Programação 2 _ T08

Filas de Prioridade. Heaps.

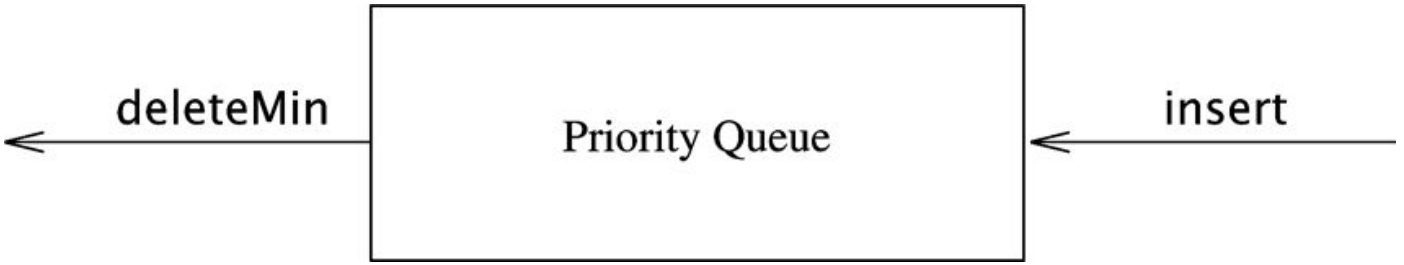
Rui Camacho
(slides por Luís Teixeira)
MIEEC 2020/2021

FILA DE PRIORIDADE

Fila
É uma estrutura de dados linear, do tipo **FIFO (First-In-First-Out)**, que capta a noção de **ordem de chegada**.
No processamento de tarefas, além da **ordem de chegada** é típico atender à **prioridade.**

Fila de Prioridade
Elementos na fila têm um **número indicativo da sua prioridade:**
- elementos podem ser inseridos em qualquer ordem
- mas são apenas removidos de acordo com a sua **prioridade**

FILA DE PRIORIDADE



O elemento de **máxima prioridade** está sempre à frente na fila e é sempre o **primeiro a ser removido**.

O elemento de **máxima prioridade** é o **elemento mínimo** (ou o **máximo**, dependendo da implementação)

APLICAÇÕES DE FILAS DE PRIORIDADE

Processamento de tarefas nos sistemas operativos

Simulação baseada em eventos

Fila de impressão

Ordenação (Heapsort)

etc.

O CONCEITO DE PRIORIDADE

Considerando o seguinte mapa de exames (fictício) como organizaria o seu estudo para os exames?

Unidade Curricular	Data do exame	Prioridade
Programação 2	6 Junho	2
Análise Matemática	26 Junho	4
Circuitos	31 Maio	1
Física	12 Junho	3

A prioridade é estabelecida com base nas datas dos exames.

É necessário comparar todas as datas para estabelecer a prioridade.

IMPLEMENTAÇÕES DA FILA DE PRIORIDADE

A implementação deve considerar o custo das operações fundamentais da fila de prioridade: **inserir** novos elementos e **remover** o elemento mínimo.

Lista ligada **não ordenada**

- inserção na cabeça da lista: **O(1)**
- remover o mínimo: **O(N)**

Lista ligada **ordenada**

- inserção: **O(N)**
- remover o mínimo: **O(1)**

Árvore binária de pesquisa

- Caso médio para ambas as operações: **O(log N)**

HEAP BINÁRIO

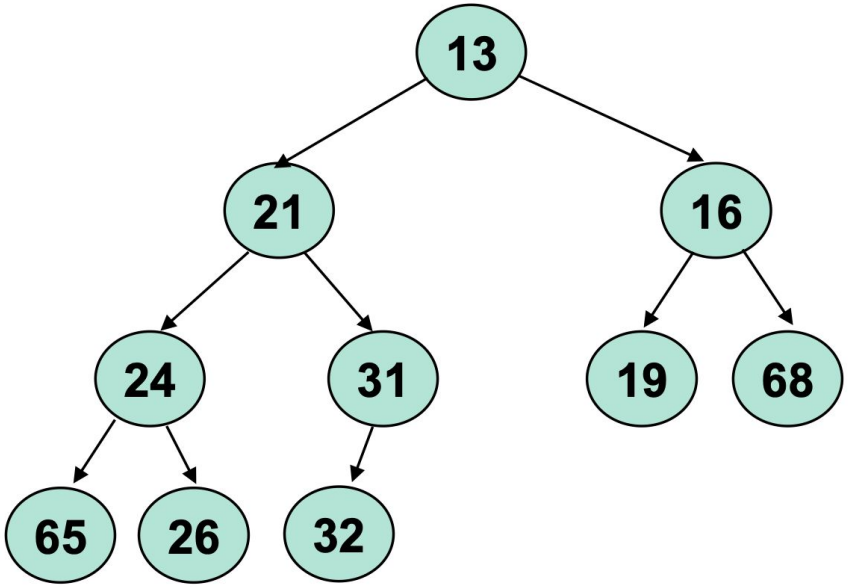
O **heap binário** é uma estrutura de dados que apresenta as seguintes duas propriedades fundamentais:

1. **Estrutura:** corresponde a uma árvore binária completa (representação implícita em vetor)
2. **Ordem:** dado um **nó X** com o **pai P** na árvore, a chave de P é **menor ou igual** à chave de X (no caso da ***min-heap*** ; **maior ou igual** no caso da ***max-heap***)

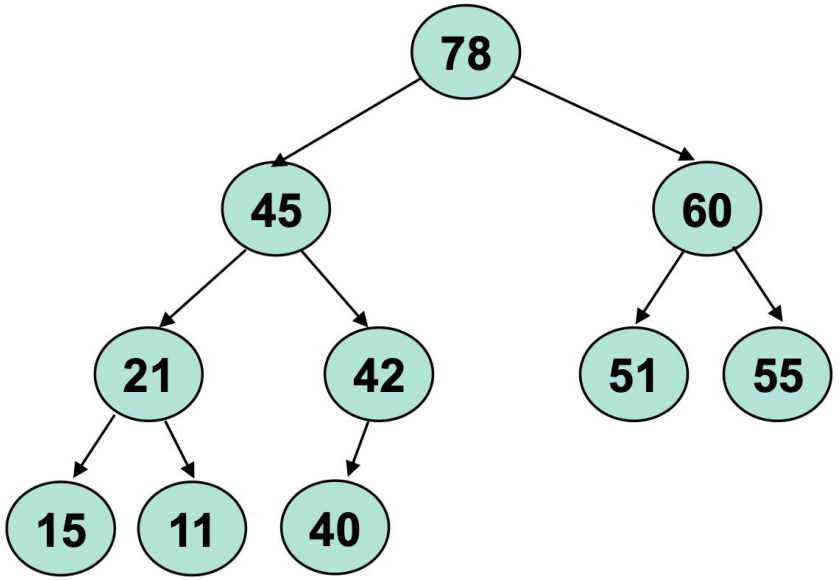
Conveniente para implementar a fila de prioridade

- Pode ser suportado num vetor
- **Inserir e remover o mínimo** (máximo na *max-heap*)**:** O(log N) no pior caso
- **Inserir:** tempo médio constante; encontrar o mínimo: tempo constante no pior caso

MIN-HEAP E MAX-HEAP



Min-heap



Max-heap

HEAP BINÁRIO

Implementado como uma árvore binária com duas propriedades adicionais: uma **propriedade estrutural** e uma **propriedade de ordem**

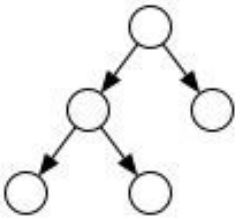
Propriedade estrutural:

- todos os níveis da árvore estão preenchidos exceto o último
- o último nível é preenchido da esquerda para a direita
- **árvore binária completa**

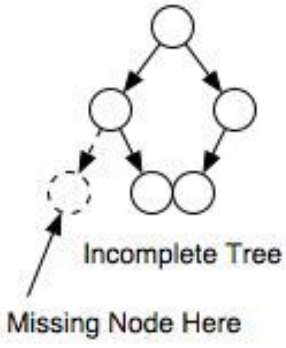
Propriedade de ordem (*min-heap*):

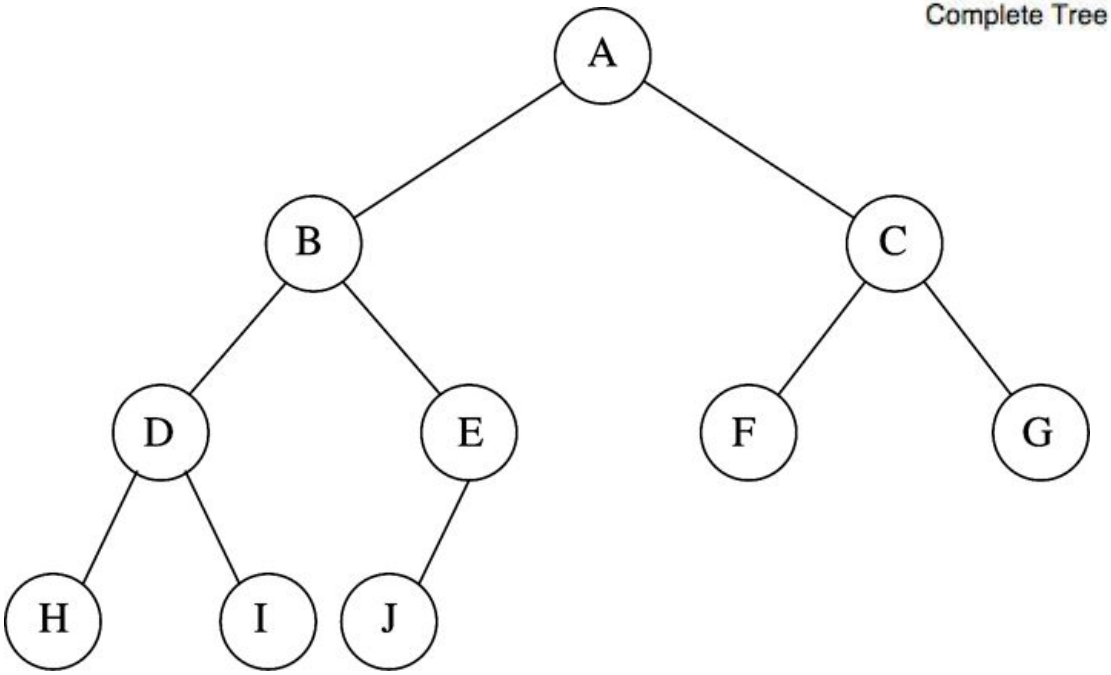
- para todos os nós X (excepto a raíz) o pai de X é menor ou igual a X
- o menor elemento está sempre na raíz da árvore
- à semelhança das árvores AVL as operações de inserção e remoção podem requerer **pós-processamento para manter as propriedades de heap**

ÁRVORE BINÁRIA COM



Complete Tree





Implementação da árvore binária completa baseada em vetor:

	A	B	C	D	E	F	G	H	I	J			
0	1	2	3	4	5	6	7	8	9	10	11	12	13

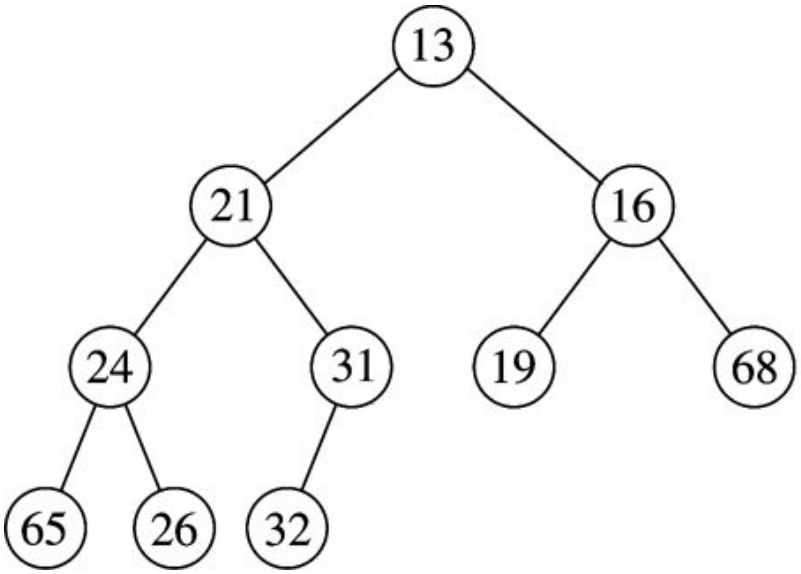
IMPLEMENTAÇÃO DE HEAP BINÁRIO

```
struct HeapStruct {
    int Capacity;
    int Size;
    ElementType *Elements; /* vetor dinâmico que irá conter os
              elementos da fila de prioridade */
};

typedef struct HeapStruct *PriorityQueue;
```

IMPLEMENTAÇÃO DE HEAP BINÁRIO

```
#define MinData (-32767)

PriorityQueue Initialize(int MaxElements) {
    PriorityQueue H;

    H = malloc(sizeof ( struct HeapStruct));
    if (H == NULL) { printf("Out of space!!!"); return NULL; }

    H->Elements = malloc((MaxElements + 1) * sizeof ( ElementType));
    if (H->Elements == NULL) { printf("Out of space!!!"); return NULL; }

    H->Capacity = MaxElements;
    H->Size = 0;
    H->Elements[ 0 ] = MinData;

    return H;
}
```

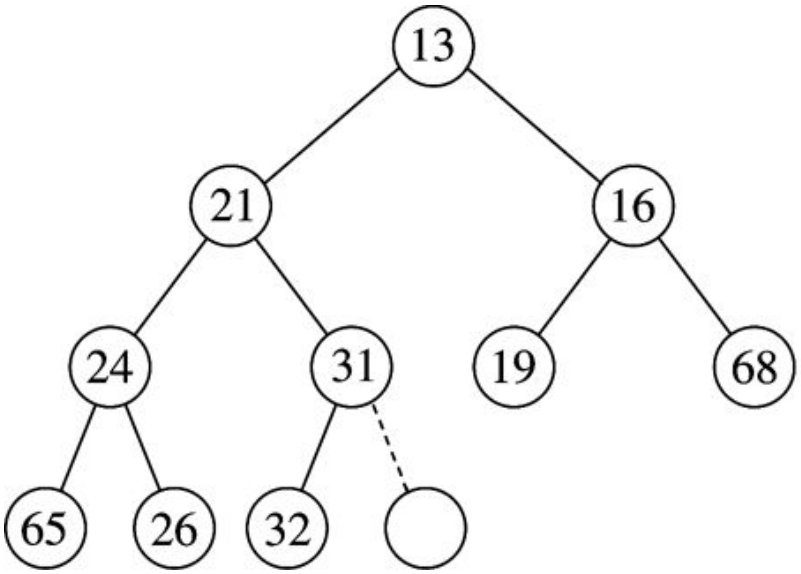
EXEMPLO DE INSERÇÃO EM HEAP



Pretende-se inserir o elemento **14** neste heap.

Onde será inserido o elemento, tendo em conta as propriedades de **estrutura** e de **ordem**?

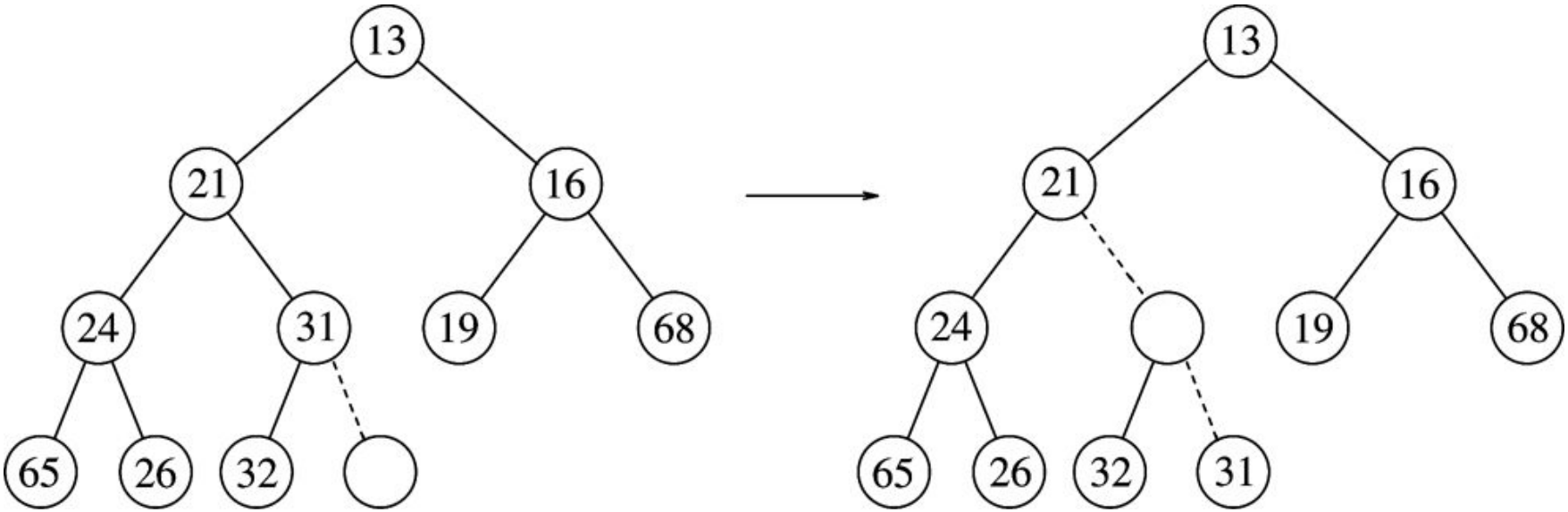
EXEMPLO DE INSERÇÃO EM HEAP



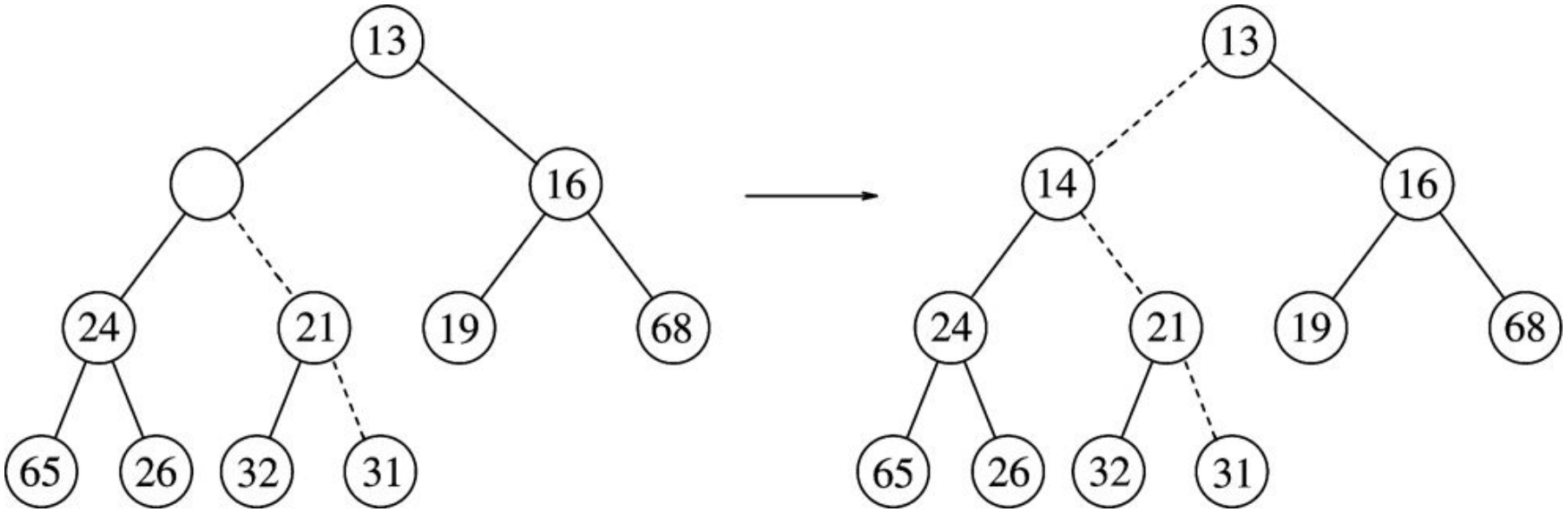
Tenta-se inserir o 14 na heap atendendo à propriedade estrutural.

Depois, com base na propriedade de ordem vai-se trocando a posição do elemento 14 com a do nó pai até se encontrar a posição correcta.

EXEMPLO DE INSERÇÃO EM HEAP



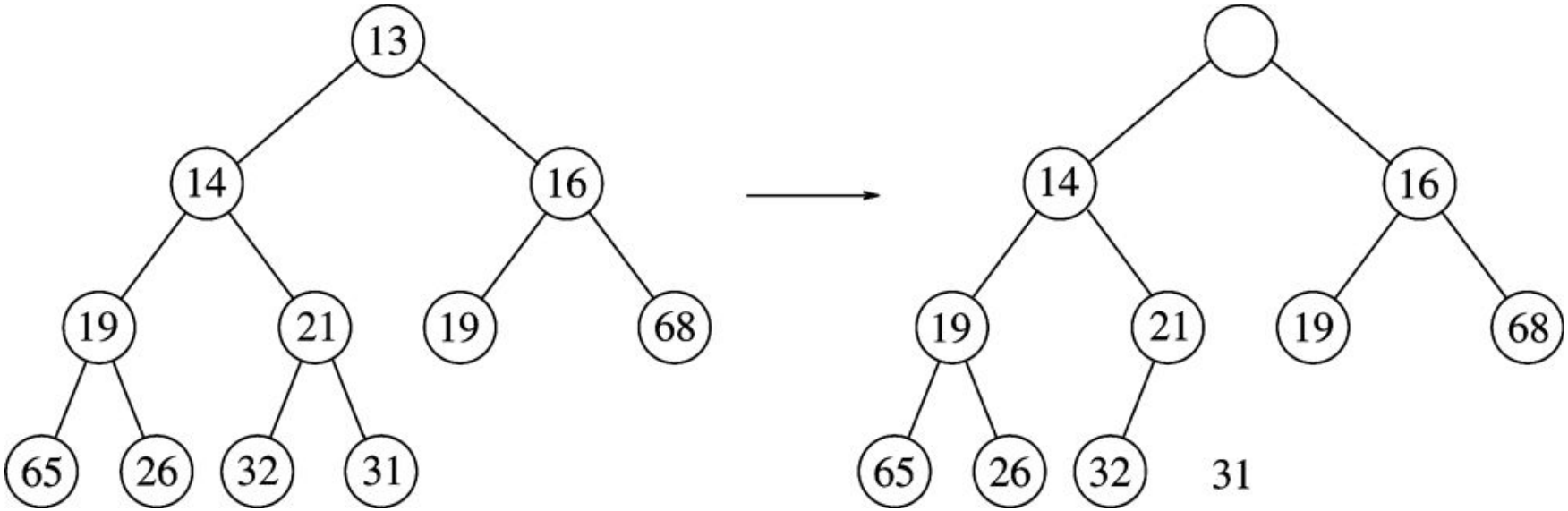
EXEMPLO DE INSERÇÃO EM HEAP



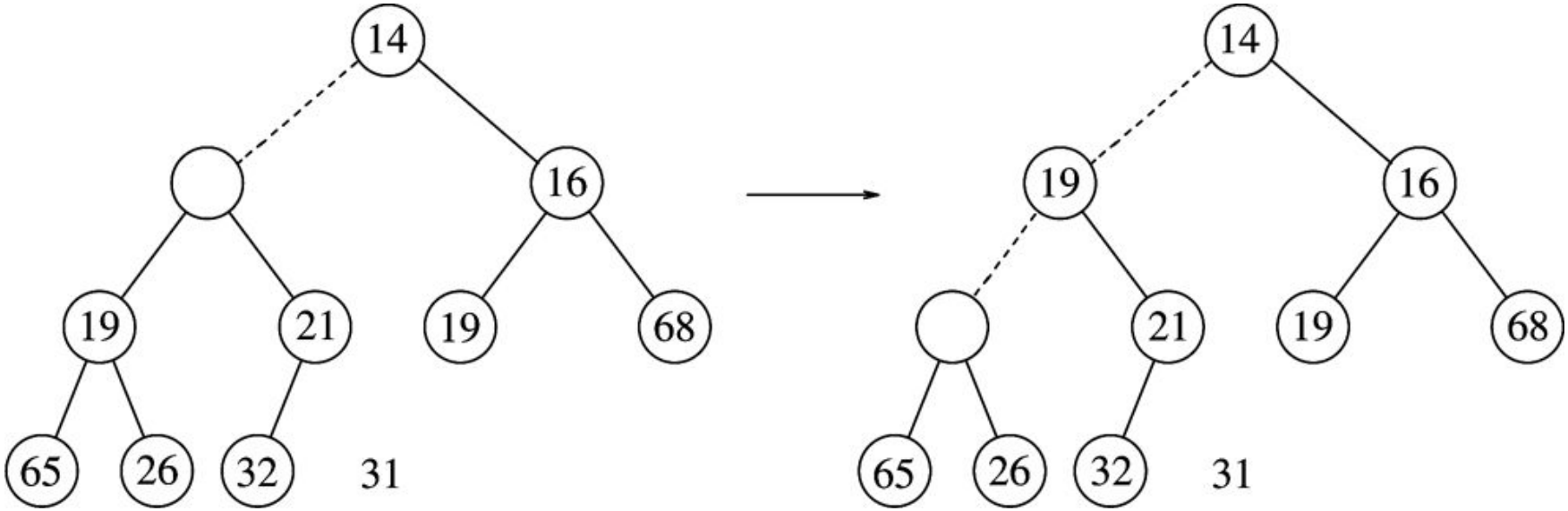
IMPLEMENTAÇÃO DE HEAP BINÁRIO

```
void Insert(ElementType X, PriorityQueue H) {
    int i;

    if (IsFull(H)) {
        printf("Priority queue is full");
        return;
    }

    H->Size++;

    for (i = H->Size; i > 1 && H->Elements[ i / 2 ] > X; i /= 2)
        H->Elements[ i ] = H->Elements[ i / 2 ];

    H->Elements[ i ] = X;
}
```

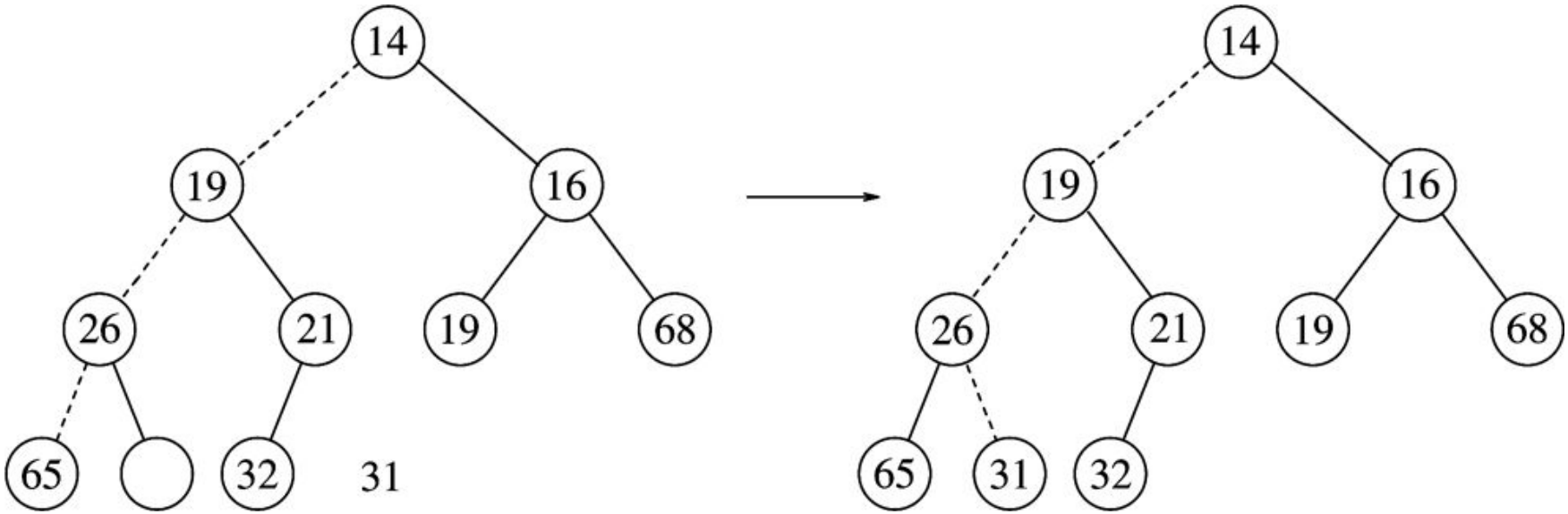
EXEMPLO DE REMOÇÃO DO MÍNIMO



EXEMPLO DE REMOÇÃO DO MÍNIMO



EXEMPLO DE REMOÇÃO DO MÍNIMO



D-HEAP