Ciência da Computação

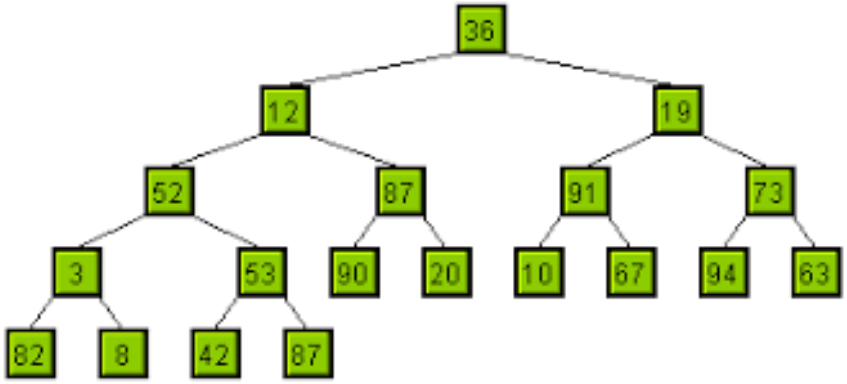
Aula 17
Heapsort

André Luiz Brun

unioeste
Universidade Estadual do Oeste do Paraná

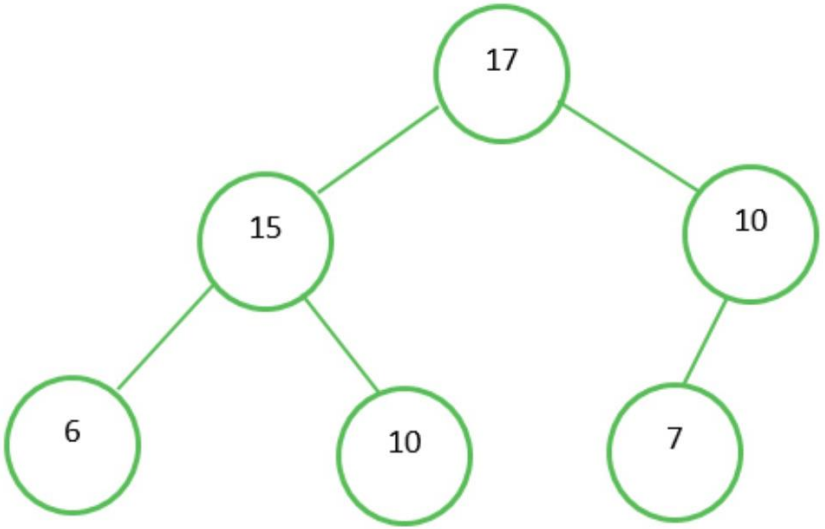
Introdução

- Estrutura de dados que pode ser visualizada como uma árvore binária quase completa (pode ser implementada usando vetores ou árvores)
- A árvore não é completa pois o último nível pode ter alguns ponteiros null (nem todas as folhas estão no mesmo nível)
- O preenchimento do último nível se dá da esquerda para a direita



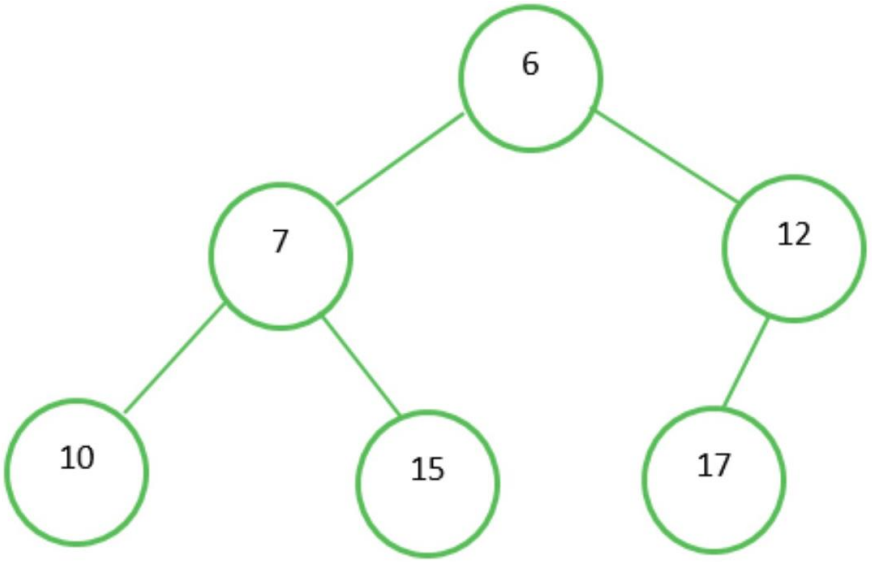
Introdução

- Se estiver trabalhar com um Max-Heap o conteúdo de cada nó será maior ou igual do conteúdo de seus dois filhos



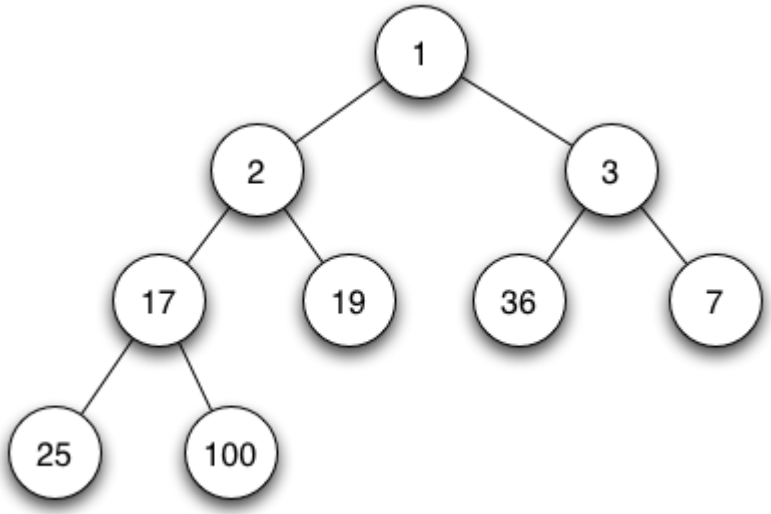
Introdução

- Se estiver trabalhar com um Min-Heap o conteúdo de cada nó será menor ou igual do conteúdo de seus dois filhos



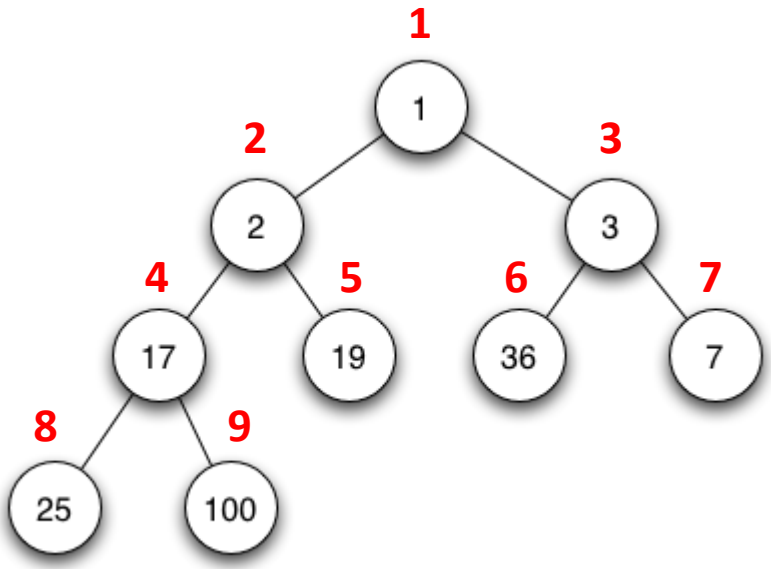
Introdução

- Cada raiz (pai), armazenado na posição i, aponta para um filho da esquerda, armazenado na posição 2i e para um filho da direita na posição 2i+1



Introdução

- Cada raiz (pai), armazenado na posição i, aponta para um filho da esquerda, armazenado na posição 2i e para um filho da direita na posição 2i+1



Introdução

Max-Heap

$V[i] \geq V[2i]$

$V[i] \geq V[2i+1]$

$V[i] \leq V[\frac{i}{2}]$ se i for par

$V[i] \leq V[\frac{i-1}{2}]$ se i for ímpar

Introdução

Min-Heap

$V[i] \leq V[2i]$

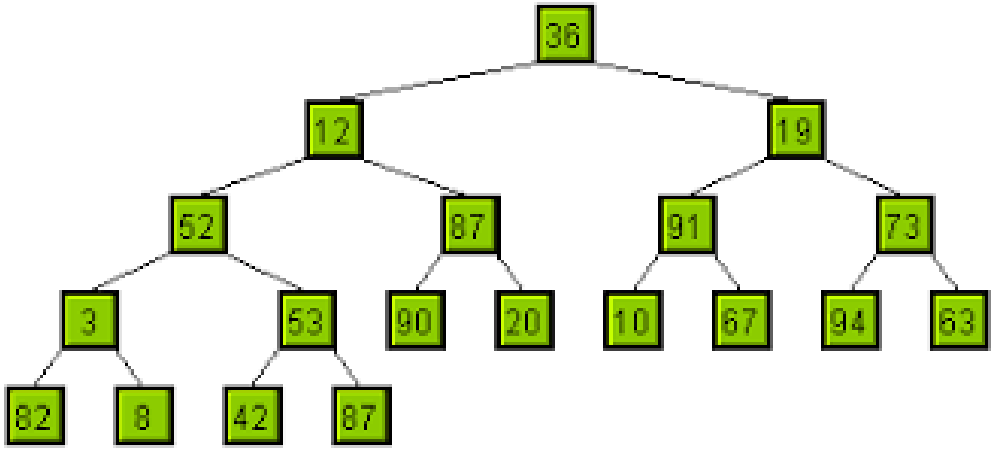
$V[i] \leq V[2i+1]$

$V[i] \geq V[\frac{i}{2}]$ se i for par

$V[i] \geq V[\frac{i-1}{2}]$ se i for ímpar

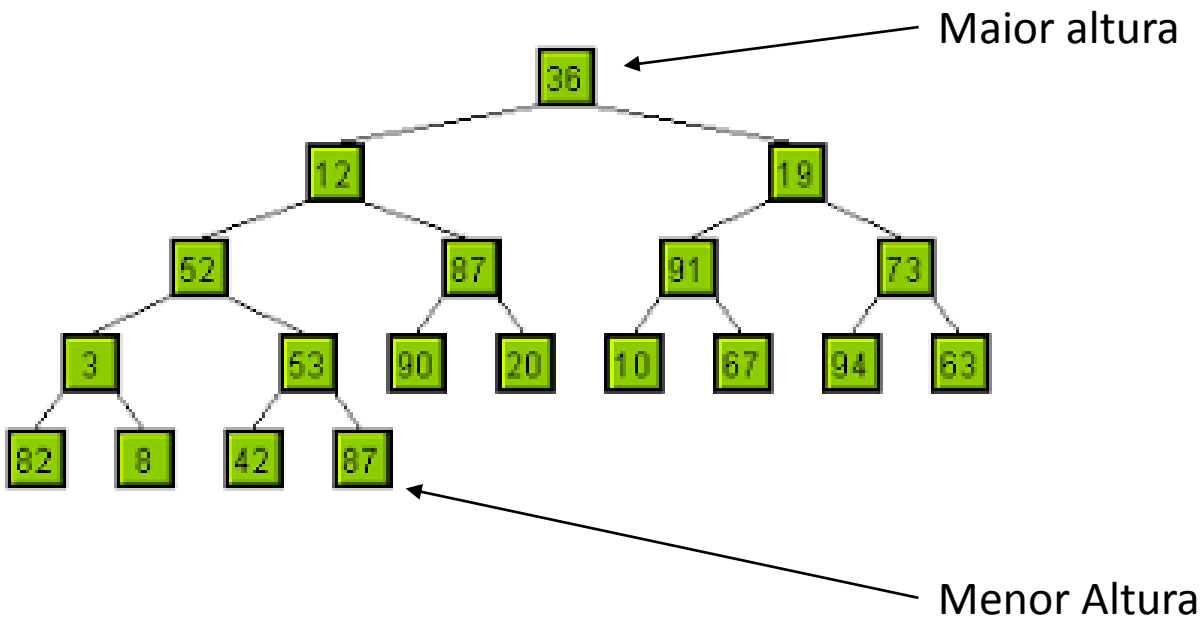
Introdução

Altura



Introdução

Altura

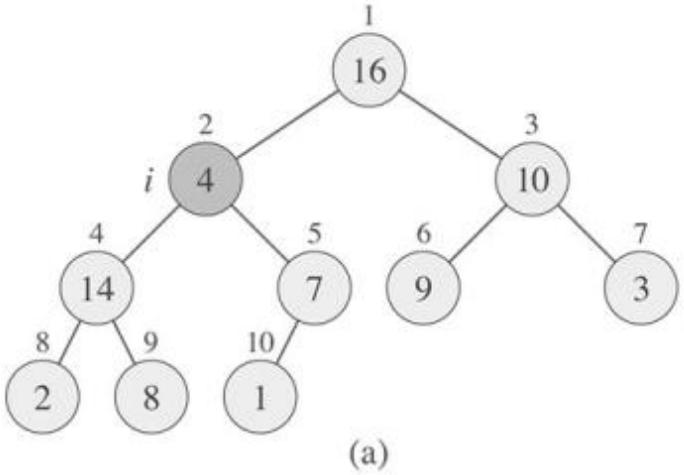


$$h = \log_2 n$$

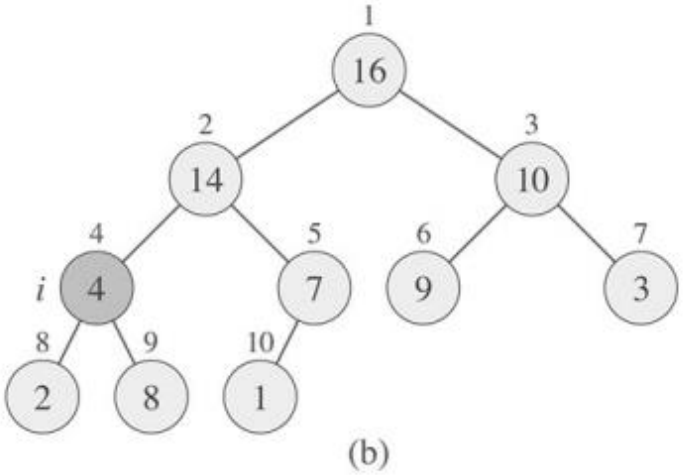
Heapsort

```
MAX-HEAPIFY(A, i)
 1  l = LEFT(i)
 2  r = RIGHT(i)
 3  if l ≤ A·tamanho-do-heap e A[l] > A[i]
 4       maior = l
 5  else maior = i
 6  if r ≤ A·tamanho-do-heap e A[r] > A[maior]
 7     maior = r
 8  if maior ≠ i
 9        trocar A[i] com A[maior]
10        MAX-HEAPIFY(A, maior)
```

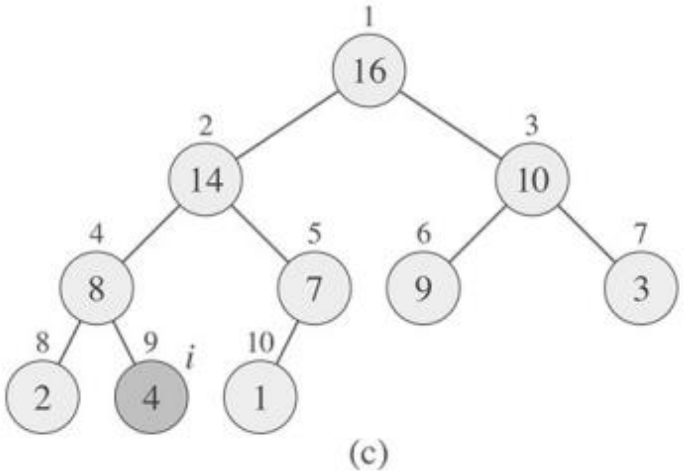
Heapsort



(a)



(b)



(c)

Heapsort

```
BUILD-MAX-HEAP(A)
1  A·tamanho-do-heap = A·comprimento
2  for i = [comprimento[A]/2] downto 1
3      MAX-HEAPIFY(A, i)
```

A	4	1	3	2	16	9	10	14	8	7

Heapsort

```
BUILD-MAX-HEAP(A)
1  A·tamanho-do-heap = A·comprimento
2  for i = [comprimento[A]/2] downto 1
3      MAX-HEAPIFY(A, i)
```

Heapsort

```
HEAPSORT(A)
1 BUILD-MAX-HEAP(A)
2 for i = comprimento[A] downto 2
3     trocar A[1] com A[i]
4         A·tamanho-do-heap = A·tamanho-do-heap – 1
5         MAX-HEAPIFY(A, 1)
```

Heapsort

```
HEAPSORT(A)
1 BUILD-MAX-HEAP(A)
2 for i = comprimento[A] downto 2
3     trocar A[1] com A[i]
4         A·tamanho-do-heap = A·tamanho-do-heap – 1
5         MAX-HEAPIFY(A, 1)
```

A	4	1	3	2	16	9	10	14	8	7