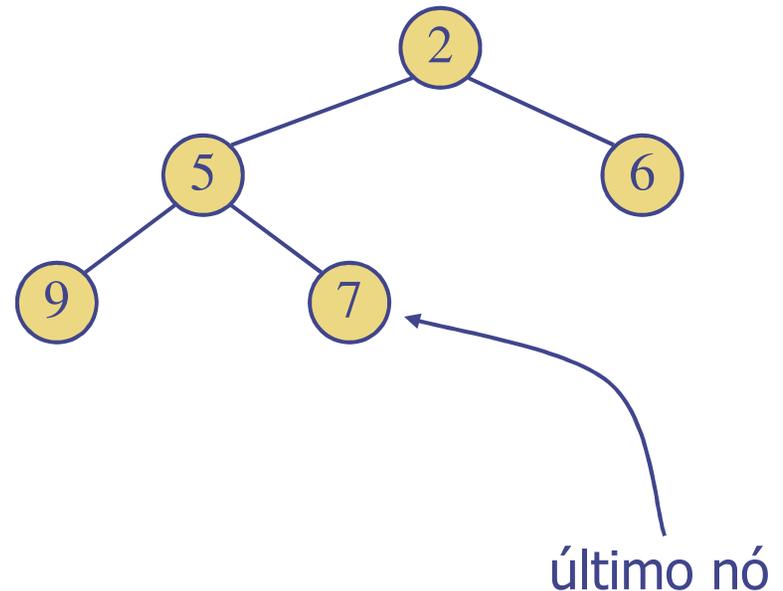


Heap

- ◆ Um *heap* é uma árvore binária armazenando chaves nos nós e satisfazendo as seguintes propriedades
 - **Heap-Order:** Para todo nó interno v , que não a raiz, $key(v) \geq key(parent(v))$
 - **Árvore binária completa:** Seja h a altura de um heap
 - ◆ para $i = 0, \dots, h - 1$, existem 2^i nós na profundidade i
 - ◆ Na altura h , Os nós internos estão a esquerda dos nós externos

- ◆ O último nó de um heap é o mais a direita da altura h

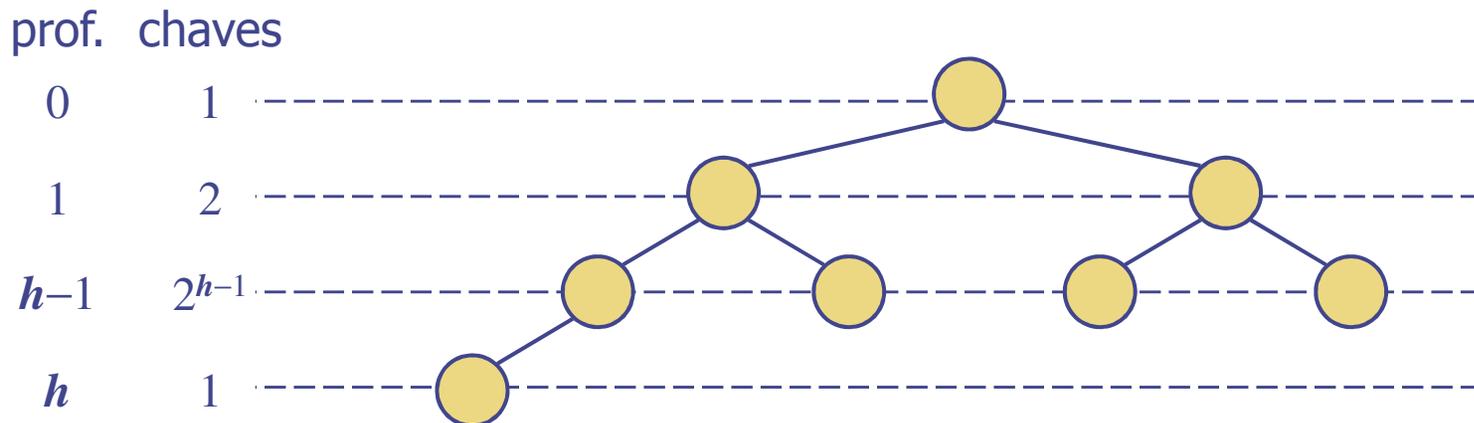


Altura de um heap

◆ **Teorema:** Um heap armazenando n chaves tem altura $O(\log n)$

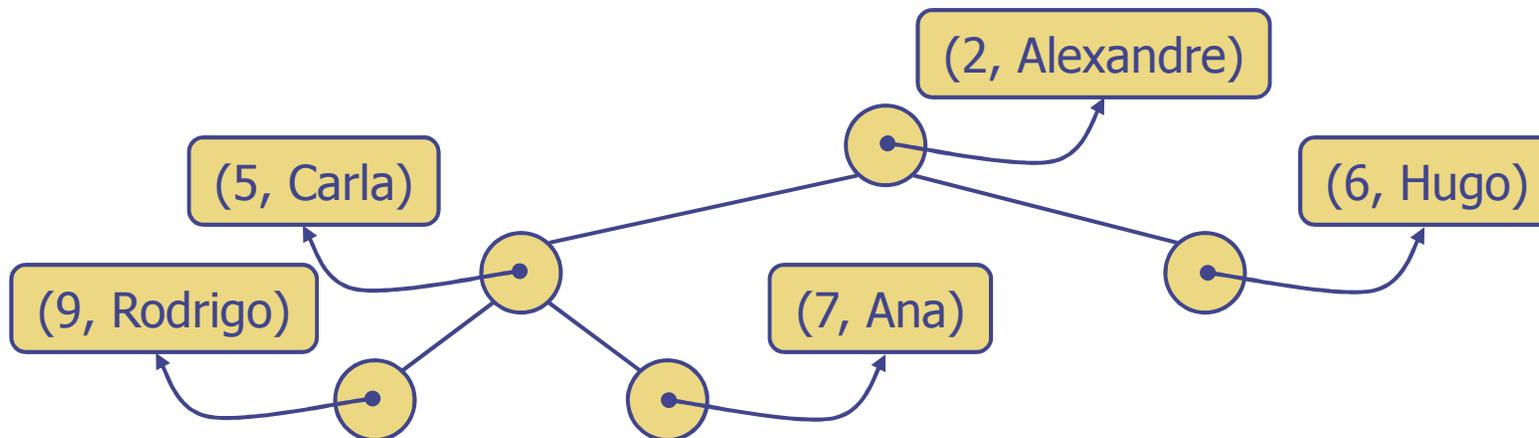
Prova: (Aplicamos a propriedade da árvore binária completa)

- Seja h a altura de um heap com n chaves
- Como existem 2^i chaves na profundidade $i=0, \dots, h-1$ e pelo menos uma chave na profundidade h , temos que $n \geq 1 + 2 + 4 + \dots + 2^{h-1} + 1$
- Dessa forma, $n \geq 2^h$, i.e., $h \leq \log n$



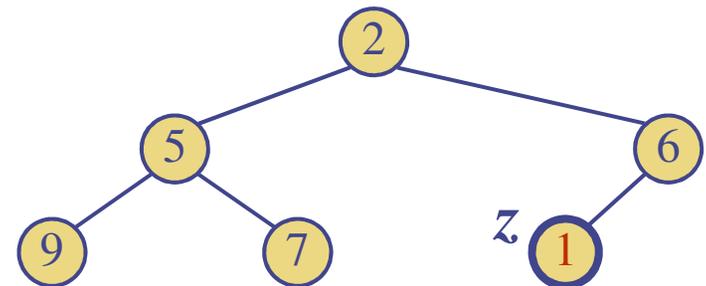
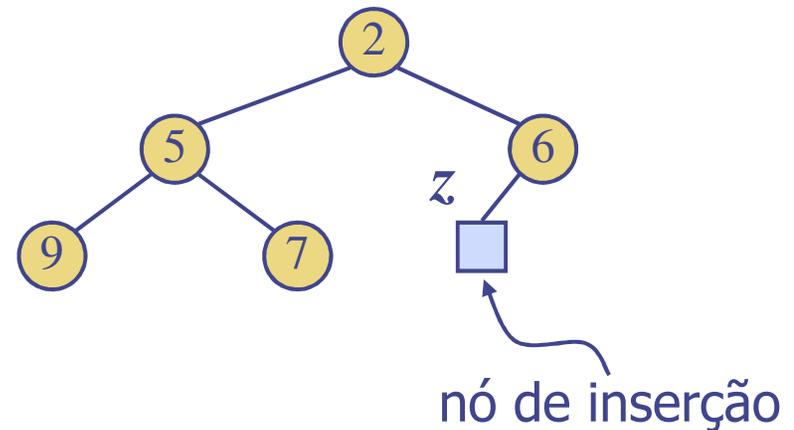
Heaps e filas de prioridade

- ◆ Podemos usar um heap para implementar uma fila de prioridade
- ◆ Armazenamos um item (chave, elemento) em cada nó
- ◆ Sabemos qual o último nó
- ◆ Por simplificação, mostraremos apenas as chaves nos nós



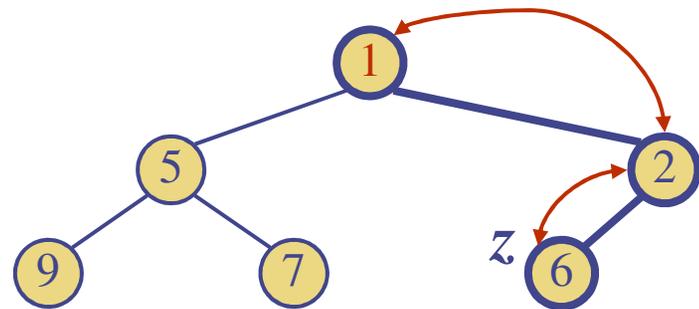
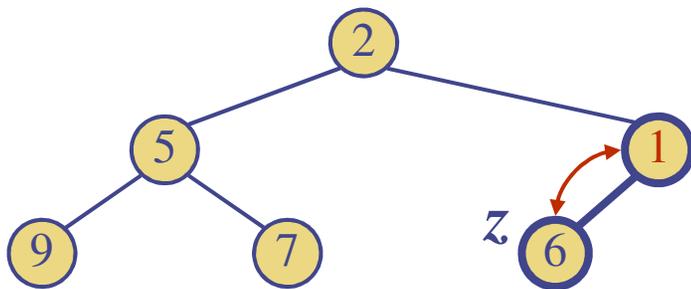
Inserção em uma heap

- ◆ O método insert do TAD FilaDePrioridade corresponde a inserção de uma chave k no heap
- ◆ O algoritmo de inserção consiste de 3 passos
 - Encontrar o nó de inserção z (último nó)
 - Armazenar k em z
 - Restaurar a heap-order



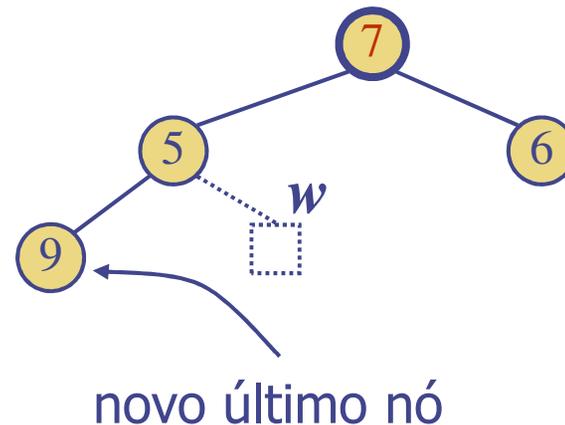
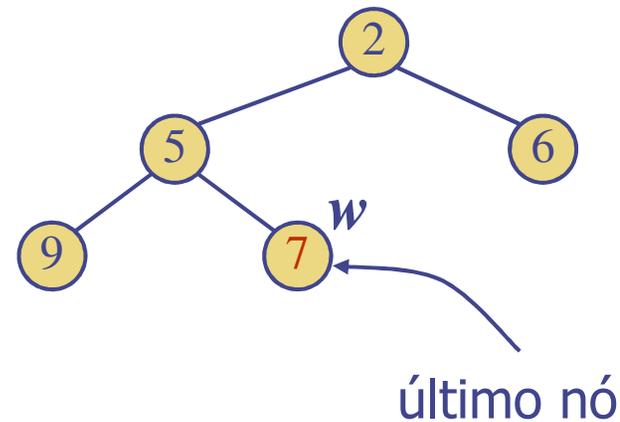
Upheap

- ◆ Após a inserção de uma nova chave k , a propriedade heap-order pode estar violada
- ◆ O algoritmo upheap restaura a propriedade heap-order trocando k sobre o "caminho acima" a partir do nó de inserção
- ◆ Upheap termina quando a chave k encontra o nó raiz ou um nó cujo pai possua uma chave menor ou igual a k
- ◆ Como um heap tem altura $O(\log n)$, upheap roda em tempo $O(\log n)$



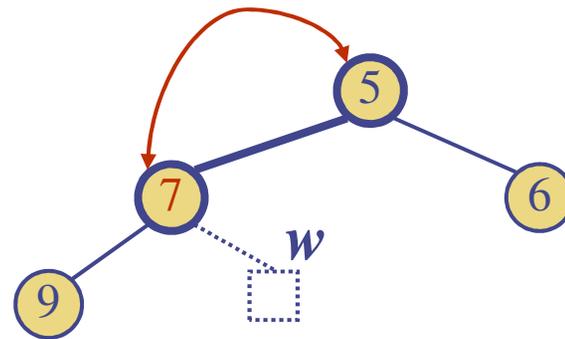
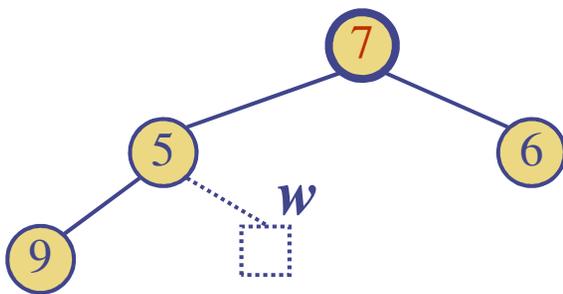
Remoção de um heap

- ◆ O Método `removeMin` de uma fila de prioridade corresponde a remoção de uma raiz de um heap
- ◆ Esta remoção consiste de 3 passos
 - Substituir a chave da raiz com a chave no último nó w
 - Eliminar w
 - Restaurar a heap-order



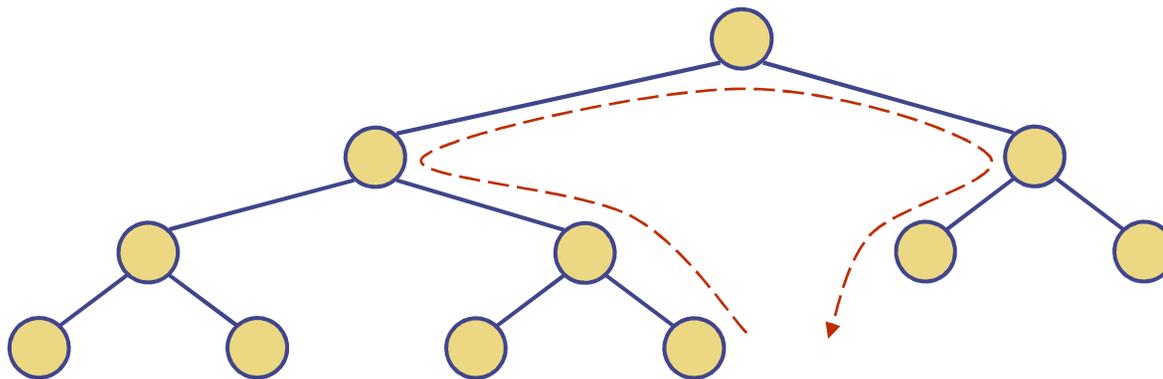
Downheap

- ◆ Depois de substituir a chave da raiz com a chave do último nó, a propriedade heap-order pode estar violada
- ◆ O algoritmo downheap restaura esta propriedade trocando a chave k sobre o "caminho abaixo" da raiz
- ◆ Downheap termina quando a chave k encontra uma folha ou um nó cuja chave é maior do que k
- ◆ Como um heap tem altura $O(\log n)$, downheap roda em tempo $O(\log n)$



Atualizando o último nó

- ◆ O nó de inserção pode ser encontrado atravessando um caminho de $O(\log n)$ nós
 - Vá acima até um filho da esquerda ou a raiz for encontrada
 - Se um filho da esquerda é encontrado, vá para o filho da direita
 - Vá para baixo pela esquerda até encontrar uma folha
- ◆ Algoritmo similar ao de atualizar o último nó após uma remoção

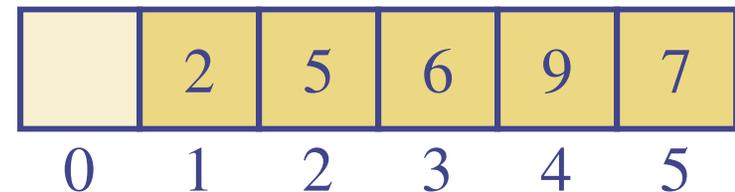
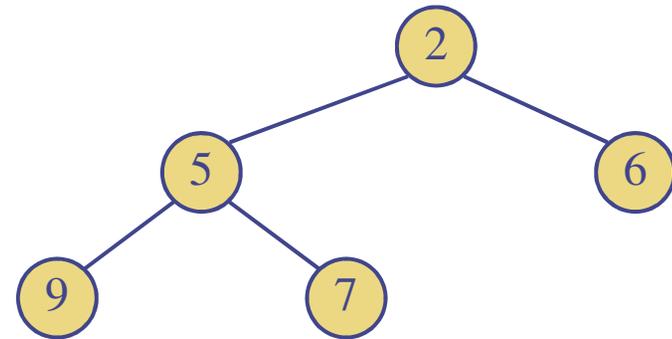


Heap-Sort

- ◆ Considere uma fila de prioridade com n itens implementado com um heap
 - O espaço usado é $O(n)$
 - métodos **insert** e **removeMin** rodam em tempo $O(\log n)$
 - métodos **size**, **isEmpty**, e **min** rodam em tempo $O(1)$
- ◆ Usando uma fila de prioridade baseada em heap, podemos ordenar uma sequência de n elementos em tempo $O(n \log n)$
- ◆ O algoritmo é chamado de *heap-sort*
- ◆ *heap-sort* é muito mais rápido do que algoritmos quadráticos, como inserção e seleção

Implementação com Vetor

- ◆ Podemos representar um heap com n chaves usando um vetor de tamanho $n + 1$
- ◆ Para o nó na colocação i
 - o filho esquerdo está em $2i$
 - o filho direito está em $2i + 1$
- ◆ “Ligações” entre os nós não são explicitamente armazenadas
- ◆ A colocação 0 não é usada
- ◆ A operação insertI corresponde a inserir na colocação $n + 1$
- ◆ Operação removeMin corresponde a remover da colocação n
- ◆ Pode ser usado para heap-sort “in-place”



Juntando dois heaps

- ◆ Temos dois heaps e uma chave k
- ◆ Criamos um novo heap com a raiz armazenando k e com os dois heaps com subárvores
- ◆ Aplicamos downheap para restaurar a heap-order

