Algoritmos e Estrutura de Dados II

Prof. Fellipe Guilherme Rey de Souza

Aula 24 – Listas de Prioridades

Agenda

- Introdução
- Lista não-ordenada
- Lista ordenada
- Heap
- Alteração de Prioridades

Introdução

- As motivações para uma lista de prioridades são:
  i. Os dados possuem prioridades.
  ii. A prioridade de um dado pode variar ao longo do tempo.
  iii. Quando desejado, deve-se selecionar o dado de maior prioridade.

- A situação acima modela uma grande quantidade de problemas.

Introdução

- A Lista de Prioridades é uma tabela, na qual a cada um de seus dados está associada uma prioridade. Em geral, a prioridade é um valor numérico.

- Operações básicas:
  - Seleção do elemento de maior prioridade
  - Inserção de um novo dado
  - Remoção do dado de maior prioridade

Introdução

- Uma Lista de Prioridades pode ser implementada por vários métodos.

  São alguns deles:

  i. Lista não ordenada

  ii. Lista ordenada

  iii. Heap

Introdução

- Para cada um desses métodos será avaliada a complexidade de cada uma das seguintes operações:
  - Seleção (busca)
  - Inserção
  - Remoção
  - Alteração
  - Construção

Lista não-ordenada

- Os dados formam uma lista não ordenada com n nós.

- Complexidades:

Operação	Complexidade
Seleção	$O(n)$
Inserção	$O(1)$
Remoção	$O(n)$
Alteração	$O(n)$
Construção	$O(n)$

Lista ordenada

- Os dados formam uma lista ordenada, em ordem decrescente de suas prioridades.

- As operações de seleção e remoção referem-se sempre ao dado de maior prioridade. Para construir a lista, é necessário ordená-la.

Lista ordenada

- Complexidades:

Operação	Complexidade
Seleção	*O(1)*
Inserção	*O(n)*
Remoção	*O(1)*
Alteração	*O(n)*
Construção	*O(n log n)*

Heap

- Um heap é uma lista linear composta de elementos com chaves $s_1, \ldots, s_n$, satisfazendo:

$$\boldsymbol{s_1 \leq s_n}, 1 \leq i \leq n$$

- A chave representa a prioridade do elemento.

Heap

- Um heap pode ser visualizado através de uma árvore binária completa *T*.
  - Os nós de *T* são numerados seqüencialmente, da raiz para os nós
  - Cada nó de *T* corresponde a uma chave, sendo o rótulo do nó igual à prioridade da chave.
  - Os nós do último nível de *T* são preenchidos da esquerda para a direita.

Heap

- Complexidades:

Operação	Complexidade
Seleção	*O(1)*
Inserção	*O(log n)*
Remoção	*O(log n)*
Alteração	*O(log n)*
Construção	*O(n)*

- A seleção é O(1) porque sempre iremos pegar o elemento com maior prioridade (raiz da árvore, ou seja, a posição 0 do vetor).

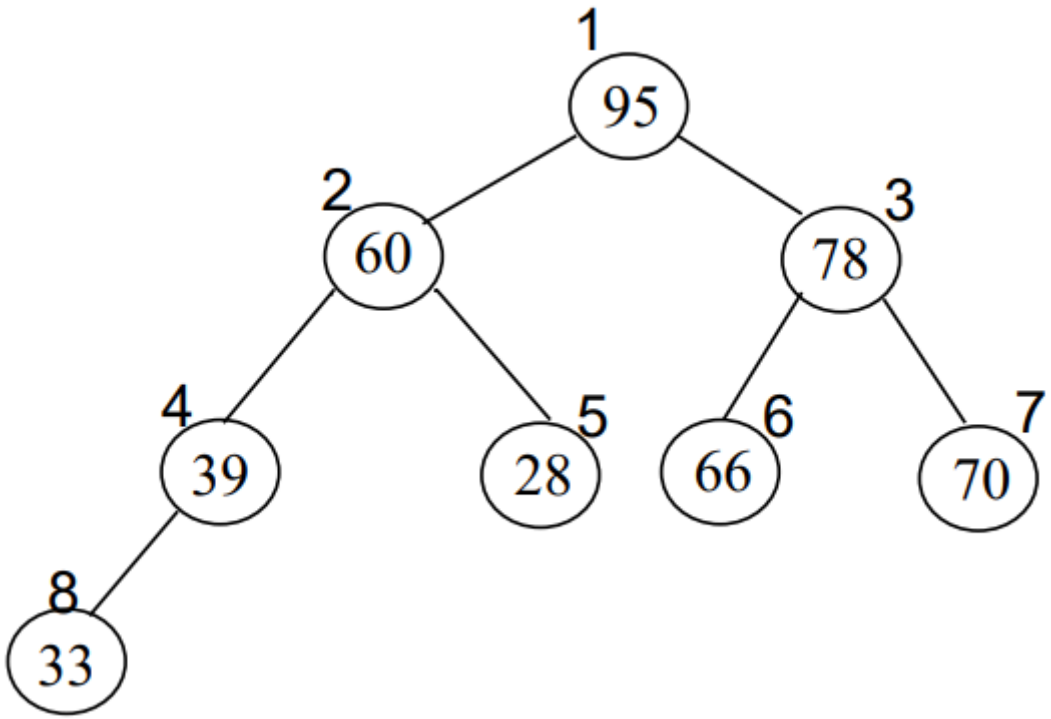
Alteração de Prioridades

- Aumento ou diminuição de prioridade de um nó.
  - O aumento está associado à "subida"do nó, na árvore binária correspondente.
  - A diminuição está associada à "descida"do nó, na árvore binária correspondente.

- A subida e a descida de nós na árvore serão realizadas sempre através de caminhos.

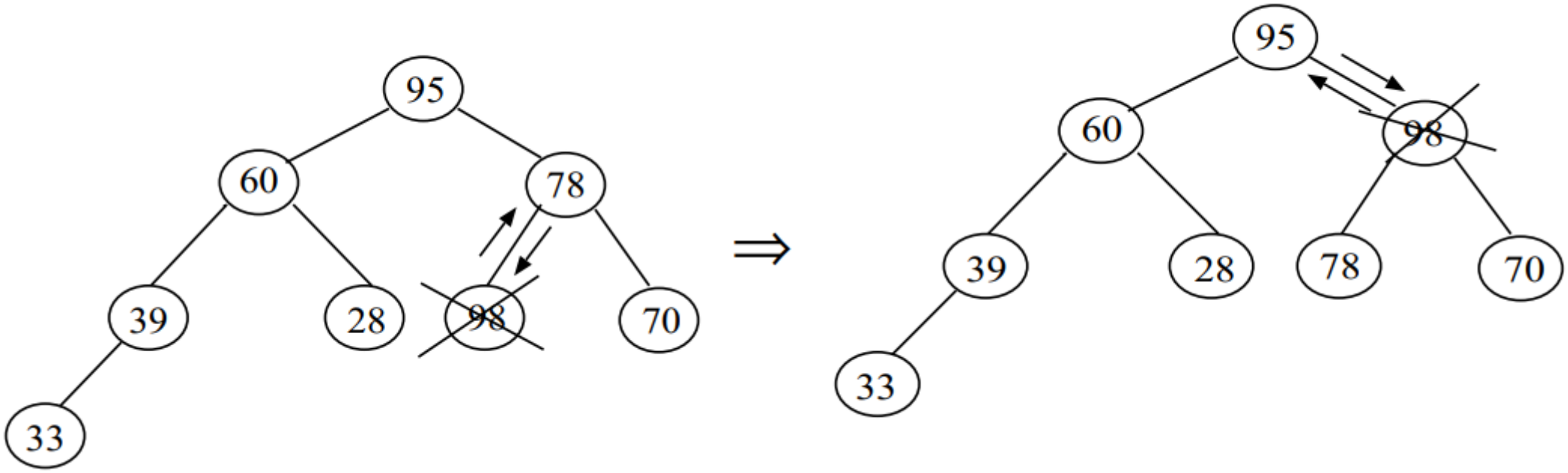
Alteração de Prioridades

- Exemplo: Alterar a prioridade do nó 6, de 66 para 98



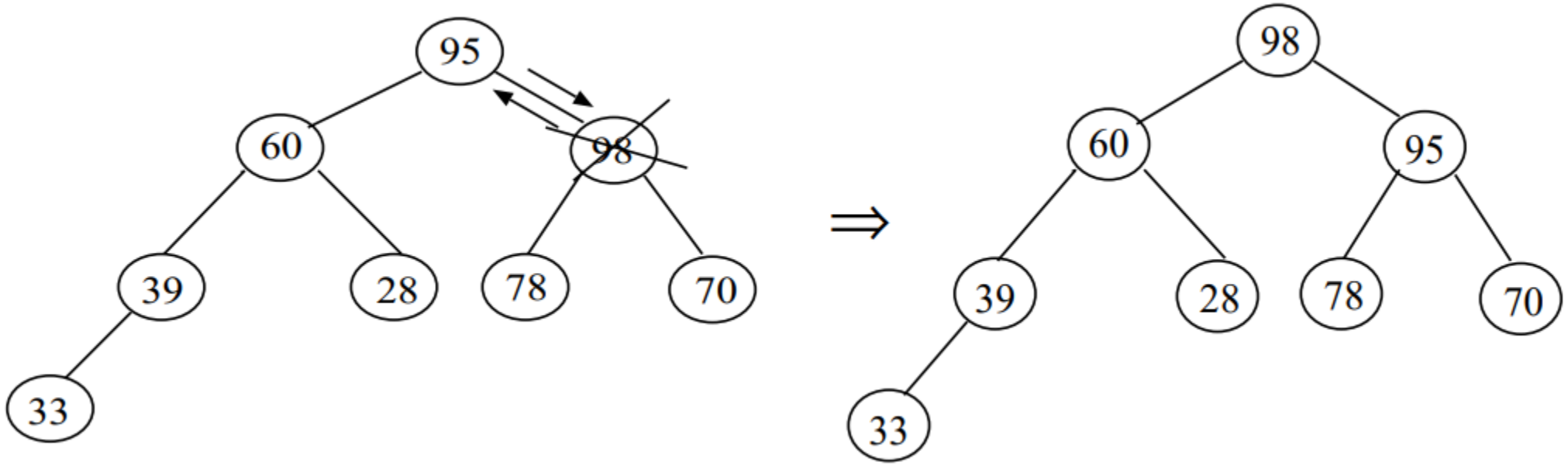
Alteração de Prioridades

- Exemplo: Alterar a prioridade do nó 6, de 66 para 98 (*cont.*)



Alteração de Prioridades

- Exemplo: Alterar a prioridade do nó 6, de 66 para 98 (*cont.*)



Alteração de Prioridades

- Seja *v* o nó cuja prioridade foi aumentada. Caso a prioridade do pai de *v*, se existir, seja menor do que a do *v*, trocar de posições *v* e o pai de *v*.

- Iterativamente, repetir esta operação, tornando *v* igual a seu pai, até que o nó considerado seja a raiz da árvore, ou que sua prioridade seja menor ou igual que a prioridade do seu pai.

Alteração de Prioridades

```
procedimento subir (i)
    j := ⌊i/2⌋
    se  j≥1 então
        se T[i].chave > T[j].chave então
            T[i] ⇔ T[j]
            subir (j)
```

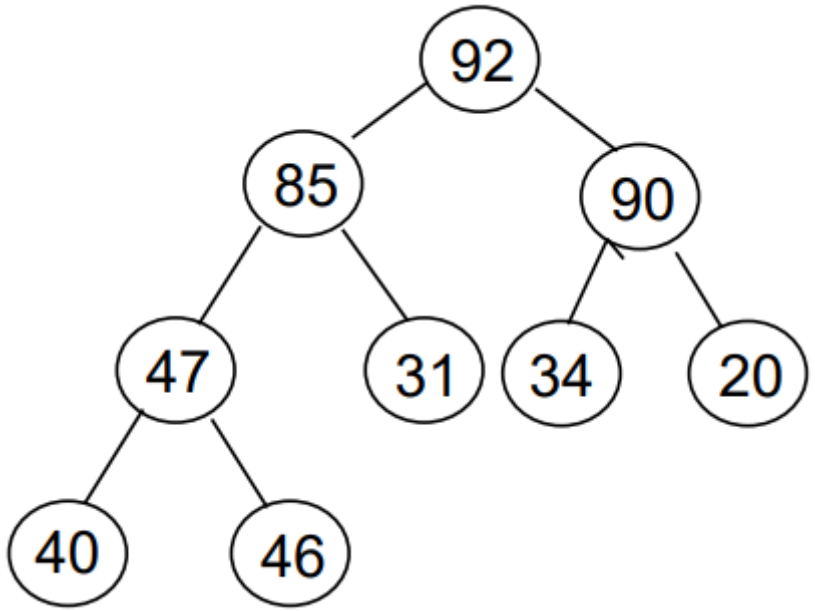
Alteração de Prioridades

- O heap está armazenado na tabela T. O parâmetro i indica a posição do elemento a ser revisto.
  - O campo chave armazena a prioridade do nó.
  - A notação T[i] ⇔ T[j] indica a troca de posições em T, entre os nós i e j.
  - Complexidade: Da ordem da altura da árvore. Como a árvore é completa, complexidade *O (log n)*.

```
procedimento subir (i)
    j := ⌊i/2⌋
    se  j≥1 então
        se T[i].chave > T[j].chave então
            T[i] ⇔ T[j]
            subir (j)
```

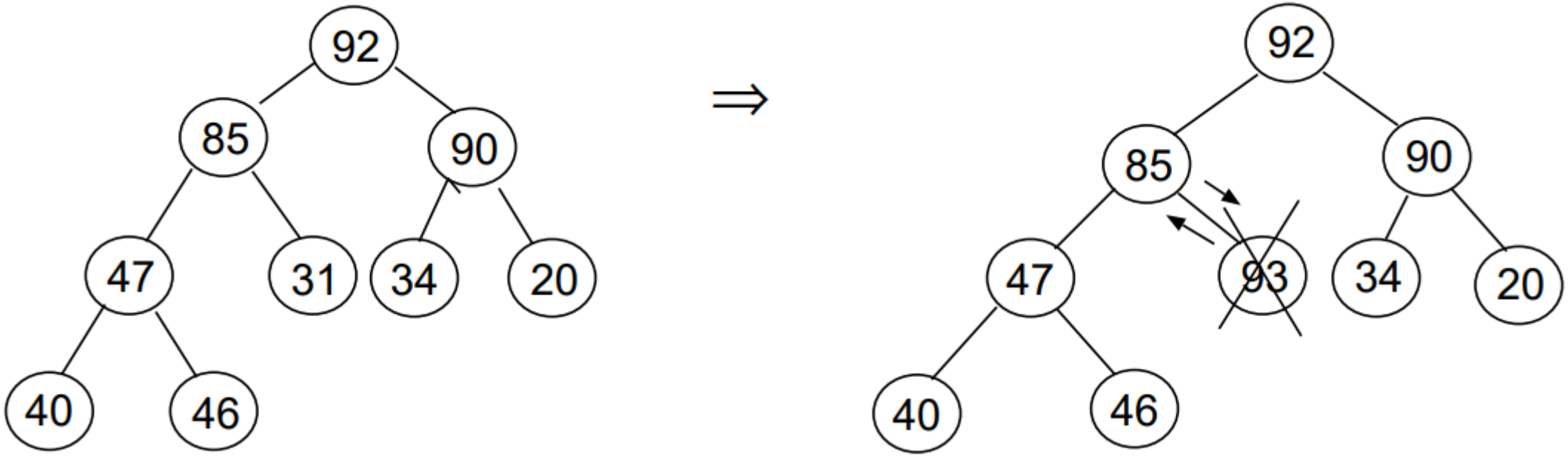
Alteração de Prioridades

- Seja o heap:

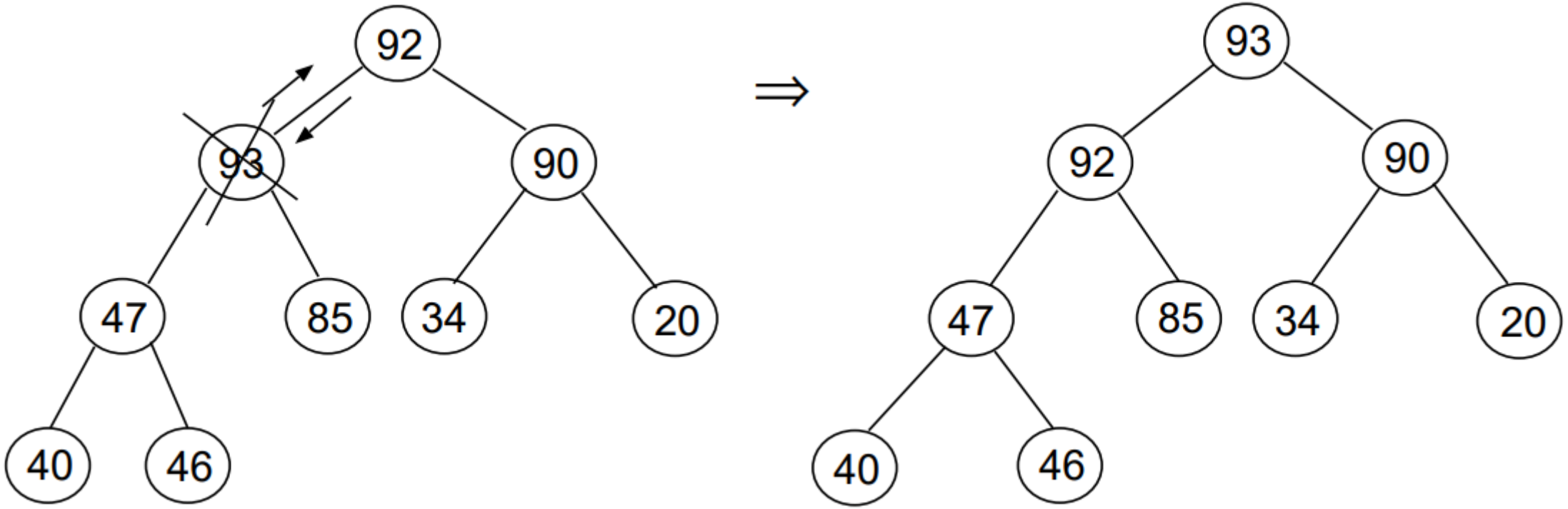


- Aplicando o algoritmo de subida, determinar o heap resultante da alteração de prioridade do quinto nó, de 31 por 93.

Alteração de Prioridades

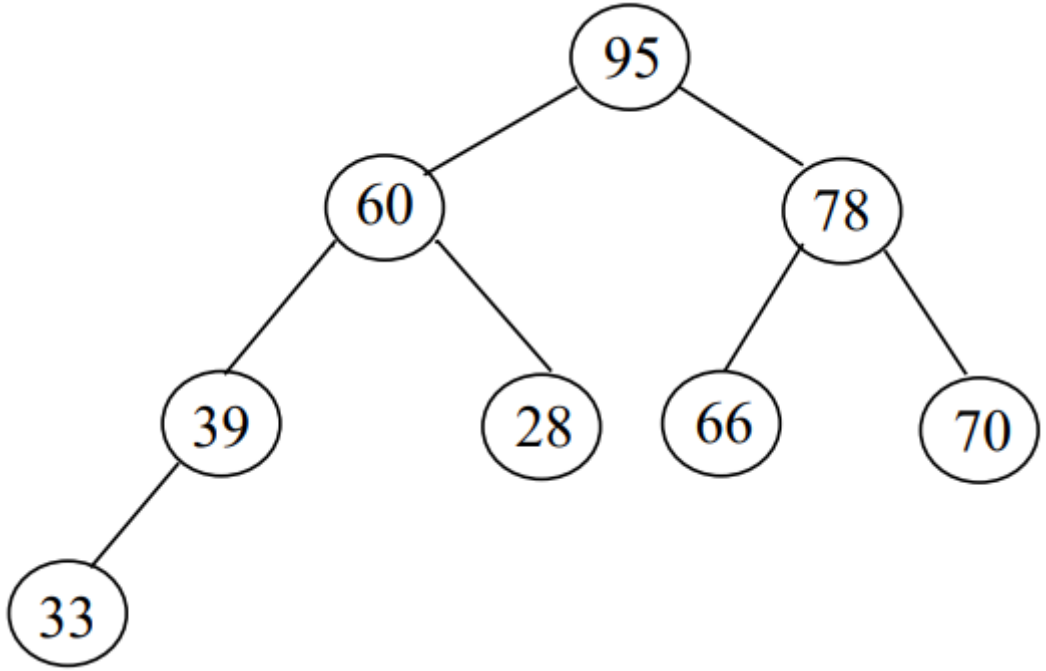


Alteração de Prioridades



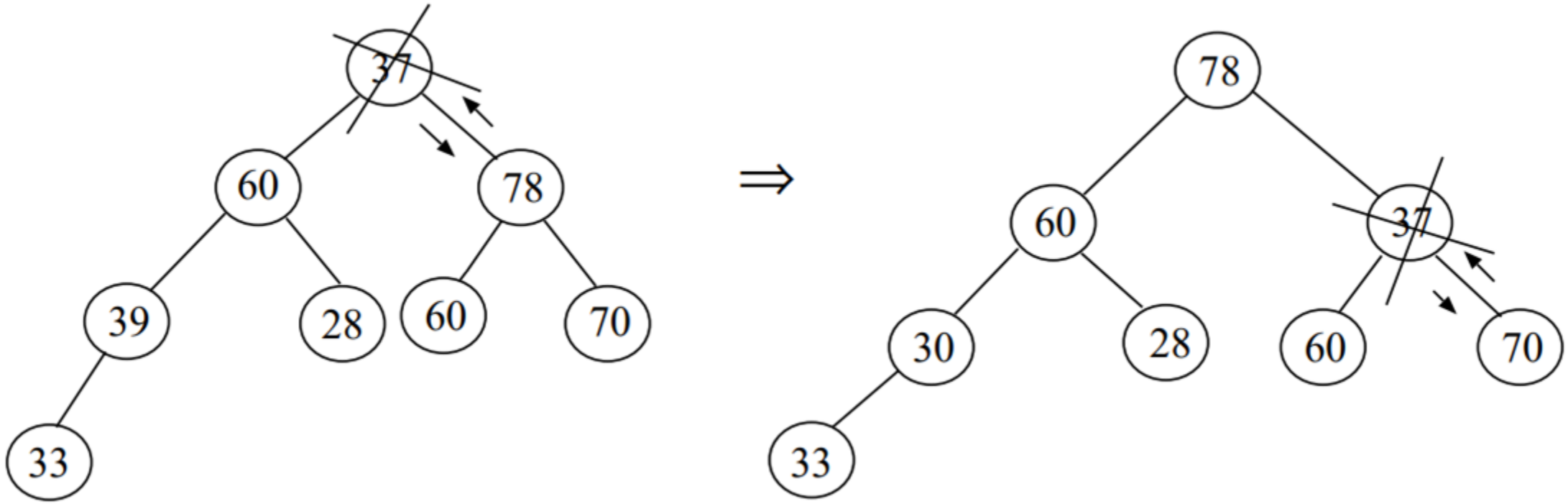
Alteração de Prioridades

- Exemplo: Diminuir a prioridade do nó 1, de 95 por 37.



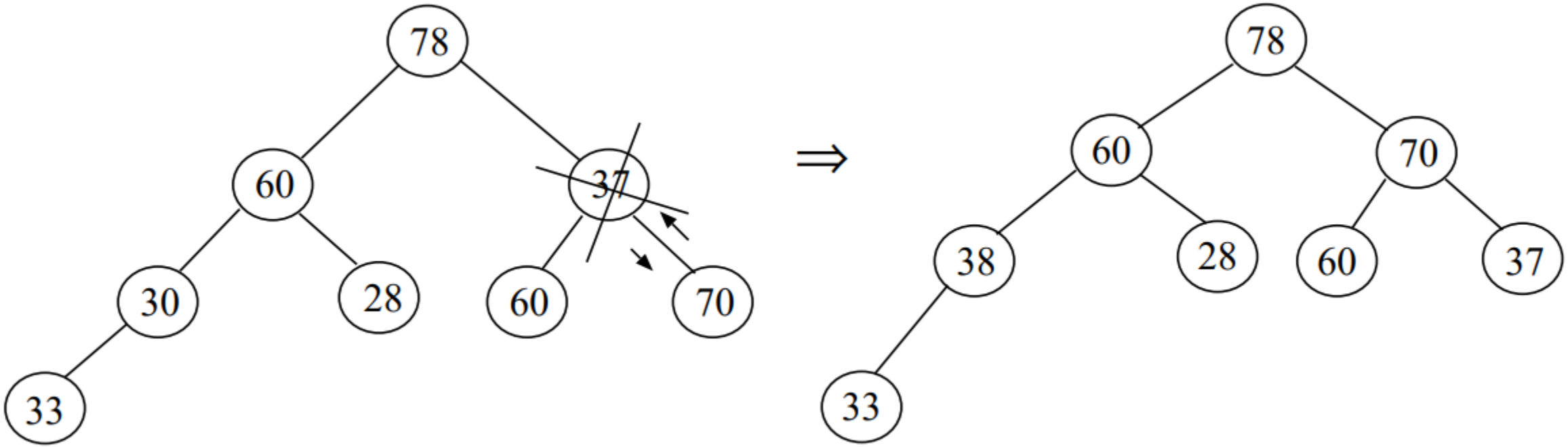
Alteração de Prioridades

- Exemplo: Diminuir a prioridade do nó 1, de 95 por 37 (*cont.*).



Alteração de Prioridades

- Exemplo: Diminuir a prioridade do nó 1, de 95 por 37 (*cont.*).



Alteração de Prioridades

- Seja *v* o nó cuja prioridade foi diminuída. Caso a prioridade de algum filho de *v*, se existir, seja maior do que a de *v*, trocar as posições de *v* e seu filho de maior prioridade.

- Iterativamente, repetir esta operação, tornando *v* igual a seu filho de maior prioridade, até que o nó considerado seja uma folha, ou que sua prioridade seja maior ou igual do que a de seus filhos.

Alteração de Prioridades

Procedimento: descer (i, n)

```
j : = 2 x i
se j ≤ n   então
    se j < n  então
        se T [j+1].chave > T [j].chave  então
            j : = j+1
    se T [i].chave < T [j].chave  então
        T [i] ⇔ T [j]
        descer (j, n)
```

Alteração de Prioridades

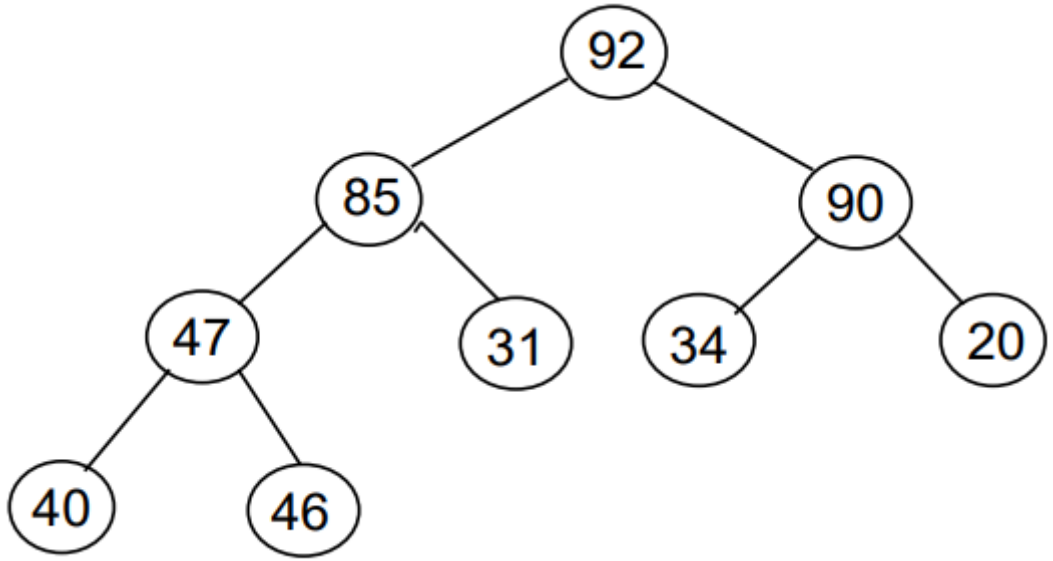
- O heap está armazenado na tabela *T*.
  - O parâmetro *i* indica a posição do elemento a ser revisto.
  - O parâmetro *n* indica a quantidade elementos na árvore heap.
  - O campo chave armazena a prioridade do nó.
  - A notação *T[i]* ⇔ *T[j]* indica a troca de posição em *T*, entre os nós *i* e *j*.

```
Procedimento: descer (i, n)

j : = 2 x i
se j ≤ n   então
    se j < n  então
        se T [j+1].chave > T [j].chave  então
            j : = j+1
    se T [i].chave < T [j].chave  então
        T [i] ⇔ T [j]
        descer (j, n)
```

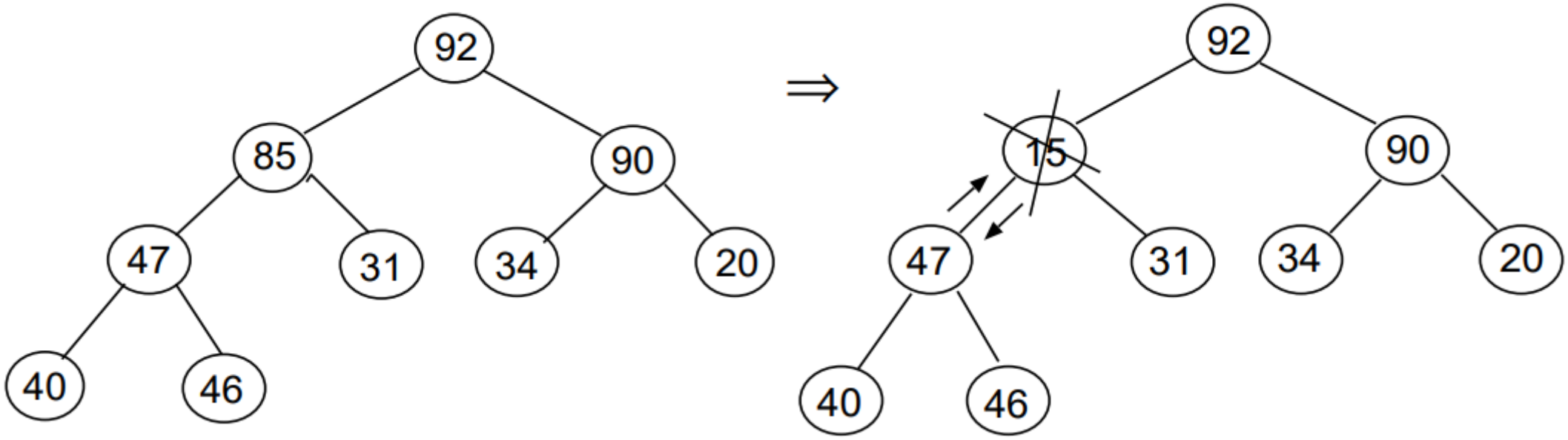
Alteração de Prioridades

- Seja o heap

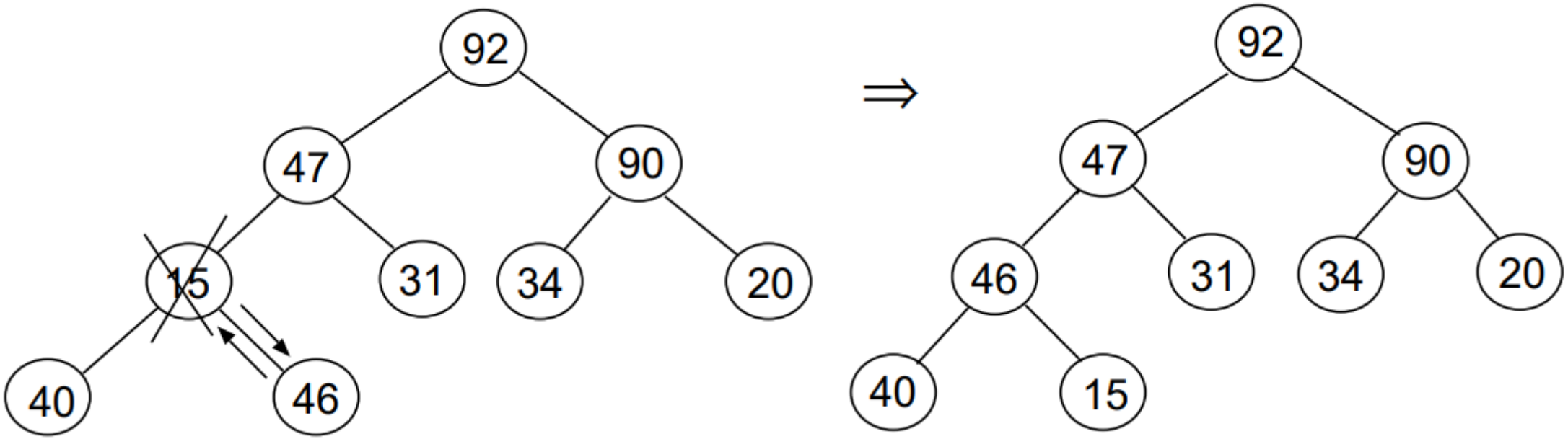


- Aplicando o algoritmo de descida, determine o heap resultante da alteração de prioridade do 2º nó, de 85 por 15.

Alteração de Prioridades



Alteração de Prioridades



Referências

- **Aula 28 – Lista de Prioridades**. Disponível em:

  <http://www2.ic.uff.br/~fabio/Aula-heaps-1.pdf>

- **Fila de Prioridade**. Disponível em:

  <https://www.ime.usp.br/~song/mac5710/slides/03prior.pdf>

Algoritmos e Estrutura de Dados II

Prof. Fellipe Guilherme Rey de Souza

Aula 24 – Listas de Prioridades