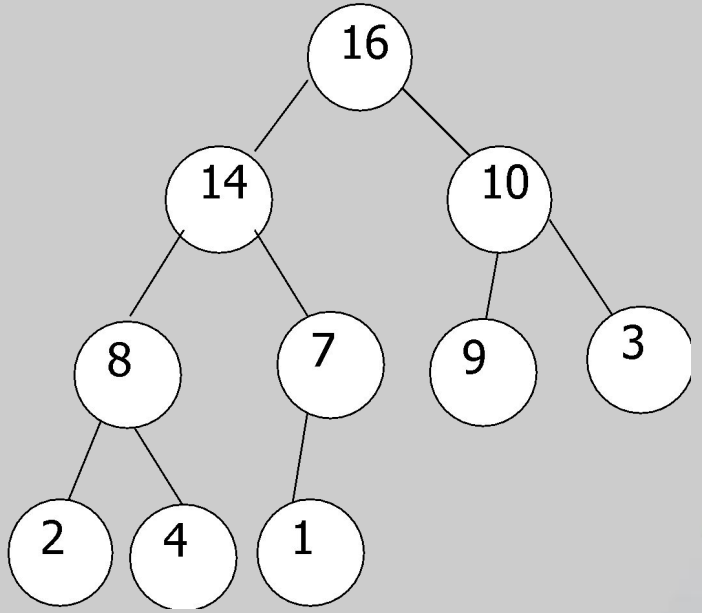
ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES II

Algoritmos clássicos de ordenação III

Heap sort

Utiliza uma estrutura heap para ordenar os elementos.

Um heap é uma estrutura de dados em que há uma ordenação dos elementos: representação via árvore binária.



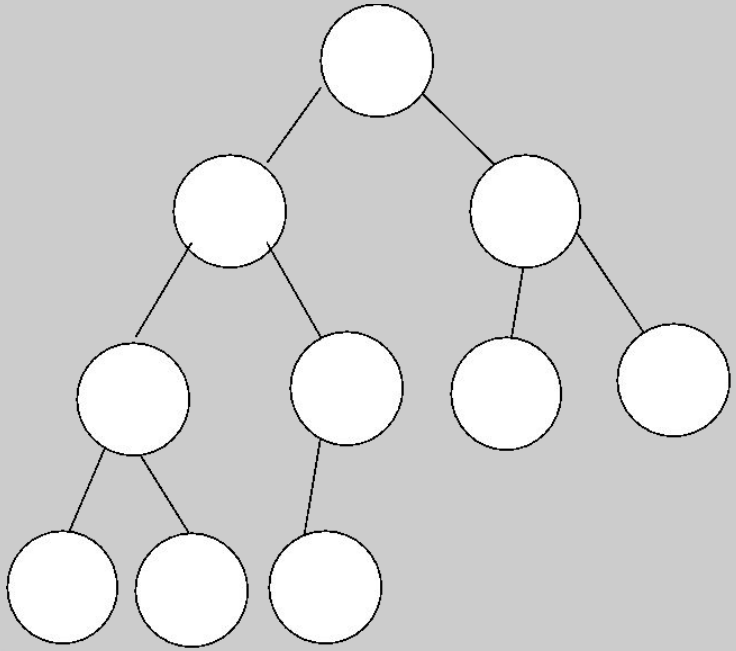
Heap sort

Um heap observa conceitos de ordem e de forma.

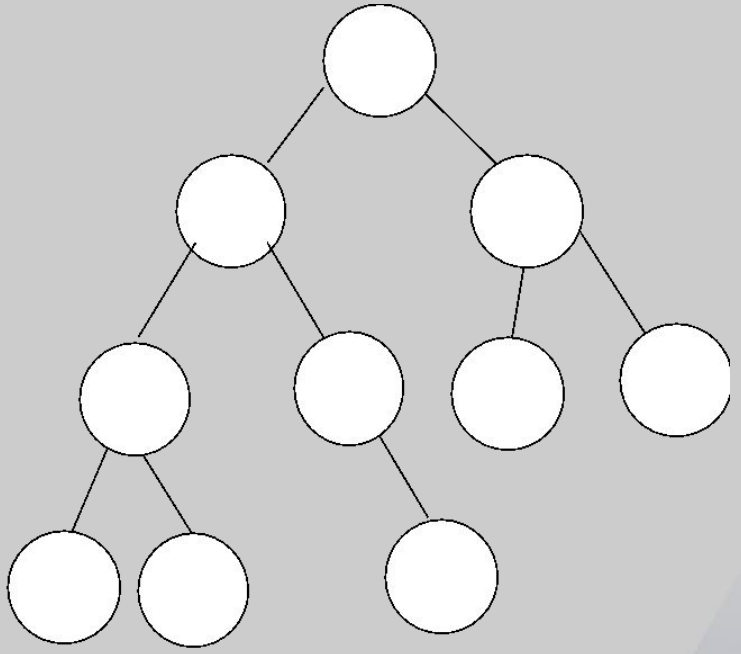
- Ordem: o item de qualquer nó deve satisfazer uma relação de ordem com os itens dos nós filhos.
  - Heap máximo: pai >= filhos
  - Heap mínimo: pai <= filhos
- Forma: árvore binária tem que ser completa até o penúltimo nível, sendo que no último nível os nós têm que estar agrupados à esquerda.

Heap sort

Exemplos



OK

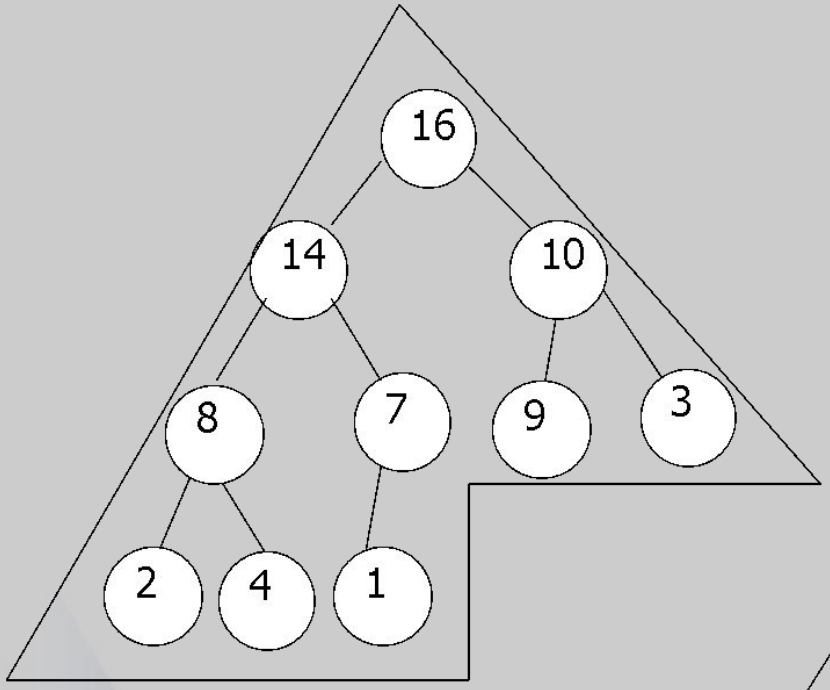


Não!

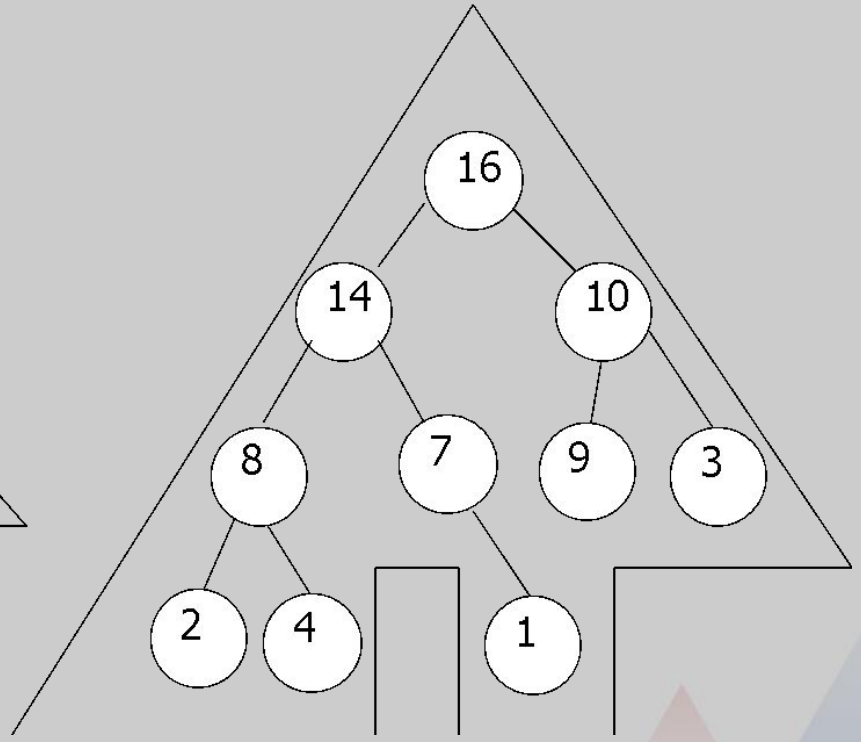
Heap sort

Exemplos

É um heap máximo



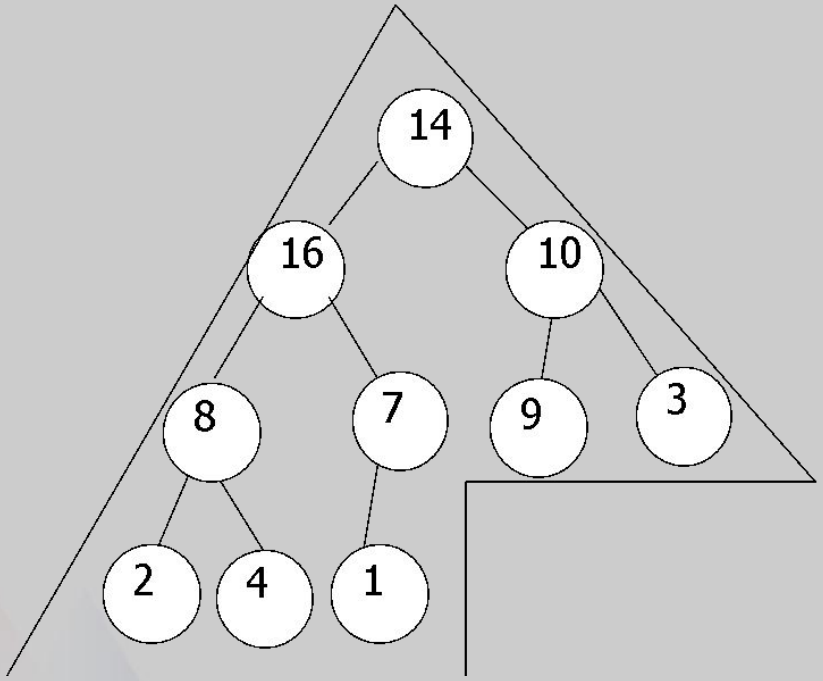
Não é um heap máximo



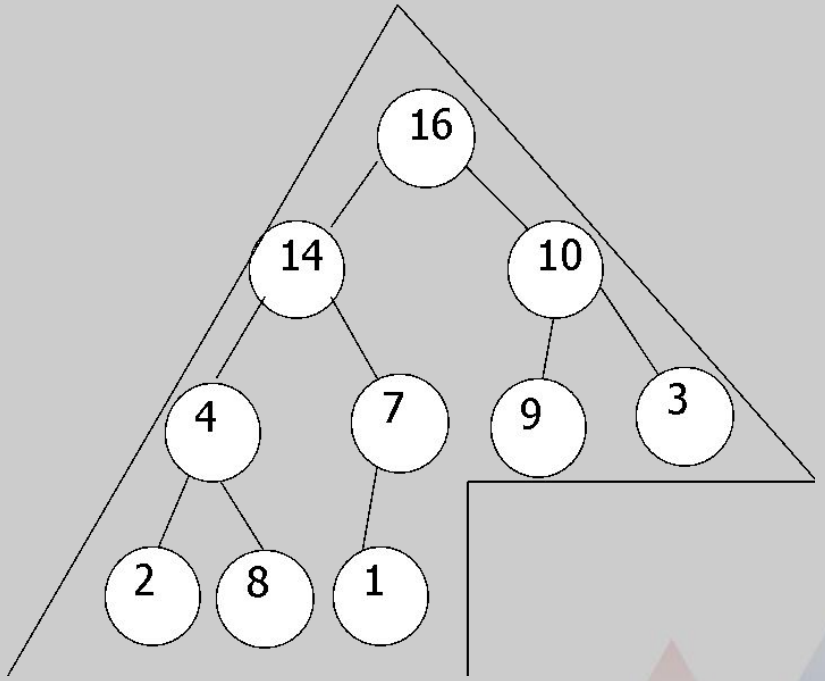
Heap sort

Exemplos

Não é um heap máximo

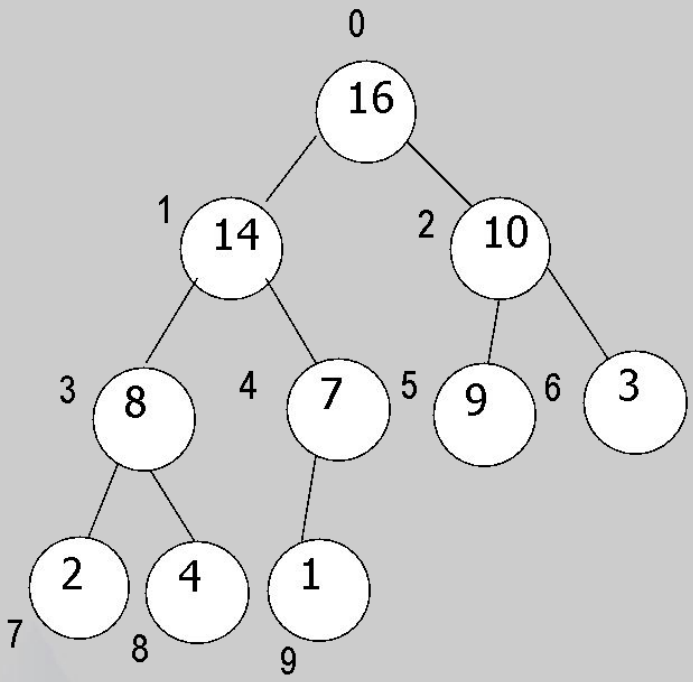


Não é um heap máximo



Heap sort

Um heap pode ser representado por um vetor.



0	1	2	3	4	5	6	7	8	9
16	14	10	8	7	9	3	2	4	1

Filhos do nó k:
- filho esquerdo = 2k + 1
- filho direito = 2k + 2

Pai do nó k: (k-1)/2

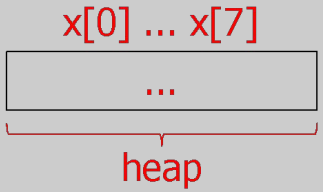
Folhas de n/2 em diante

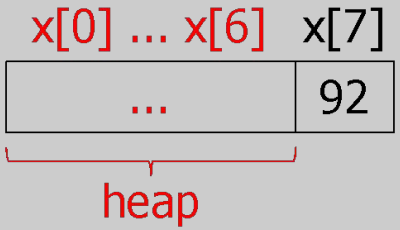
Heap sort

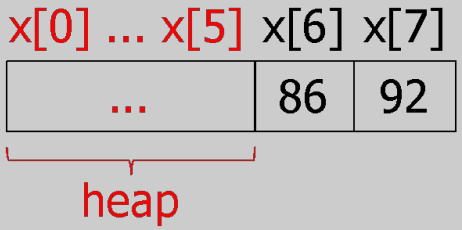
A estrutura heap pode ser usada para ordenar um vetor.

- Construir um heap máximo
- Trocar a raíz (maior elemento) com o elemento da última posição do vetor
- Diminuir o tamanho do heap em 1
- Rearranjar o heap máximo (agora com n-1 elementos), se necessário
- Repetir o processo n-1 vezes.

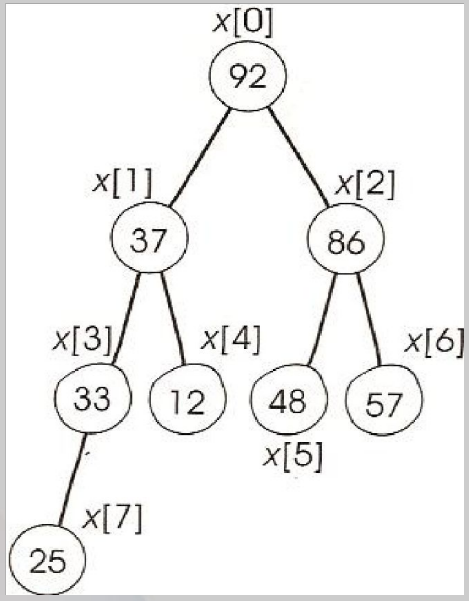
Heap sort



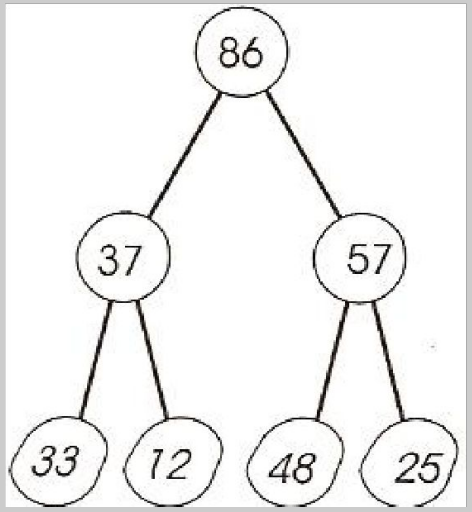




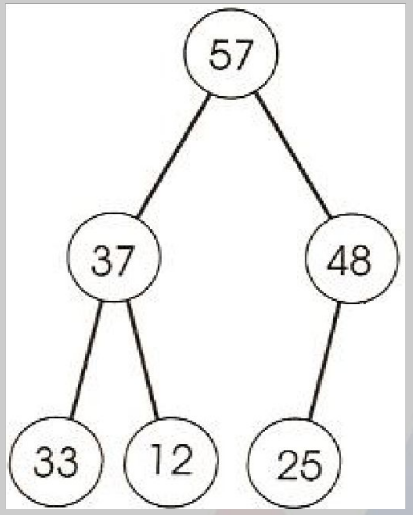
1) Monta-se o heap com base no vetor desordenado



2) Troca-se a raiz (maior elemento) com o último elemento (x[7]) e rearranja-se o heap



3) Troca-se a raiz com o último elemento (x[6]) e rearranja-se o heap



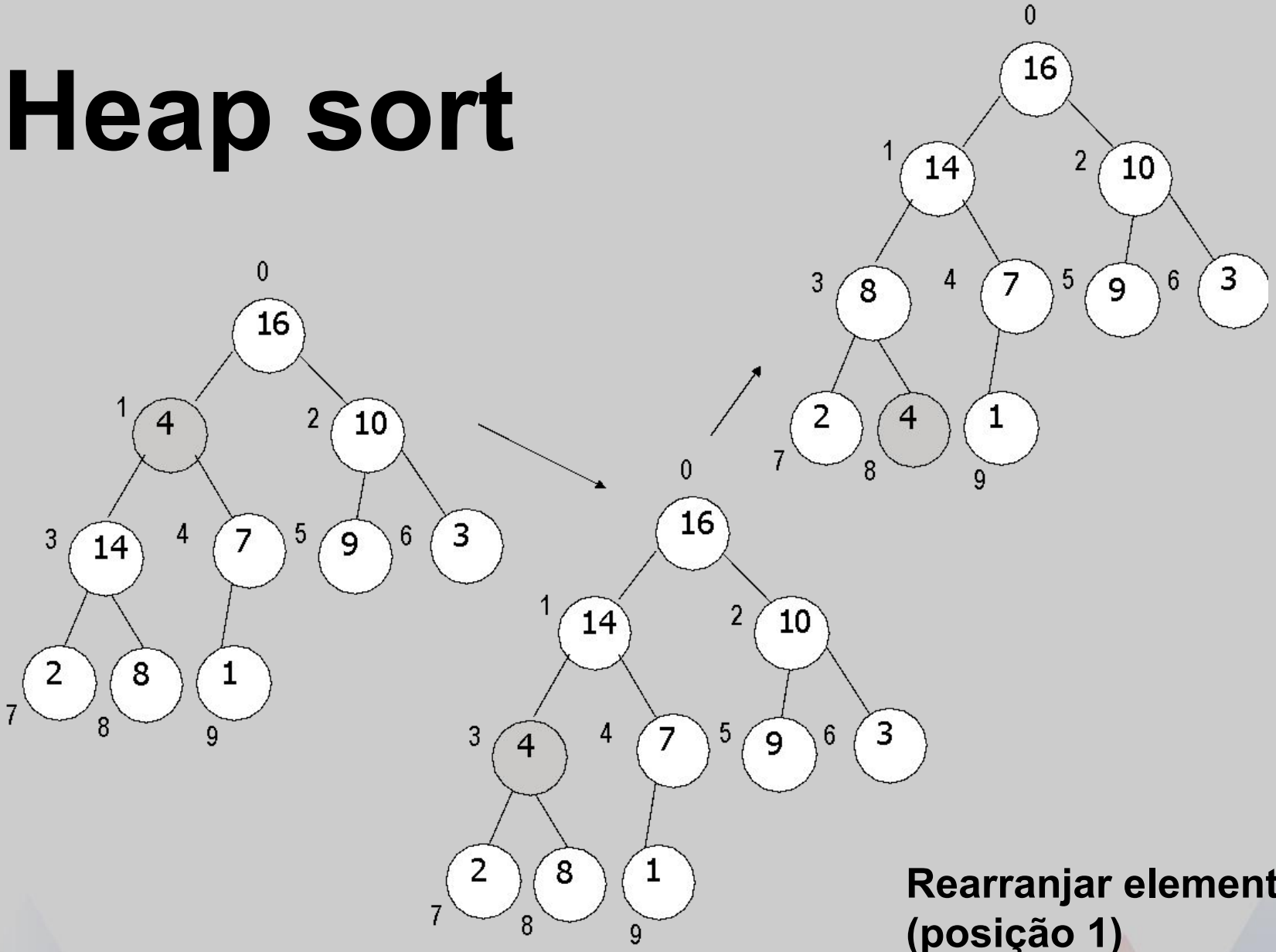
Heap sort

O processo termina até todos os elementos terem sido incluídos no vetor de forma ordenada.

É necessário:

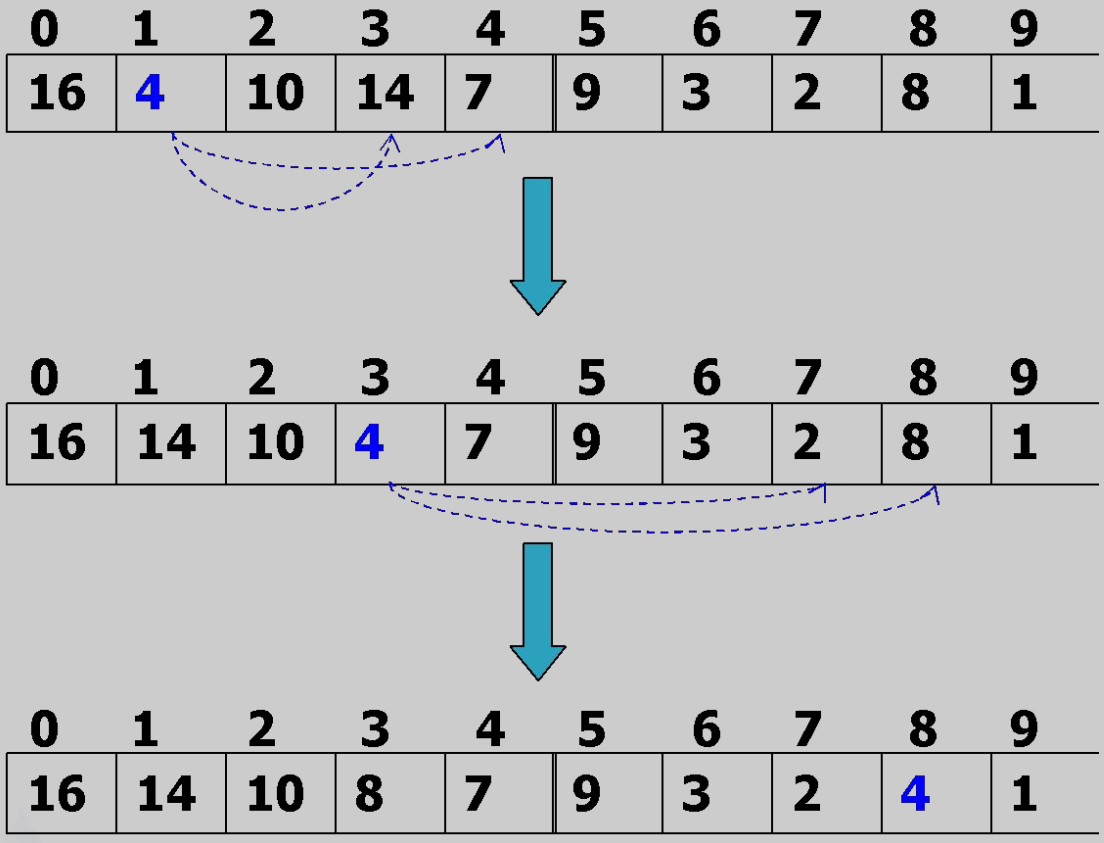
- Saber construir um heap a partir de um vetor qualquer.
- Saber como rearranjar o heap, i.e., manter a propriedade do heap máximo.

Heap sort



Rearranjar elemento 4 (posição 1)

Heap sort



Filhos do nó k:
Filho esquerdo: 2k+1
Filho direito: 2k+2

Heap sort

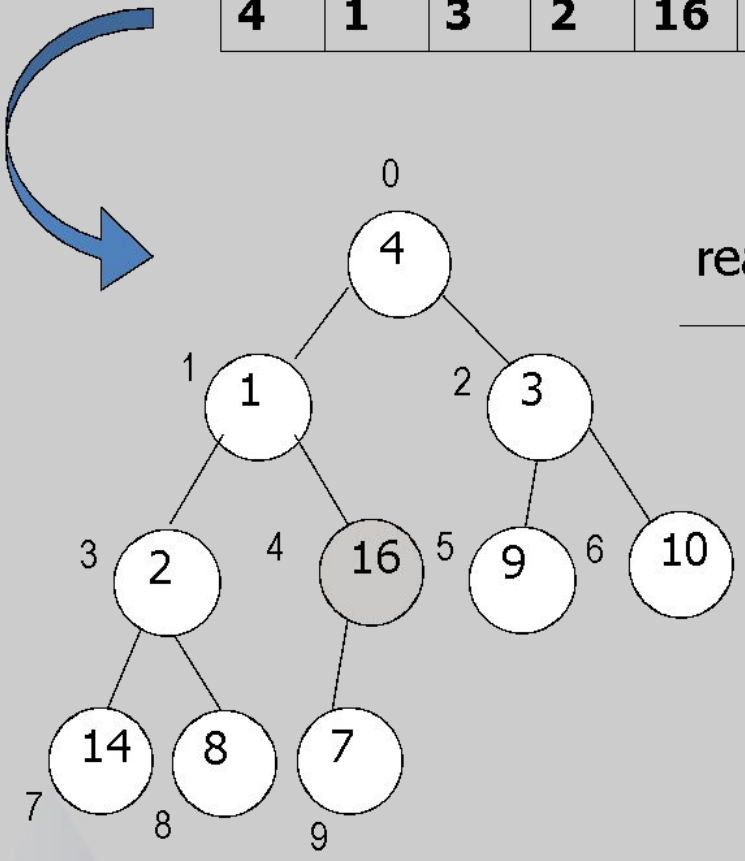
A construção do heap envolve chamar o procedimento rearranjar heap de forma ascendente para os n/2 – 1 nós da árvore (nós não folha).

Com isso, ao chamar pela última vez o procedimento para a raíz, teremos o heap máximo construído.

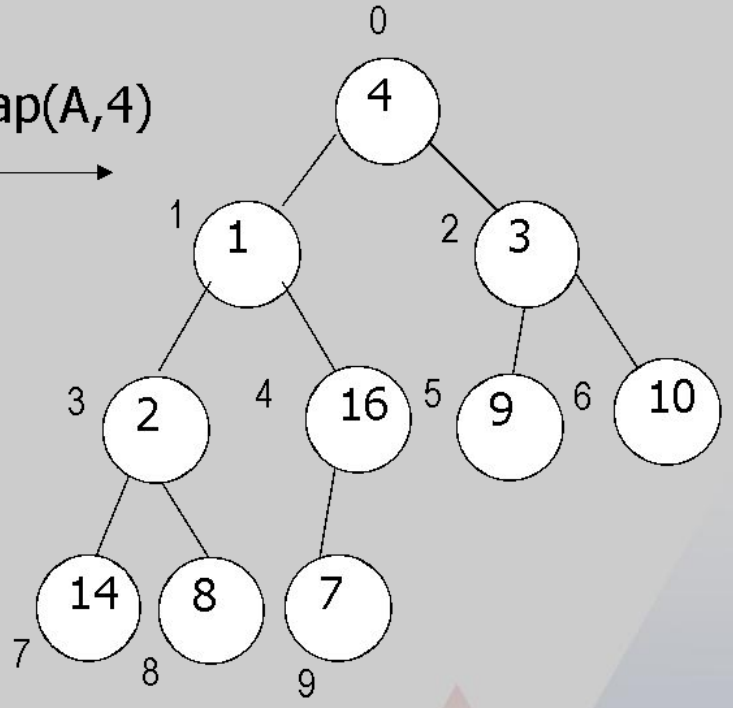
Heap sort

0	1	2	3	4	5	6	7	8	9
4	1	3	2	16	9	10	14	8	7

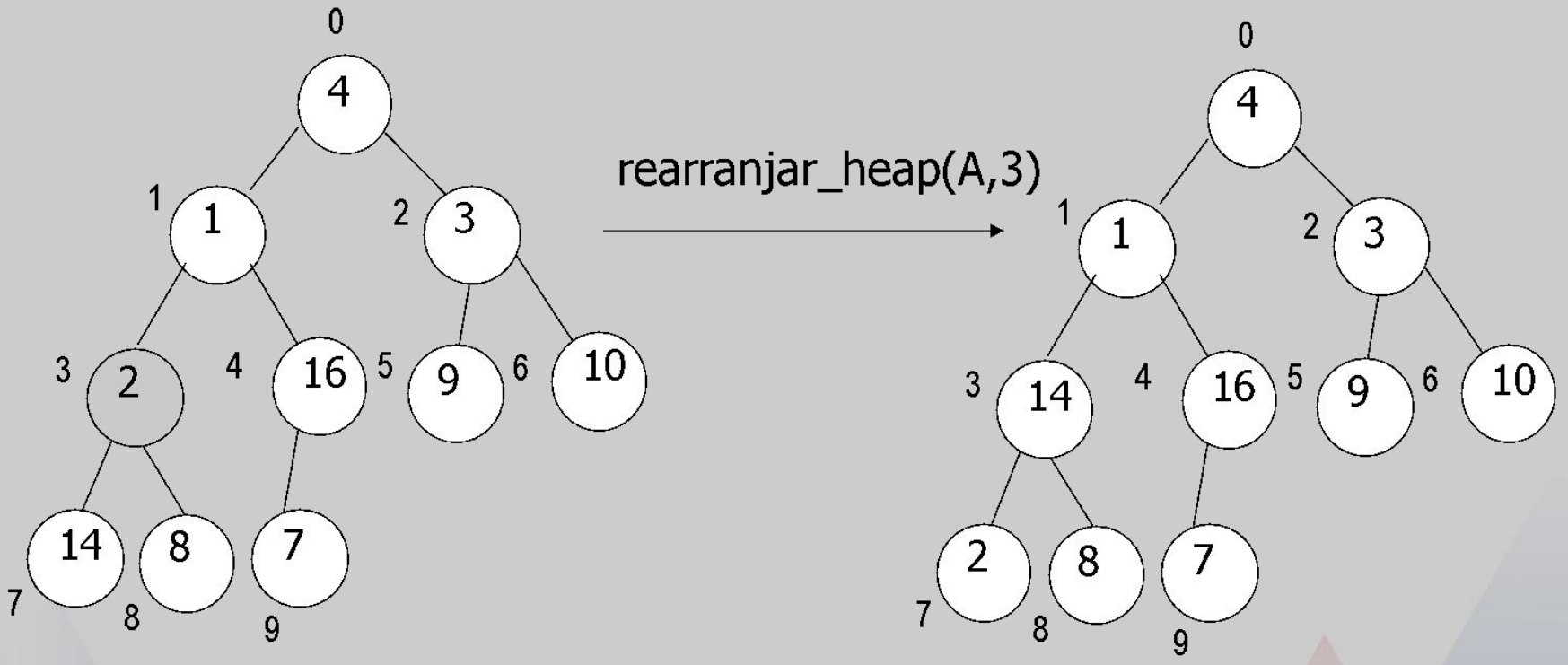
n/2 - 1=4



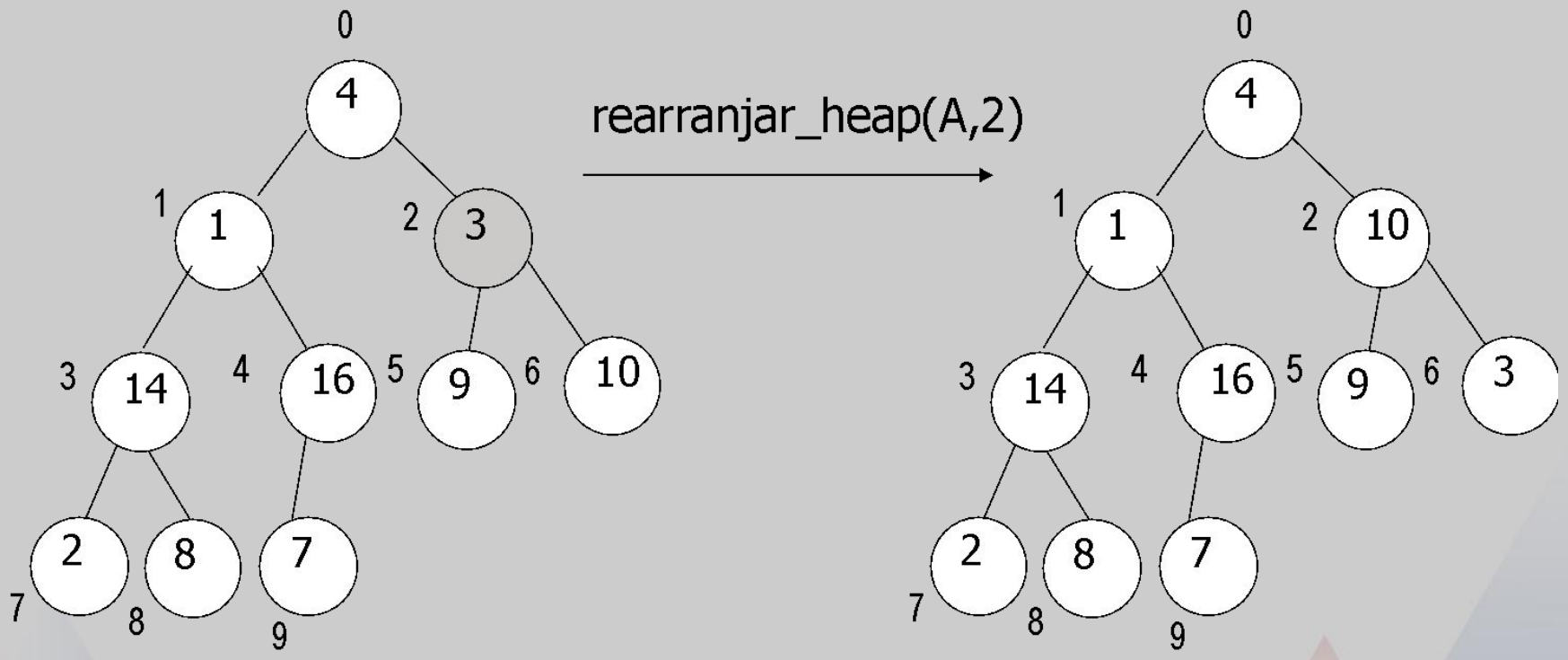
rearranjar_heap(A,4)



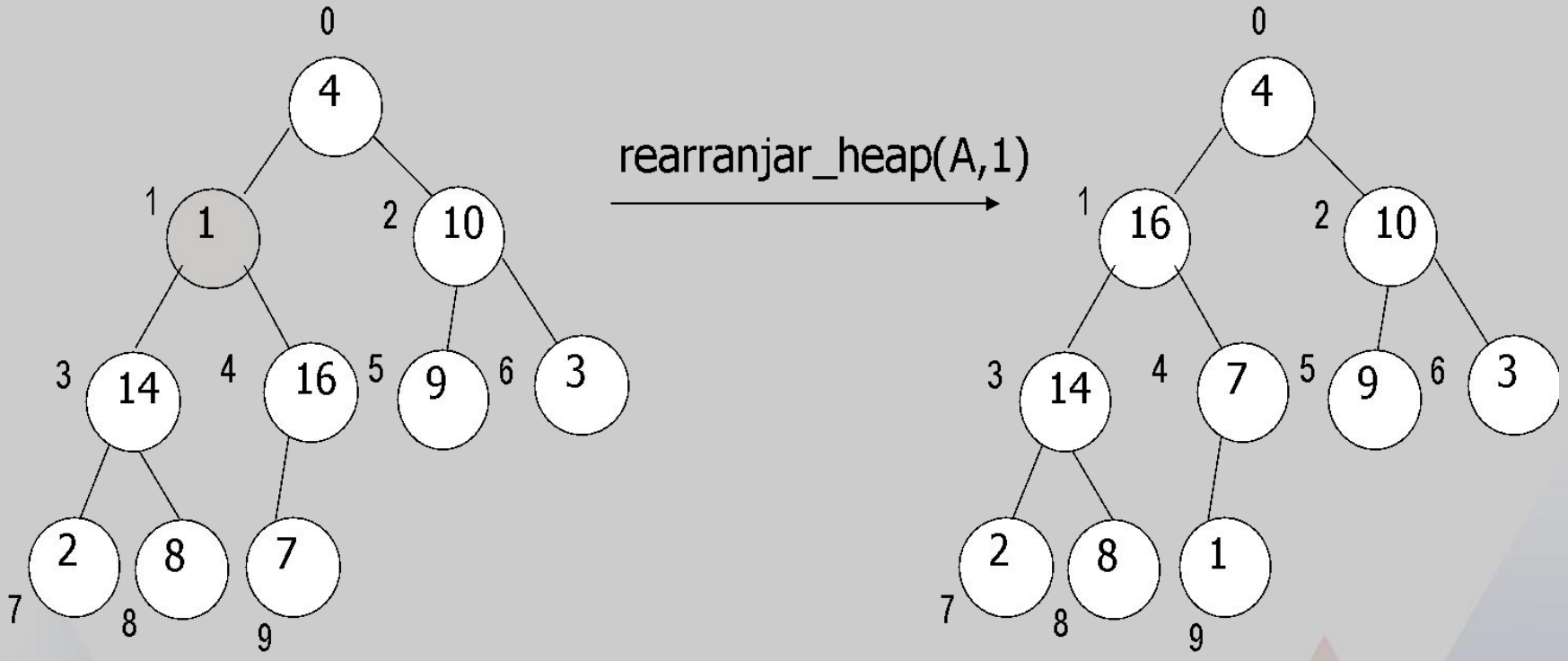
Heap sort


0
4
1
1
2
3
3
2
4
16
5
9
6
10
14
8
7
7
8
9
rearranjar_heap(A,3)
0
4
1
1
2
3
3
14
4
16
5
9
6
10
2
8
7
7
8
9

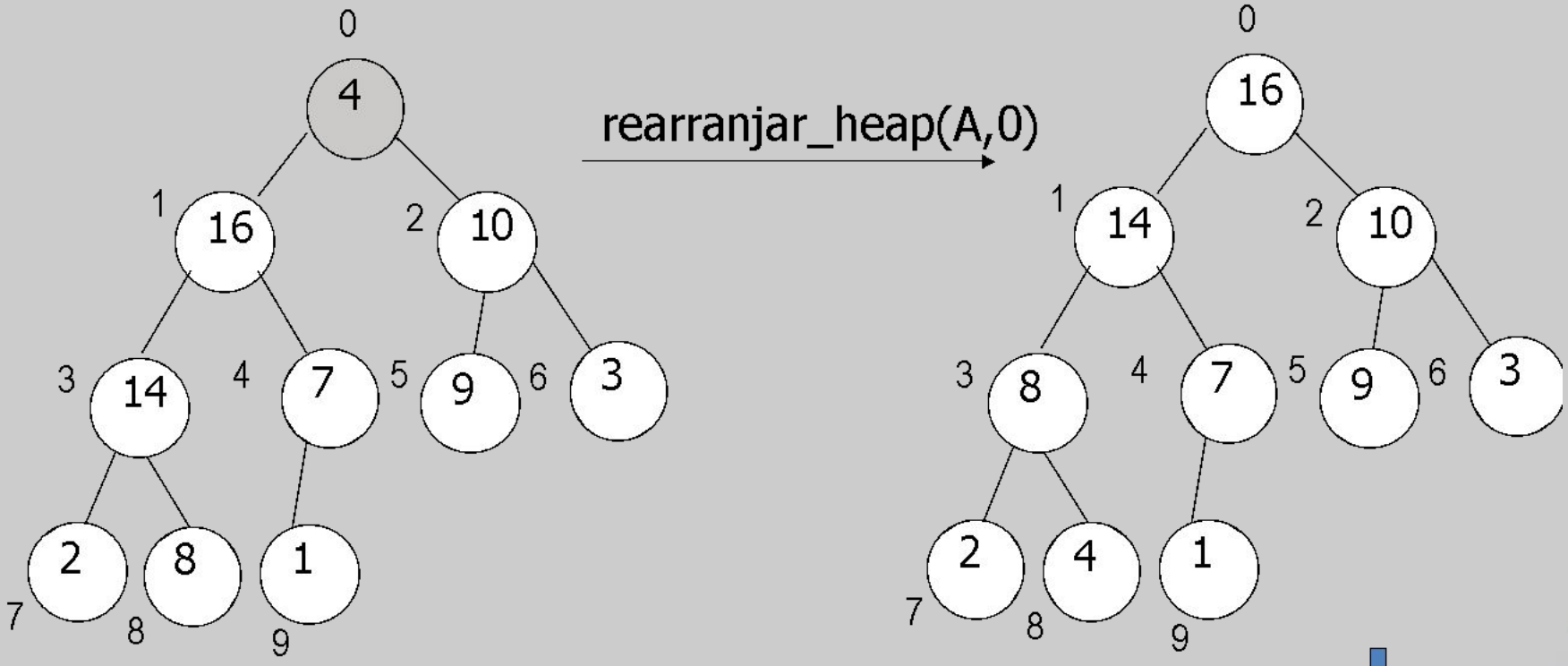
Heap sort


0
4
1
1
2
3
3
14
4
16
5
9
6
10
2
8
7
7
8
9
rearranjar_heap(A,2)
0
4
1
1
2
10
3
14
4
16
5
9
6
3
2
8
7
7
8
9

Heap sort


0
4
1
1
2
10
3
14
4
16
5
9
6
3
7
2
8
8
7
9
rearranjar_heap(A,1)
0
4
1
16
2
10
3
14
4
7
5
9
6
3
7
2
8
8
1
9

Heap sort



0	1	2	3	4	5	6	7	8	9
16	14	10	8	7	9	3	2	4	1

ALGORITMOS E PROGRAMAÇÃO DE COMPUTADORES II

Algoritmos clássicos de ordenação III