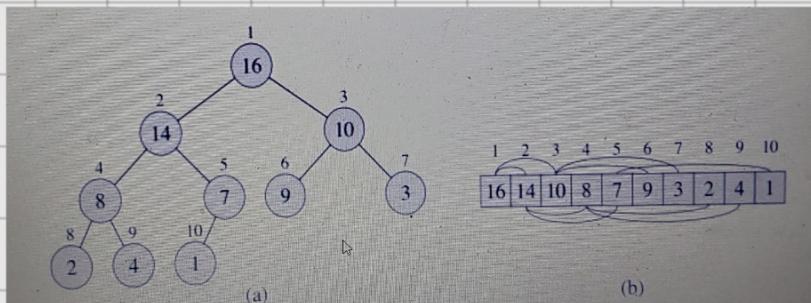
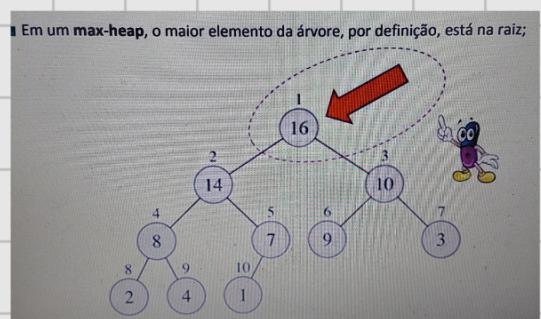


# Heaps e Fila de Prioridade

- Pode ser vista como uma árvore binária praticamente (Quase) ou completa.
- Cada nó da árvore corresponde a um elemento do array que armazena o valor do nó
- Árvore totalmente preenchida em todos os níveis, exceto talvez no nível mais baixo, que é preenchido à esquerda até certo ponto
- São usualmente implementação por meio de arrays

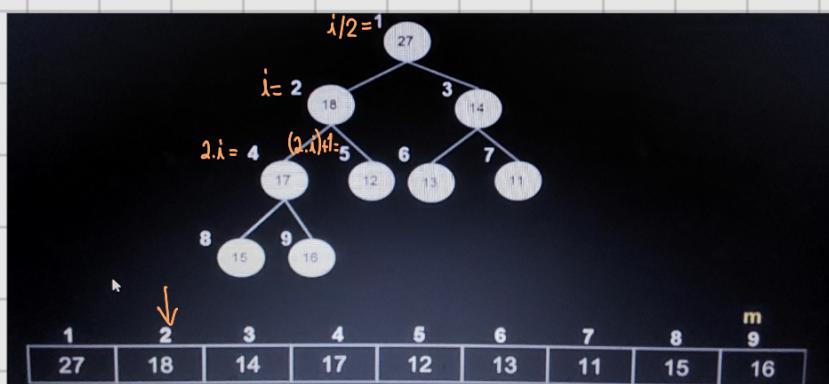
## Propriedades:

- A árvore é completa até o penultimo nível
- No último nível os filhos estão o mais à esquerda possível
- O conteúdo de um nó é maior ou igual ao conteúdo dos nós na subárvore enraizada nele (max-heap)
  - \* Caso o conteúdo seja menor ao conteúdo dos nós é min-heap
- Essas propriedades garantem que um heap pode ser implementado em um array  $A[1 \dots m]$ .



## Implementação:

- Implementa por meio de um array.
- Através do array de implementação, quais os filhos do índice 2?  $2 \cdot i = 4$  e  $(2 \cdot i) + 1 = 5$
- Através do array de implementação, qual o pai do índice 2?  $i/2 = 1$



Dado um nó em um endereço  $i$ , qual o endereço de seu pai?

PARENT( $i$ )

return  $[i/2]$

LEFT( $i$ )

return  $2i$

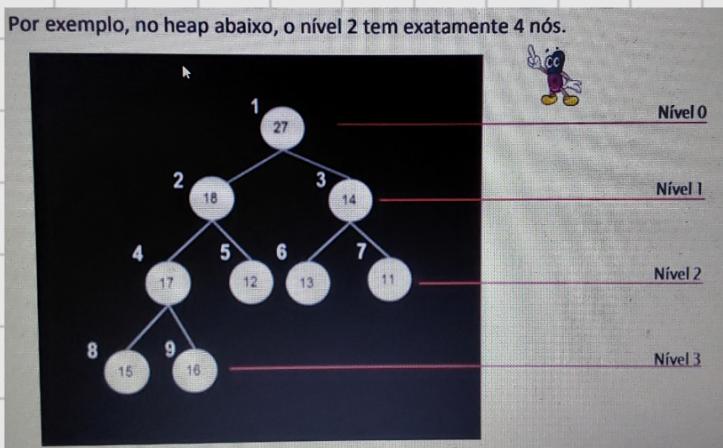
RIGHT( $i$ )

return  $2i + 1$

Detalhes, na página 104 do Cormen!

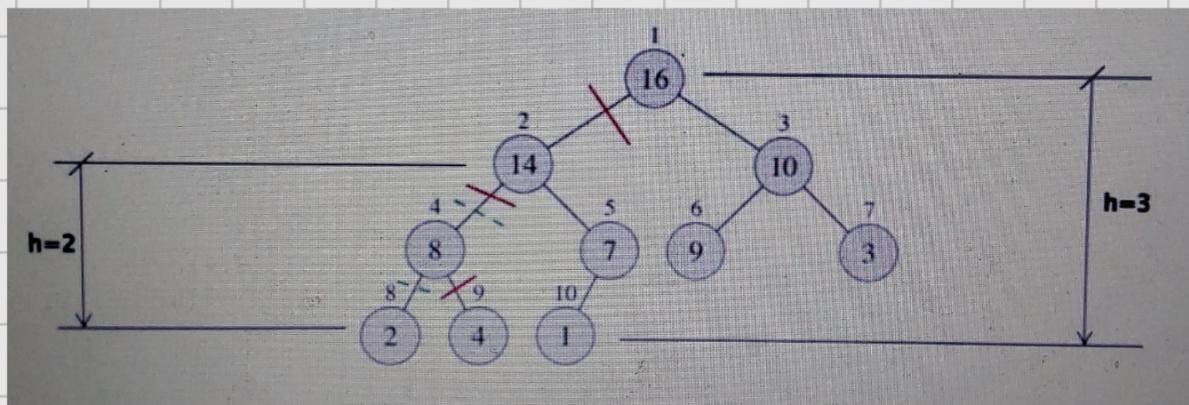
## Nível de um Heap:

- Cada nível  $p$ , tem exatamente  $2^p$  nós, exceto talvez no último nível



## Altura de um Heap:

- A altura de um nó  $i$  é o maior comprimento de um caminho de  $i$  até uma folha, que é, o número de arestas no caminho mais longo desde  $i$  até uma folha



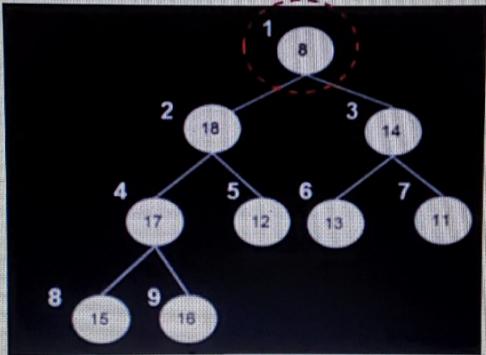
Calcular altura de um nó:  $h = \log_2 (m/i)$

Calcular altura da raiz:  $h = \log_2 (m)$

→ n° total de elementos

# Manutenção do Heap

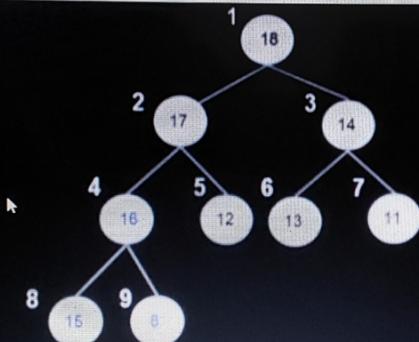
## Manutenção do Heap



O que se deve fazer para que a árvore se torne um heap?

- A ideia seria descer o nó com o valor 8 para baixo, até deixá-lo numa posição conveniente que atenda a propriedade do heap.
- Primeiramente, vamos comparar o 8 com os seus filhos;
- Vamos trabalhar com o maior dos filhos, no caso o nó com o valor 18;
- Proceda-se à troca do 8 com o 18;

## Manutenção do Heap (Heapify)



1	2	3	4	5	6	7	8	9
18	17	14	16	12	13	11	15	8

■ A árvore agora é um heap / heapify

# Algoritmo Max-Heapify

Algoritmo Max-Heapify

## Manutenção da propriedade de heap

```
MAX-HEAPIFY (A, m, i)
1 e ← 2*i
2 d ← 2*i + 1
3 se e ≤ m e A[e] > A[i]
4   então maior ← e
5   senão maior ← i
6 se d ≤ m e A[d] > A[maior ]
7   então maior ← d
8 se maior ≠ i
9   então A[i] ↔ A[maior]
10  MAX-HEAPIFY (A, m, maior)
```

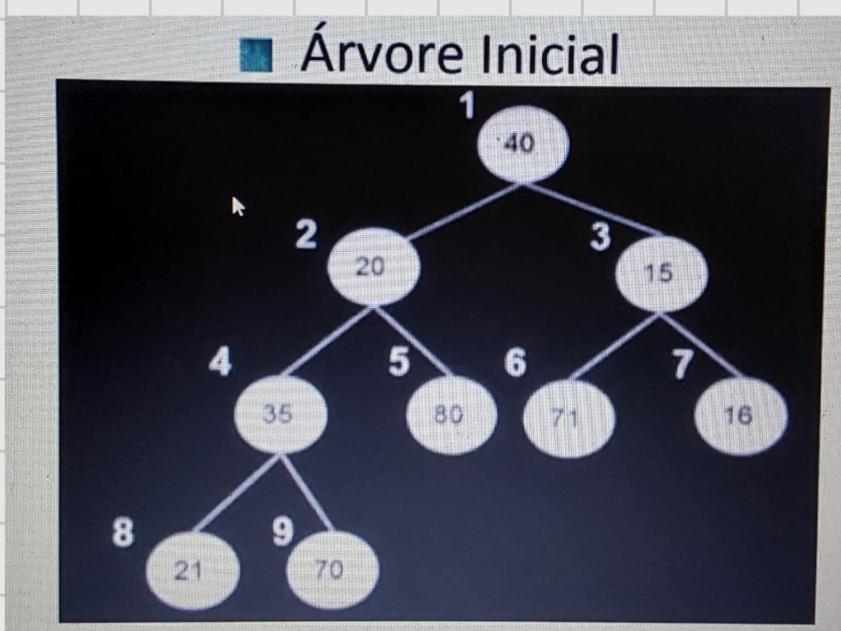
T( $h$ ) $\leq T(h-1) + \Theta(1)$   
T( $h$ ) $= O(h)$   
Como: a altura de um nó  $i$  é:  
 $h = \lfloor \lg(m/i) \rfloor$   
T( $h$ ) $= O(h) = O(\lfloor \lg(m/i) \rfloor)$   
 $= O(\lg(m/i)) = O(\lg(m))$

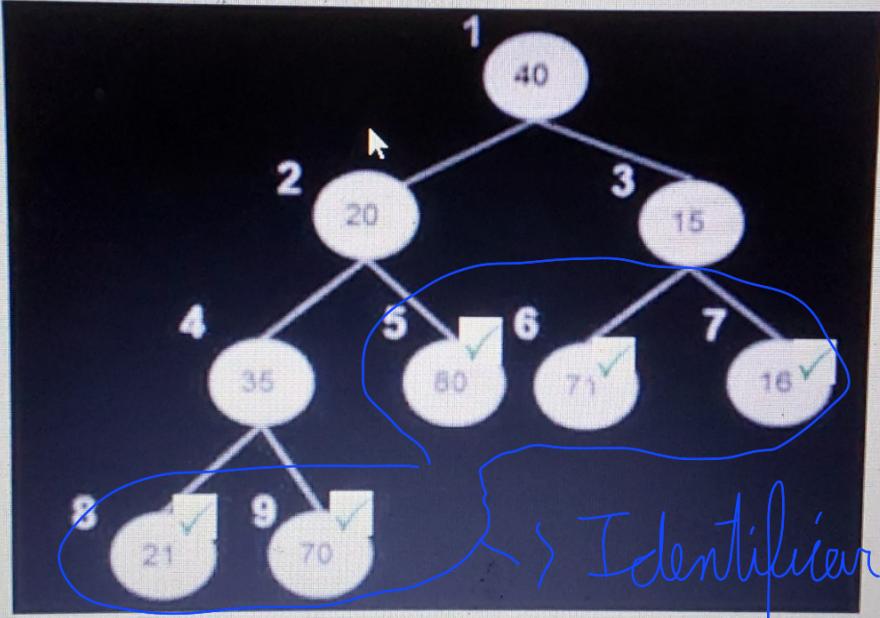
A complexidade do algoritmo Max-Heapify  
é logarítmica  $\Rightarrow O(\log m)$

## Construindo um Max Heap

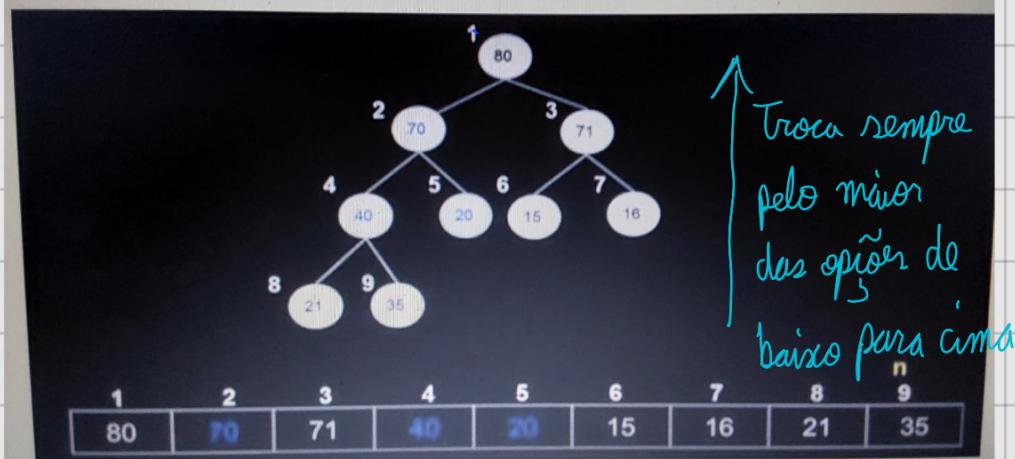
Dado um array de valores, como construir um max heap a partir deste array?

1	2	3	4	5	6	7	8	9
40	20	15	35	80	71	16	21	70





- O processo continua;
- Na verdade, o que se está fazendo é para cada nó não folha, aplica-se o algoritmo max-heapfy, até finalizar-se a árvore com a estrutura de heap.



Exemplo:

