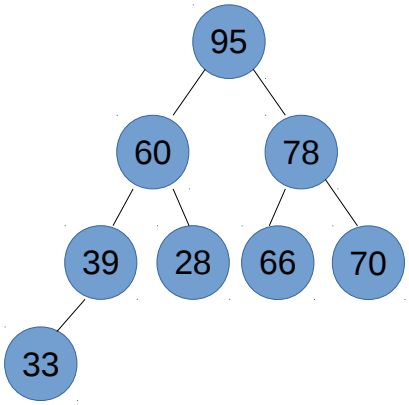
Lista de Prioridade
Heap - Inserção

- **Inserção do nó 73**

  → a lista passará a ter n+1 elementos

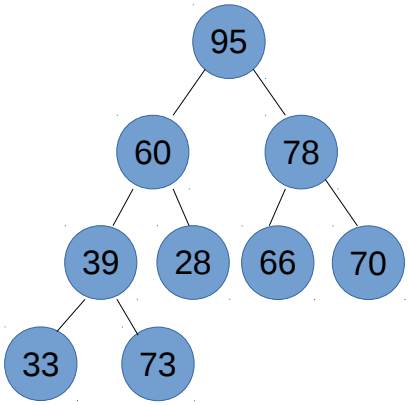
  → o novo elemento será colocado na posição n+ 1



UNIVERSIDADE FEDERAL DO CEARÁ

Lista de Prioridade
Heap - Inserção

- **Inserção do nó 73**
  - → a lista passará a ter n+1 elementos
  - → o novo elemento será colocado na posição n+ 1
  - → **A lista não respeita mais a propriedade (i)**

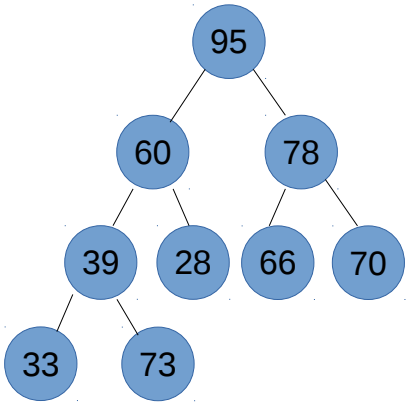


UNIVERSIDADE FEDERAL DO CEARÁ

Lista de Prioridade
Heap - Inserção

```
procedimento subir(i)
j := | i/2 |
se j >= 1 então
        se T[i].chave > T[j].chave então
                T[i] <=> T[j]
                subir(j)

 procedimento inserir (chave)
   se n < M então
       T[n+1] := chave
       n : n +1
       subir (n)
   senão overflow
```

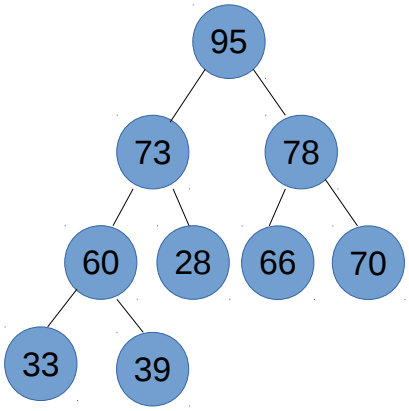


UNIVERSIDADE FEDERAL DO CEARÁ

Lista de Prioridade
Heap - Inserção

```
procedimento subir(i)
j := | i/2 |
se j >= 1 então
        se T[i].chave > T[j].chave então
                T[i] <=> T[j]
                subir(j)

 procedimento inserir (chave)
   se n < M então
       T[n+1] := chave
       n : n +1
       subir (n)
   senão overflow
```



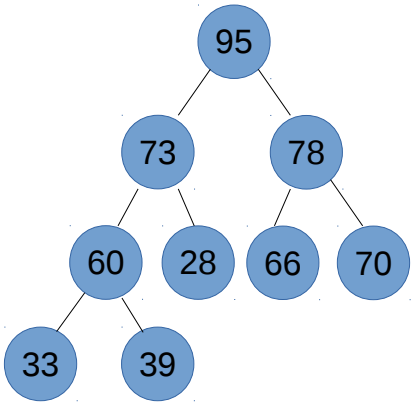
UNIVERSIDADE FEDERAL DO CEARÁ

Lista de Prioridade
Heap - Remoção

- **Remoção do nó 95**

  → a lista passará a ter n-1 elementos

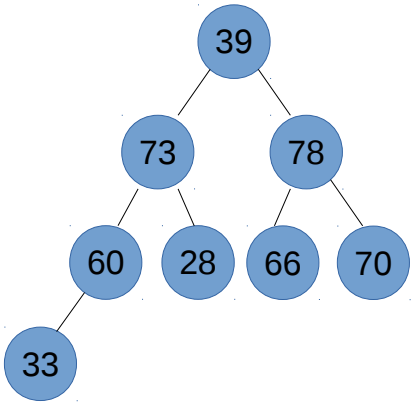
  → o último elemento substitui o primeiro



UNIVERSIDADE FEDERAL DO CEARÁ

Lista de Prioridade
Heap - Remoção

- **Remoção do nó 95**
  - → a lista passará a ter n-1 elementos
  - → o último elemento substitui o primeiro
  - → **A lista não respeita mais a propriedade (i)**

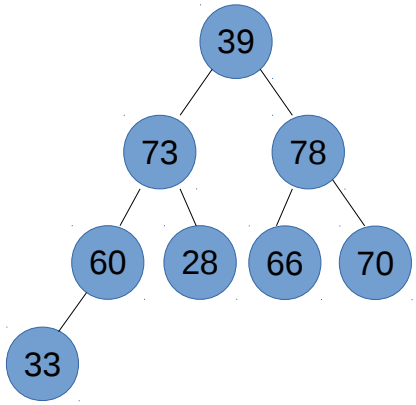


UNIVERSIDADE FEDERAL DO CEARÁ

Lista de Prioridade

Heap - Remoção

```
procedimento descer(i,n)
j := 2 * 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)
procedimento remoção ( )
se n != 0 então
    agir (T[1])
    T[1] := T[n]
    n := n -1
    descer(1, n)
senão underflow
```

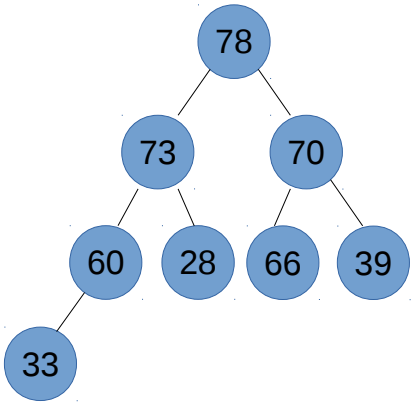


UNIVERSIDADE FEDERAL DO CEARÁ

Lista de Prioridade

Heap - Remoção

```
procedimento descer(i,n)
j := 2 * 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)
procedimento remoção ( )
se n != 0 então
    agir (T[1])
    T[1] := T[n]
    n := n -1
    descer(1, n)
senão underflow
```



UNIVERSIDADE FEDERAL DO CEARÁ

Lista de Prioridade

Heap - Construção

```
procedimento construção (S)
    para i := 2, n faça
        subir(i)
```

```
procedimento subir(i)
j := | i/2 |
se j >= 1 então
        se T[i].chave > T[j].chave então
            T[i] <=> T[j]
            subir(j)
```

UNIVERSIDADE FEDERAL DO CEARÁ