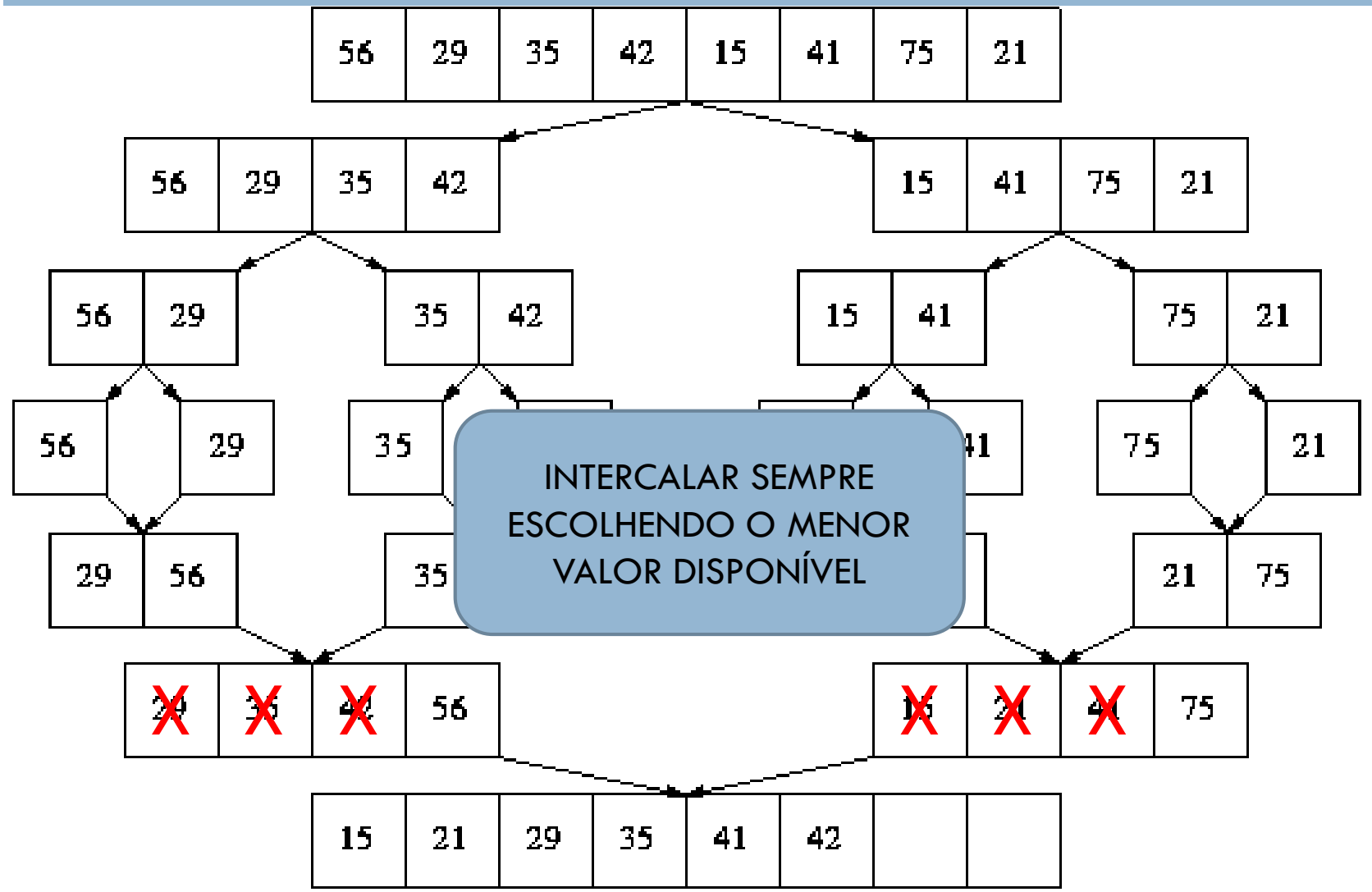
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
75
21
29 56
21 75
42 56
41 75
15 21 29 35
INTERCALAR SEMPRE
ESCOLHENDO O MENOR
VALOR DISPONÍVEL

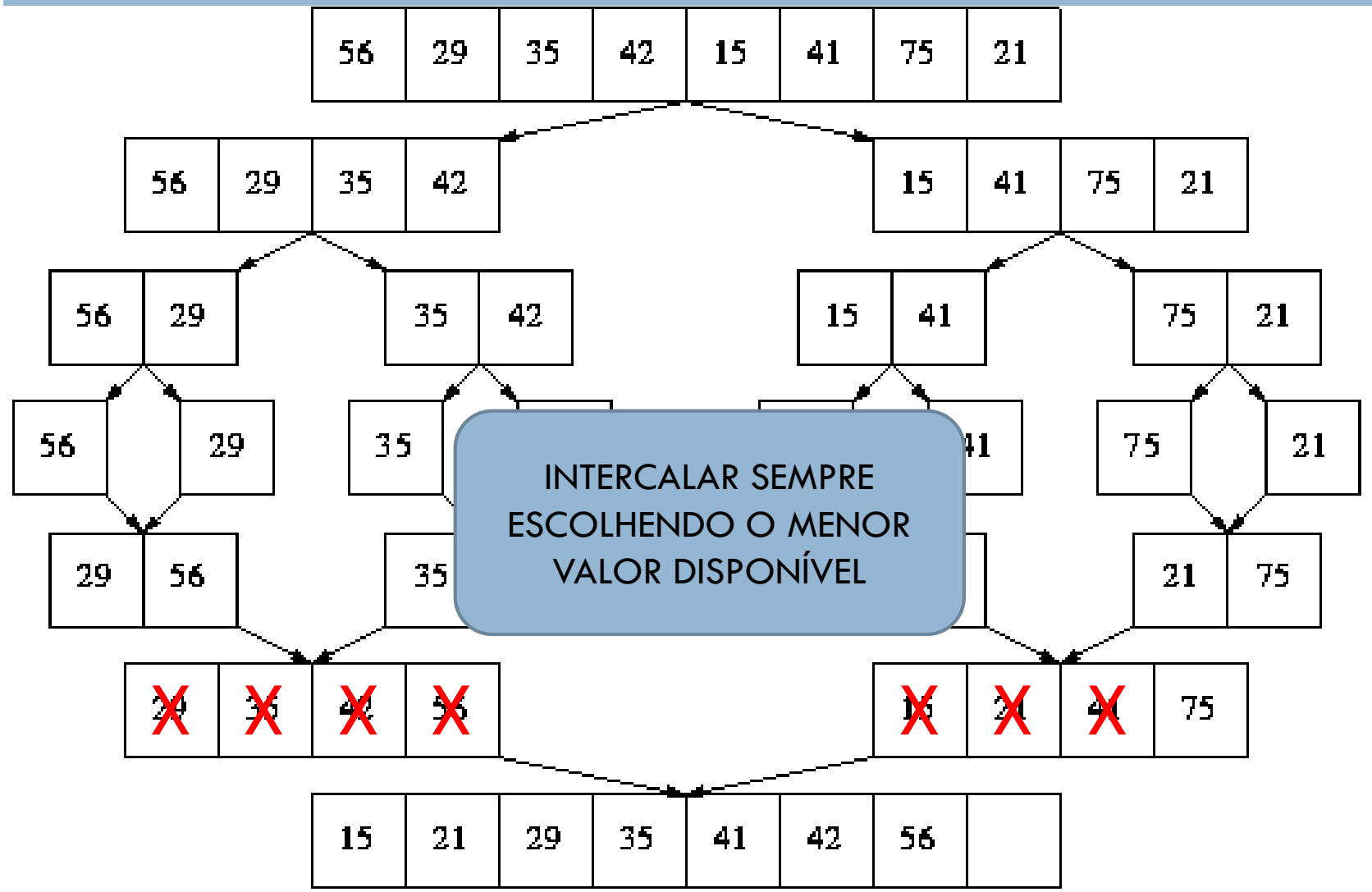
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
75
21
29 56
21 75
42 56
75
15 21 29 35 41
INTERCALAR SEMPRE
ESCOLHENDO O MENOR
VALOR DISPONÍVEL

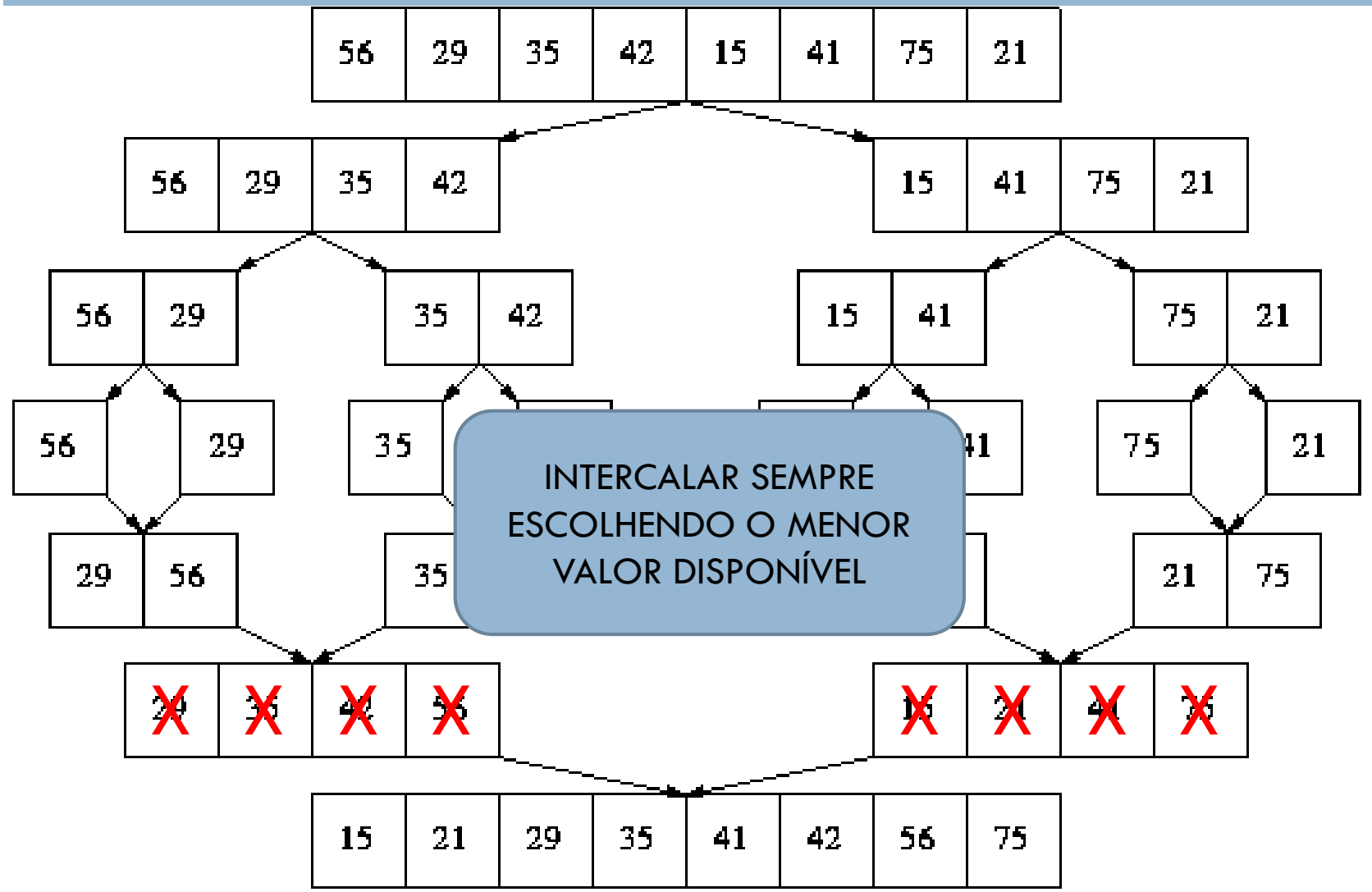
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
75
21
29 56
21 75
15 21 29 35 41 42
INTERCALAR SEMPRE ESCOLHENDO O MENOR VALOR DISPONÍVEL

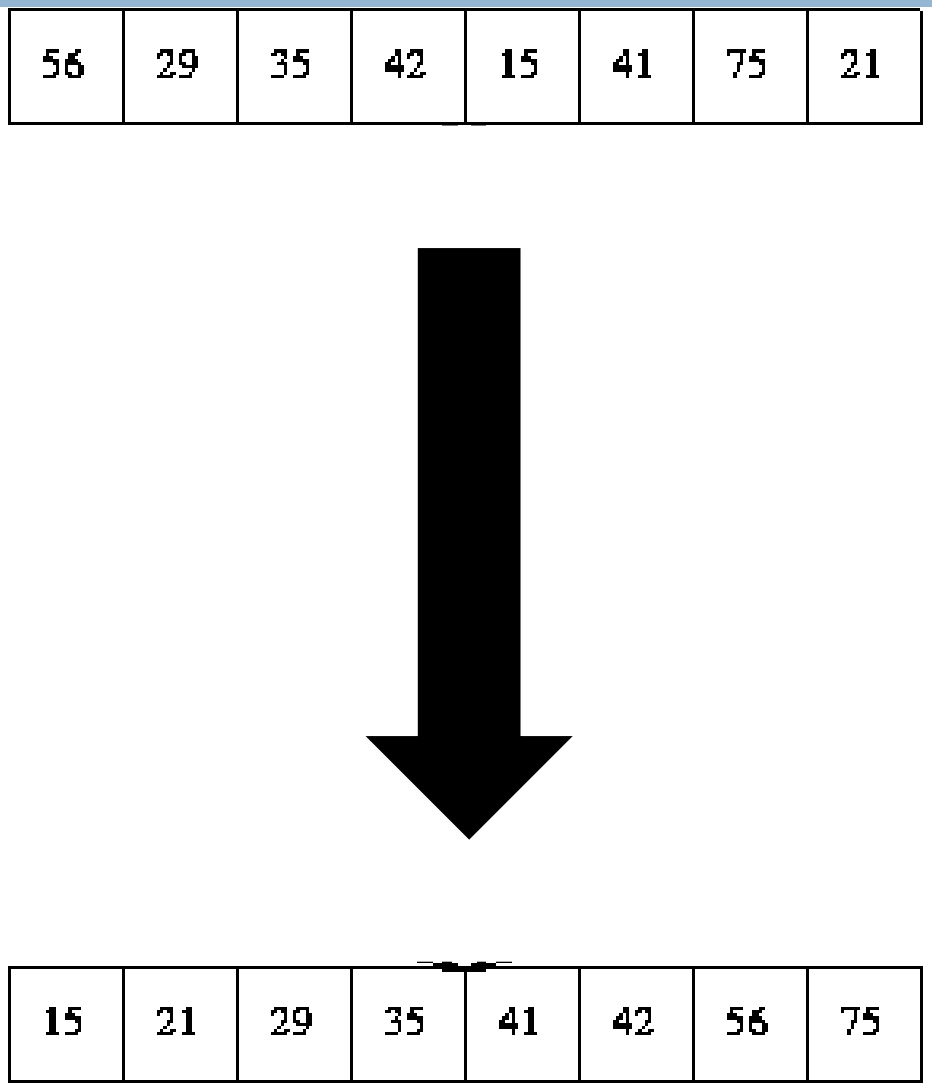
MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
75
21
29 56
21 75
15 21 29 35 41 42 56
INTERCALAR SEMPRE ESCOLHENDO O MENOR VALOR DISPONÍVEL

MergeSort


56 29 35 42 15 41 75 21
56 29 35 42
15 41 75 21
56 29
35 42
15 41
75 21
56
29
75
21
29 56
21 75
15 21 29 35 41 42 56 75
INTERCALAR SEMPRE ESCOLHENDO O MENOR VALOR DISPONÍVEL

MergeSort



MergeSort


DIVIDIR
INTERCALAR
56 29 35 42 15 41 75 21
ordenar
15 21 29 35 41 42 56 75

MergeSort

- Exemplo:

6 5 3 1 8 7 2 4

MergeSort

- Divide o vetor em intervalos, intercala os valores de cada intervalo até obter a ordenação.
- Complexidade: O($nlogn$).
- Algumas implementações consomem muita memória.
- Estável.
- Não é adaptável. Qualquer tipo de entrada irá desencadear o mesmo procedimento.
- Vídeo no Youtube

HeapSort

- Utiliza uma estrutura chamada Heap, que consiste em uma árvore binária.
  - NÃO É UMA ABB(apenas similar na estrutura).
  - A ordenação na árvore depende do tipo de ordenação requisitada (crescente ou descrescente).

HeapSort

- Utiliza uma estrutura chamada Heap, que consiste em uma árvore binária.
  - NÃO É UMA ABB(apenas similar na estrutura).
  - A ordenação na árvore depende do tipo de ordenação requisitada (crescente ou descrescente).
- Utiliza o conceito de árvore em vetor:
  - A raiz é o primeiro elemento (Vetor[0]);
  - Para um nó i:
    - Filho da esquerda está em Vetor[2*i + 1];
    - Filho da direita está em Vetor[2*i + 2].

HeapSort

- O Heap pode ser ordenado como máximo ou mínimo:
  - MaxHeap: raiz é o maior elemento, todos os pais são maiores que os seus filhos.
  - MinHeap: raiz é o menor elemento, todos os pais são menores que os seus filhos.

HeapSort

- Procedimento:
  1. Transforma a estrutura do vetor de entrada em um Heap. Normalmente se utiliza MaxHeap. Inicializa *fim* com o último índice do vetor (tamanho - 1).
  2. Troca o elemento da raiz (vetor[0]) com o elemento da última posição não alterada (vetor[*fim*]).
  3. Subtrai uma posição do *fim*. O elemento já está ordenado em sua posição.
  4. Arranja Heap novamente (para reorganizar o heap).
  5. Repete os passos 2, 3 e 4 até que *fim* alcance o índice 1.

HeapSort

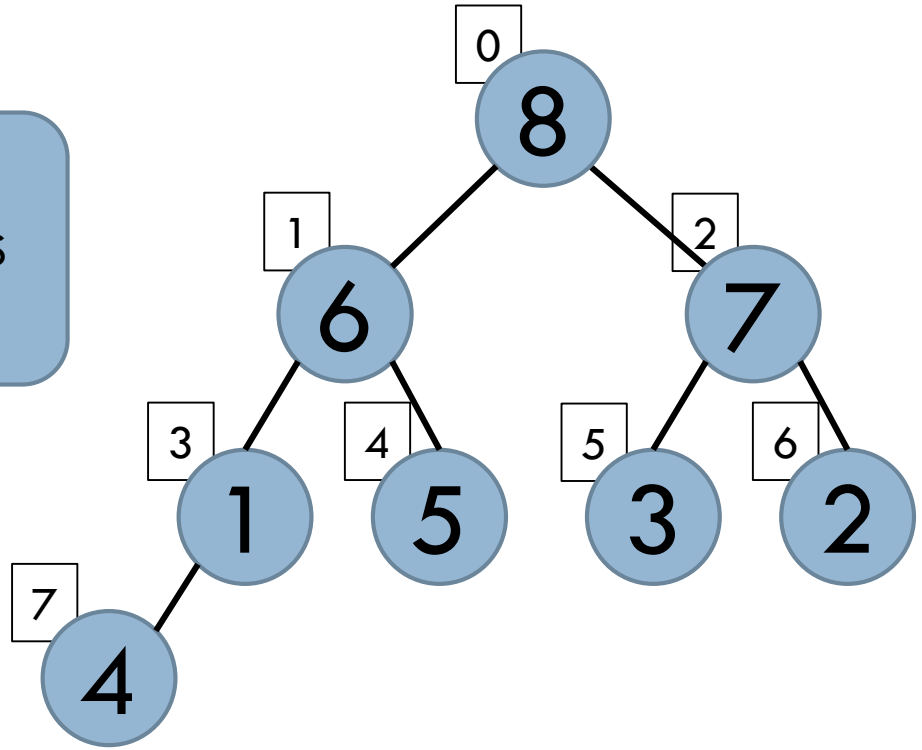
0	1	2	3	4	5	6	7
8	6	7	1	5	3	2	4

USAR REGRAS DE FILHOS DA ESQUERDA E DIREITA EM VETORES PARA MONTAR UMA HEAP.

HeapSort

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

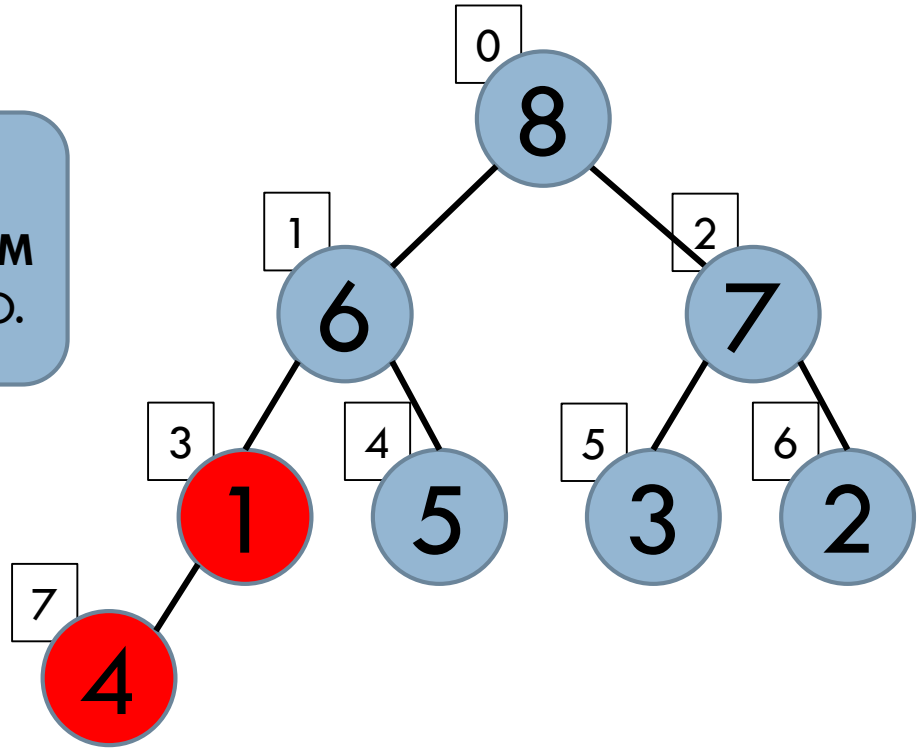
USAR REGRAS DE FILHOS DA ESQUERDA E DIREITA EM VETORES PARA MONTAR UMA HEAP.



HeapSort

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

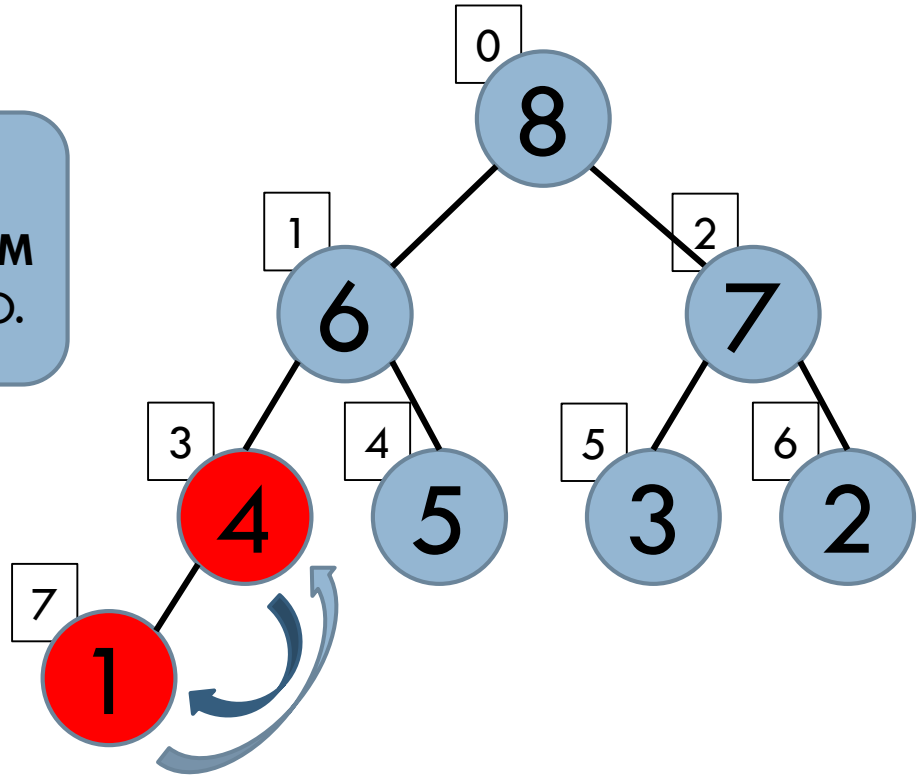
SE DURANTE A CONSTRUÇÃO ALGUM ELEMENTO **SAIR DA ORDEM DA HEAP,** ELE DEVE SER AJUSTADO.



HeapSort

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

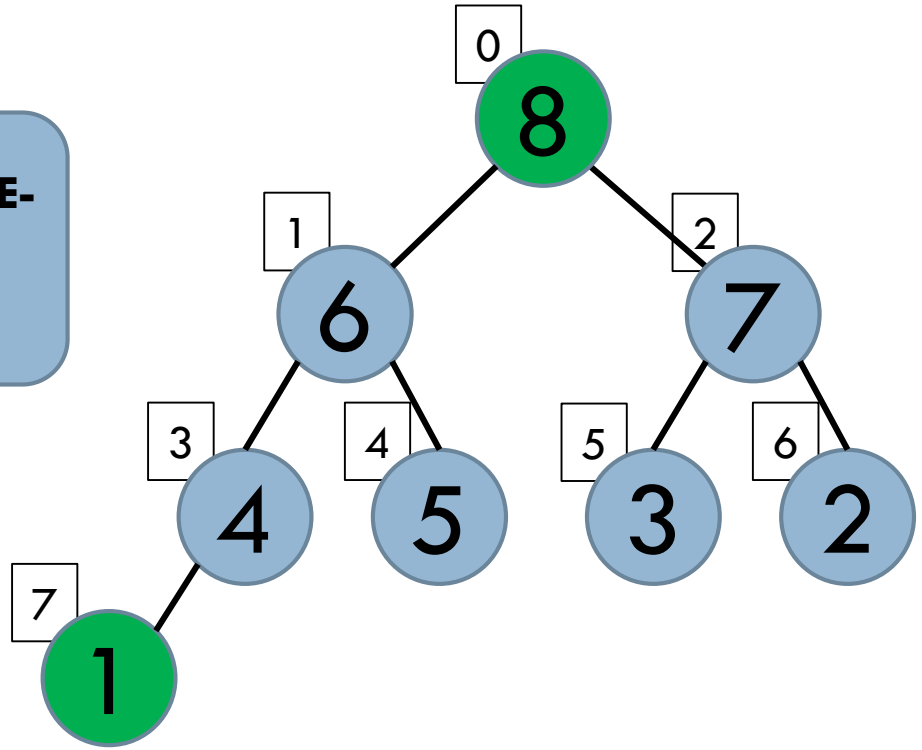
SE DURANTE A CONSTRUÇÃO ALGUM ELEMENTO **SAIR DA ORDEM DA HEAP,** ELE DEVE SER AJUSTADO.



HeapSort

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

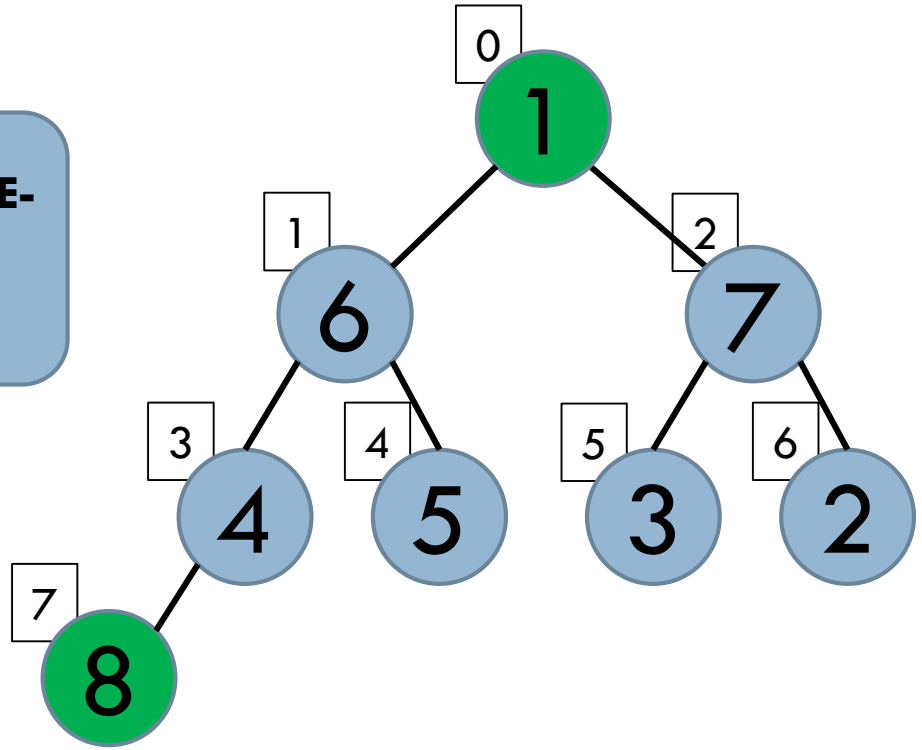
APÓS CONSTRUIR A HEAP, **INVERTE-SE** A RAIZ COM O ÚLTIMO ELEMENTO DO VETOR.



HeapSort

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

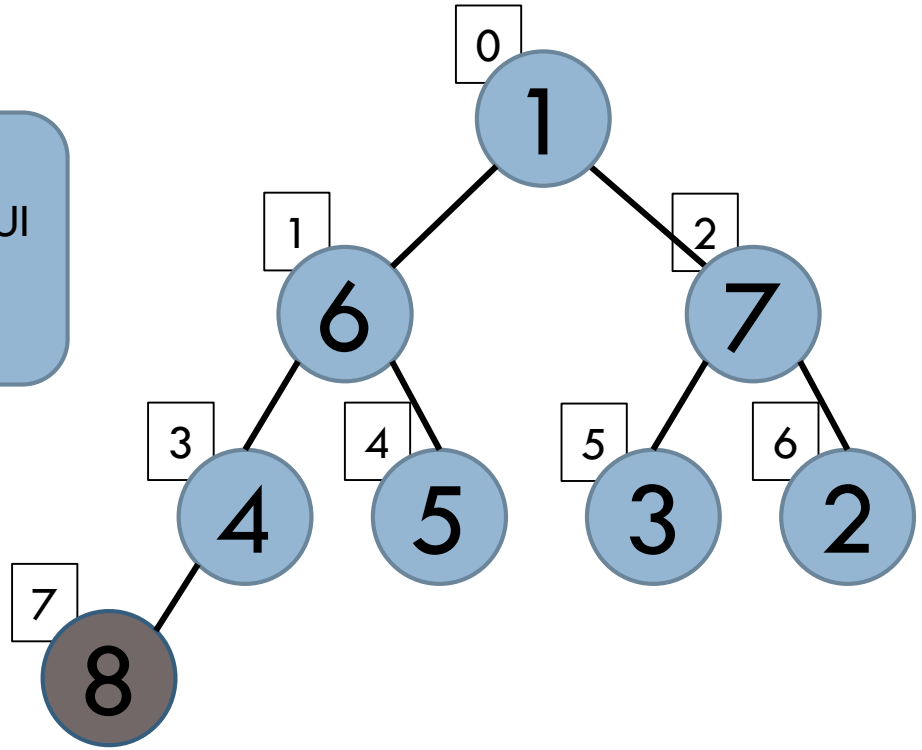
APÓS CONSTRUIR A HEAP, **INVERTE-SE** A RAIZ COM O ÚLTIMO ELEMENTO DO VETOR.



HeapSort

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

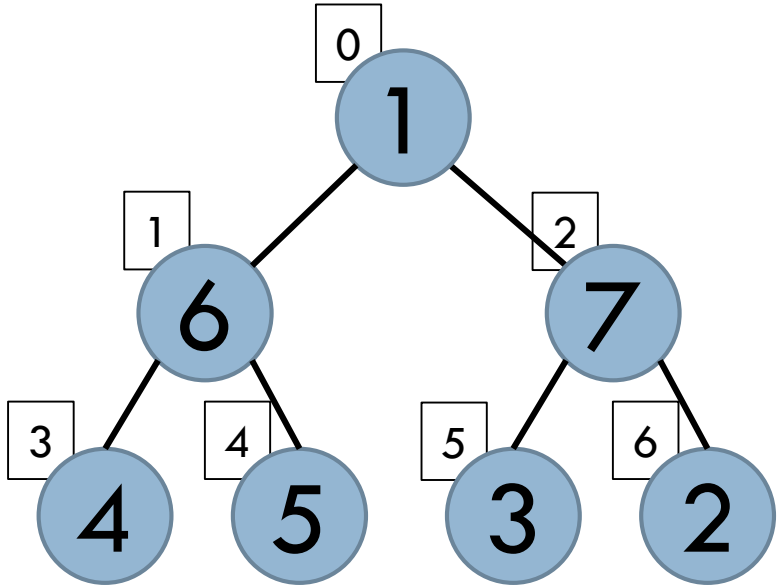
GARANTE-SE QUE O ÍNDICE POSSUI O MAIOR VALOR DA ÁRVORE.



HeapSort

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

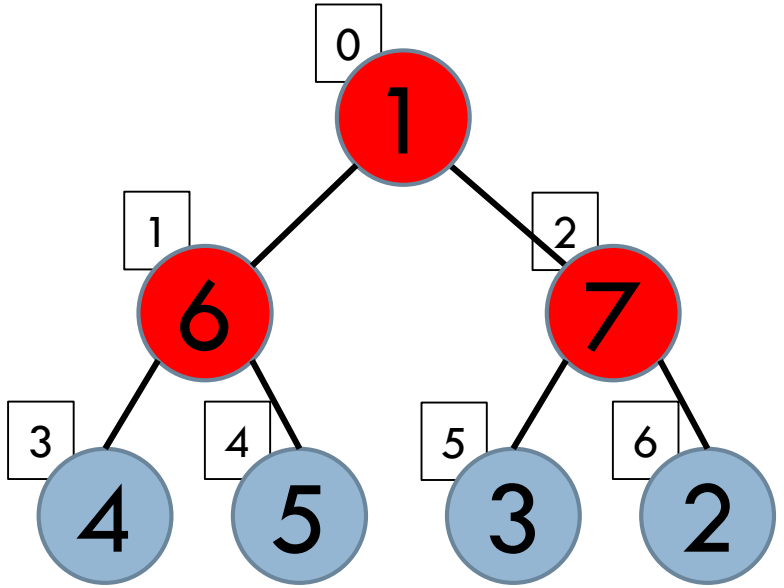
A HEAP É RECONSTRUIDA PARA A PRÓXIMA ITERAÇÃO, DESCARTANDO O ÚLTIMO ÍNDICE.



HeapSort

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

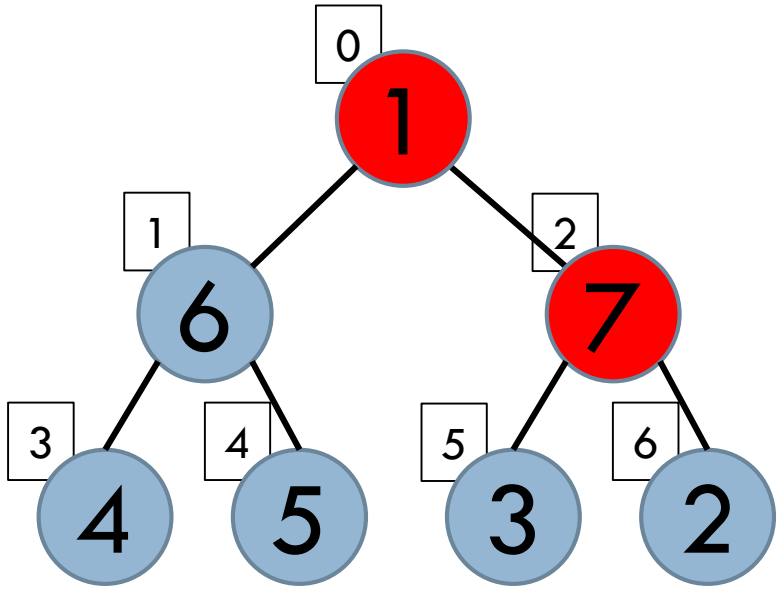
SE DURANTE A CONSTRUÇÃO ALGUM ELEMENTO **SAIR DA ORDEM DA HEAP,** ELE DEVE SER AJUSTADO.



HeapSort

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

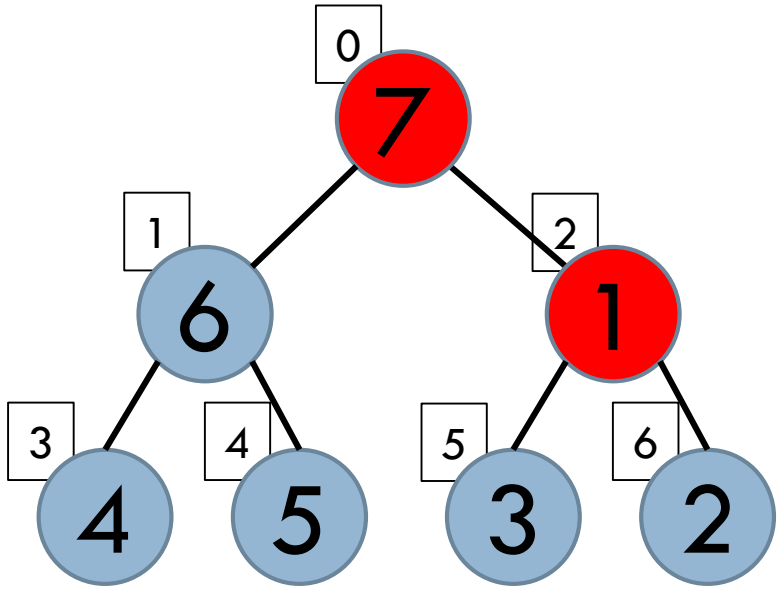
REALIZAR TROCA SELECIONANDO O MAIOR FILHO.



HeapSort

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

REALIZAR TROCA SELECIONANDO O MAIOR FILHO.

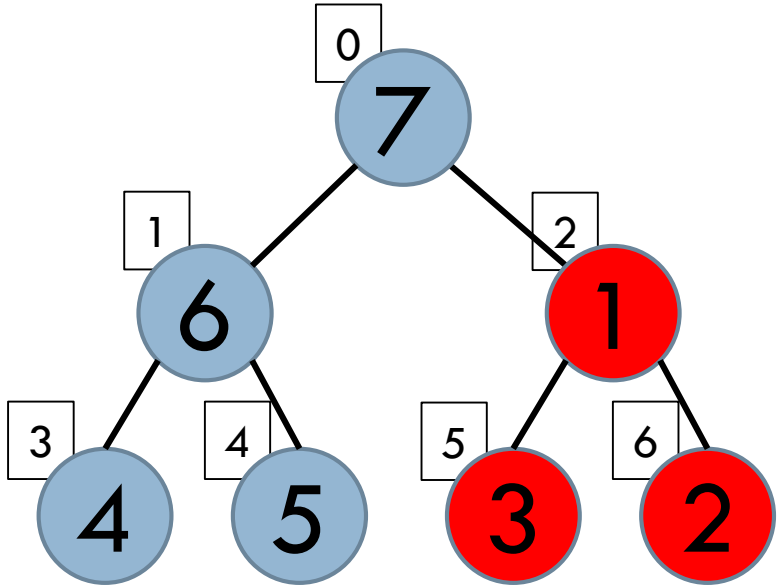


HeapSort

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

SE DURANTE A CONSTRUÇÃO ALGUM ELEMENTO **SAIR DA ORDEM DA HEAP,** ELE DEVE SER AJUSTADO.

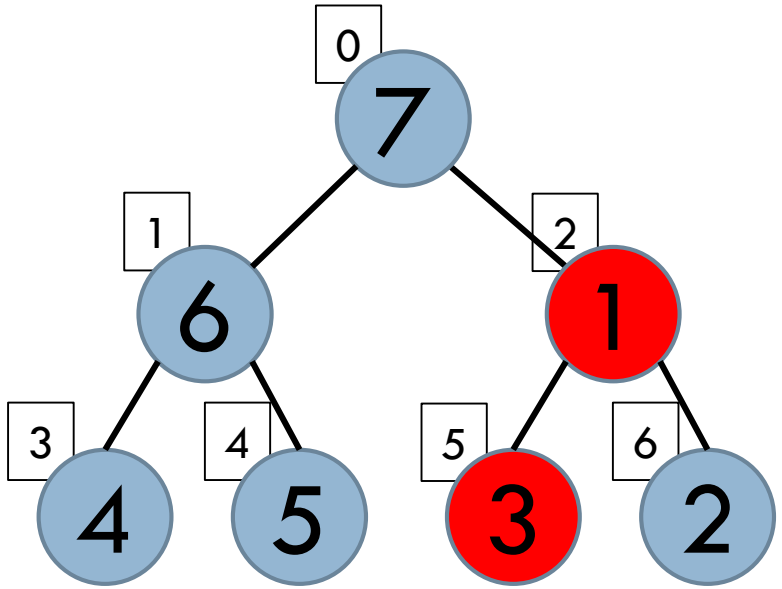
A PROPAGAÇÃO DEVE SER ANALISADA PARA TODOS OS FILHOS.



HeapSort

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

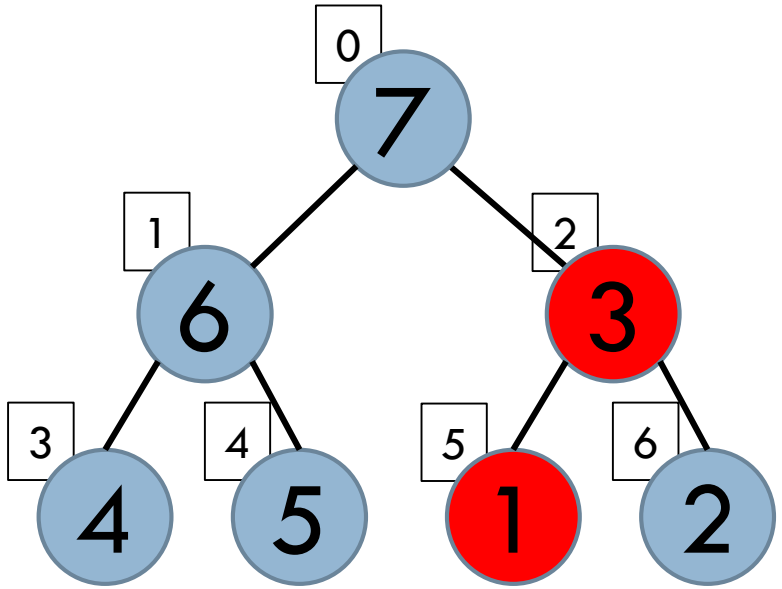
REALIZAR TROCA SELECIONANDO O MAIOR FILHO.



HeapSort

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

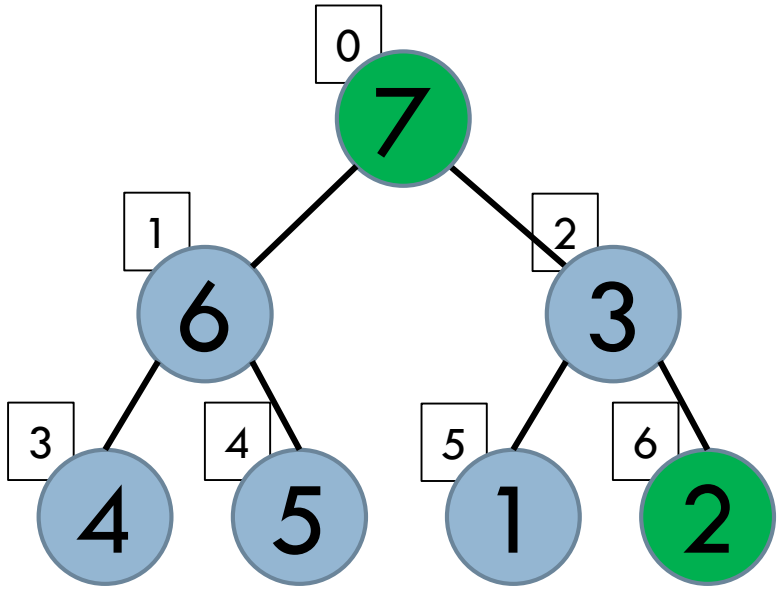
REALIZAR TROCA SELECIONANDO O MAIOR FILHO.



HeapSort

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

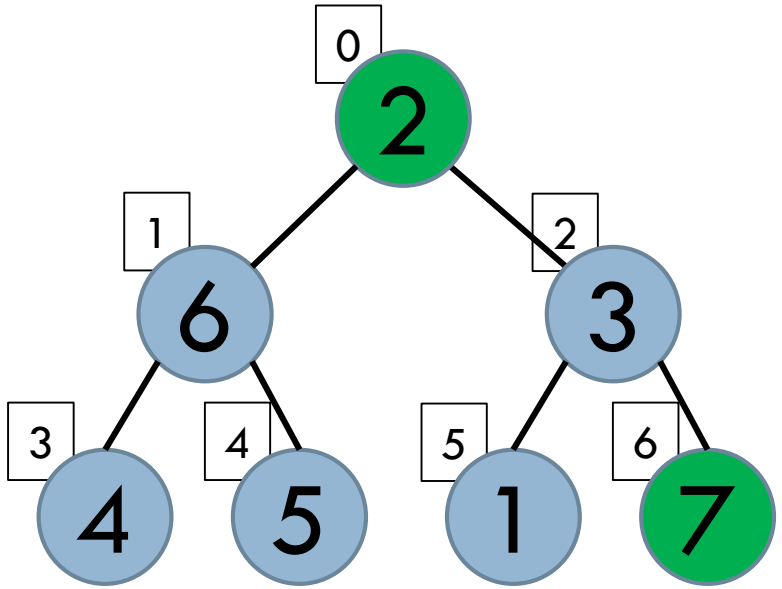
APÓS CONSTRUIR A HEAP, **INVERTE-SE** A RAIZ COM O ÚLTIMO ELEMENTO DO VETOR.



HeapSort

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

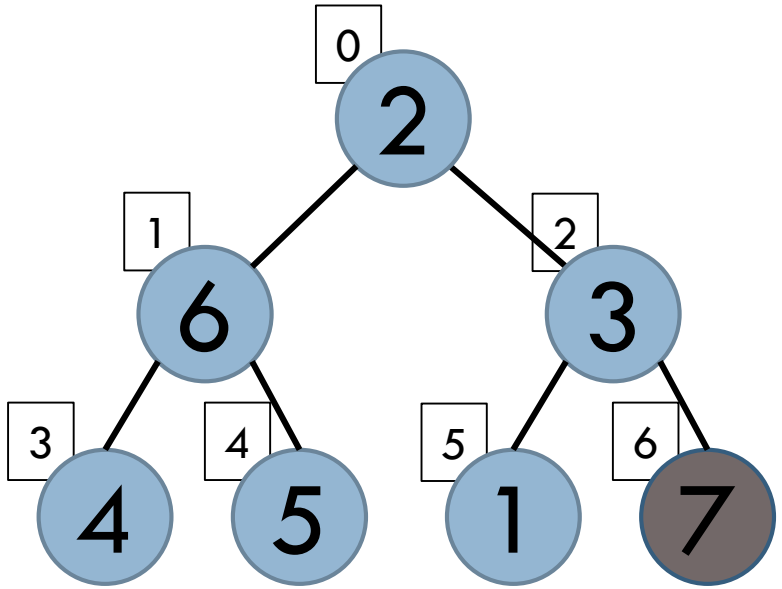
APÓS CONSTRUIR A HEAP, **INVERTE-SE** A RAIZ COM O ÚLTIMO ELEMENTO DO VETOR.



HeapSort

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

GARANTE-SE QUE O ÍNDICE POSSUI O MAIOR VALOR DA ÁRVORE.

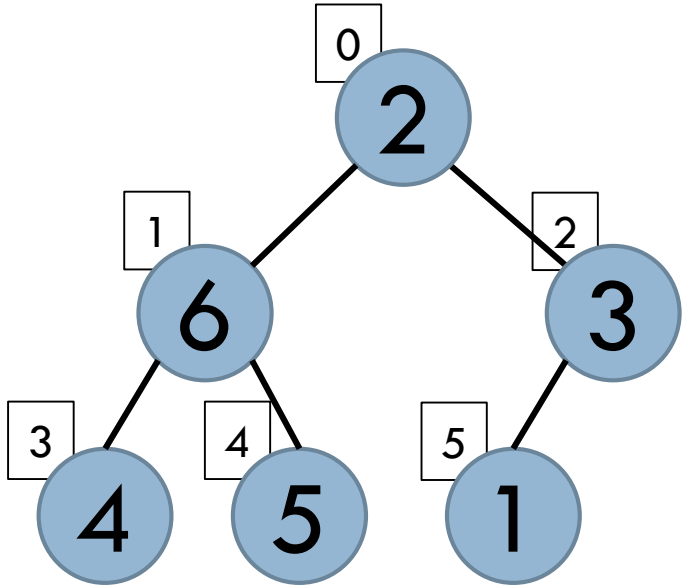


HeapSort

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

A HEAP É RECONSTRUIDA PARA A PRÓXIMA ITERAÇÃO, DESCARTANDO O ÚLTIMO ÍNDICE.

O PROCESSO DE RECONSTRUÇÃO E REMOÇÃO DA RAIZ OCORRE ATÉ O ORDENAR TODOS OS ELEMENTOS DO VETOR.



HeapSort

- Exemplo:

6 5 3 1 8 7 2 4

HeapSort

- Utiliza uma estrutura de árvore para organizar os dados. Preenche o vetor original com as raízes do Heap em cada iteração.
- Complexidade: O($nlogn$) em todos os casos.
- Melhor forma de resolução, de acordo com a complexidade.
- É instável.
- Não é adaptável.
- Vídeo no Youtube