MC202 - Estrutura de Dados

Alexandre Xavier Falcão

Instituto de Computação - UNICAMP

afalcao@ic.unicamp.br

Heap Binário

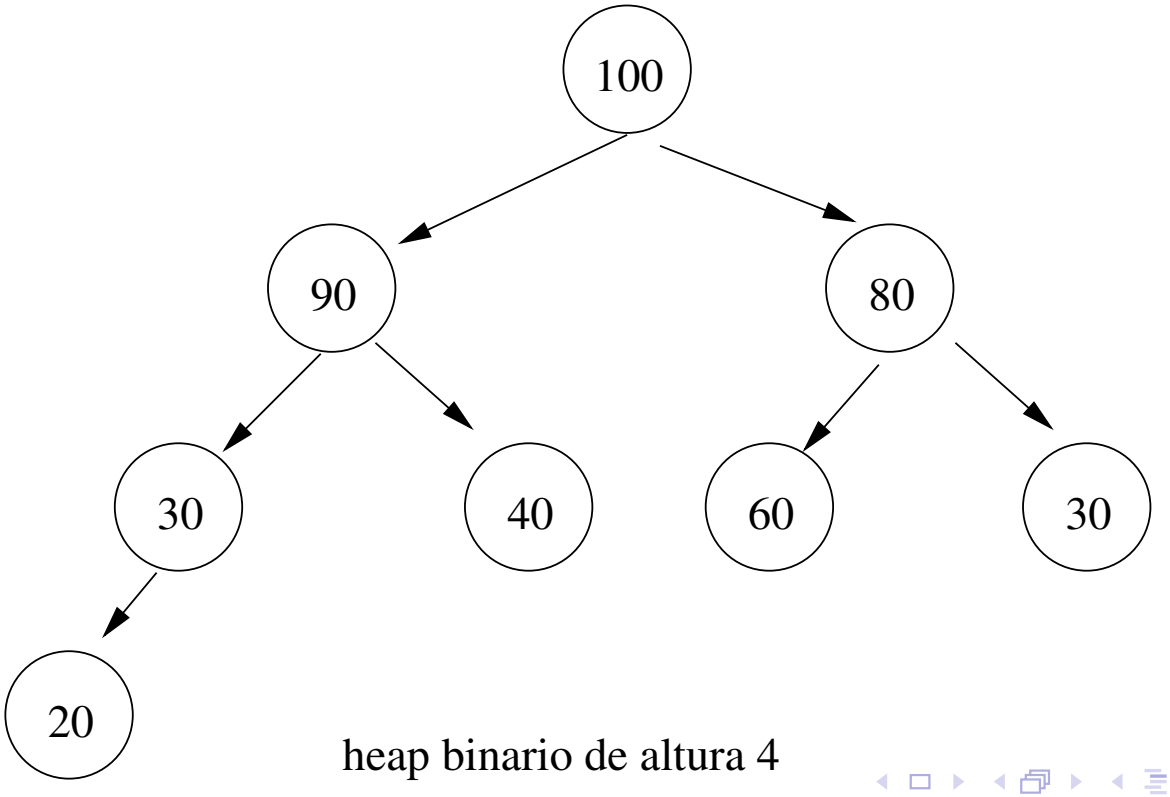
- Um heap é uma fila de prioridades onde o elemento removido é sempre o de maior (menor) prioridade.

Heap Binário

- Um heap é uma fila de prioridades onde o elemento removido é sempre o de maior (menor) prioridade.
- Um heap é dito binário quando pode ser visto como uma árvore binária cheia (ou quase cheia), onde o valor de cada nó é maior (menor) ou igual ao valor de seus filhos à esquerda e à direita.

Heap Binário

- Um heap é uma fila de prioridades onde o elemento removido é sempre o de maior (menor) prioridade.
- Um heap é dito binário quando pode ser visto como uma árvore binária cheia (ou quase cheia), onde o valor de cada nó é maior (menor) ou igual ao valor de seus filhos à esquerda e à direita.



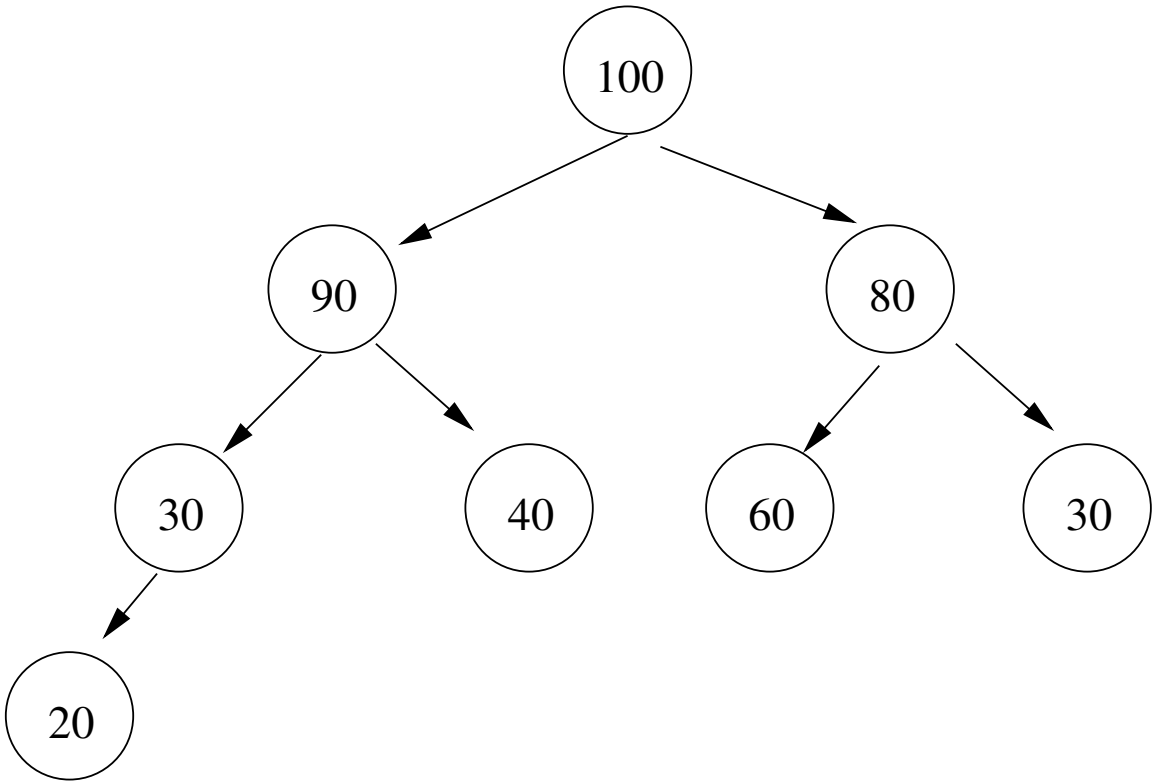
heap binario de altura 4

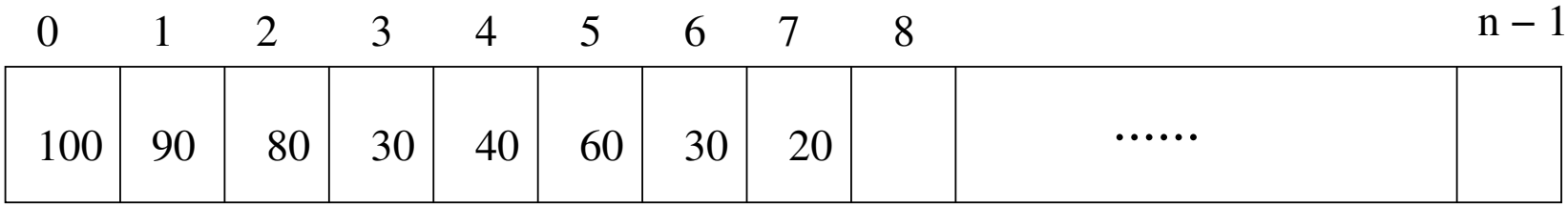
Agenda

- Aspectos de implementação de um heap binário.
- Inserção de um elemento no heap.
- Remoção de um elemento no heap.
- Aplicação em ordenação.

Heap Binário

Um heap binário de altura h pode ser implementado com um vetor de $n = 2^h - 1$ posições.





Heap Binário

- Os índices no vetor dos filhos à esquerda e à direira de um nó $0 \leq i \leq \left\lfloor \frac{n-3}{2} \right\rfloor$ são obtidos por $2i + 1$ e $2i + 2$, respectivamente.

Heap Binário

- Os índices no vetor dos filhos à esquerda e à direira de um nó $0 \leq i \leq \left\lfloor \frac{n-3}{2} \right\rfloor$ são obtidos por $2i+1$ e $2i+2$, respectivamente.
- Portanto, os índices *pai*, *filho_esq*, e *filho_dir* de um nó $1 \leq i \leq \left\lfloor \frac{n-3}{2} \right\rfloor$ são relacionados por

$$\begin{aligned} pai &= \left\lfloor \frac{i-1}{2} \right\rfloor \\ filho\_esq &= 2i+1 \\ filho\_dir &= 2i+2 \end{aligned}$$

Heap Binário

- Os índices no vetor dos filhos à esquerda e à direira de um nó $0 \leq i \leq \left\lfloor \frac{n-3}{2} \right\rfloor$ são obtidos por $2i+1$ e $2i+2$, respectivamente.
- Portanto, os índices *pai*, *filho_esq*, e *filho_dir* de um nó $1 \leq i \leq \left\lfloor \frac{n-3}{2} \right\rfloor$ são relacionados por

$$\begin{aligned} pai &= \left\lfloor \frac{i-1}{2} \right\rfloor \\ filho\_esq &= 2i+1 \\ filho\_dir &= 2i+2 \end{aligned}$$

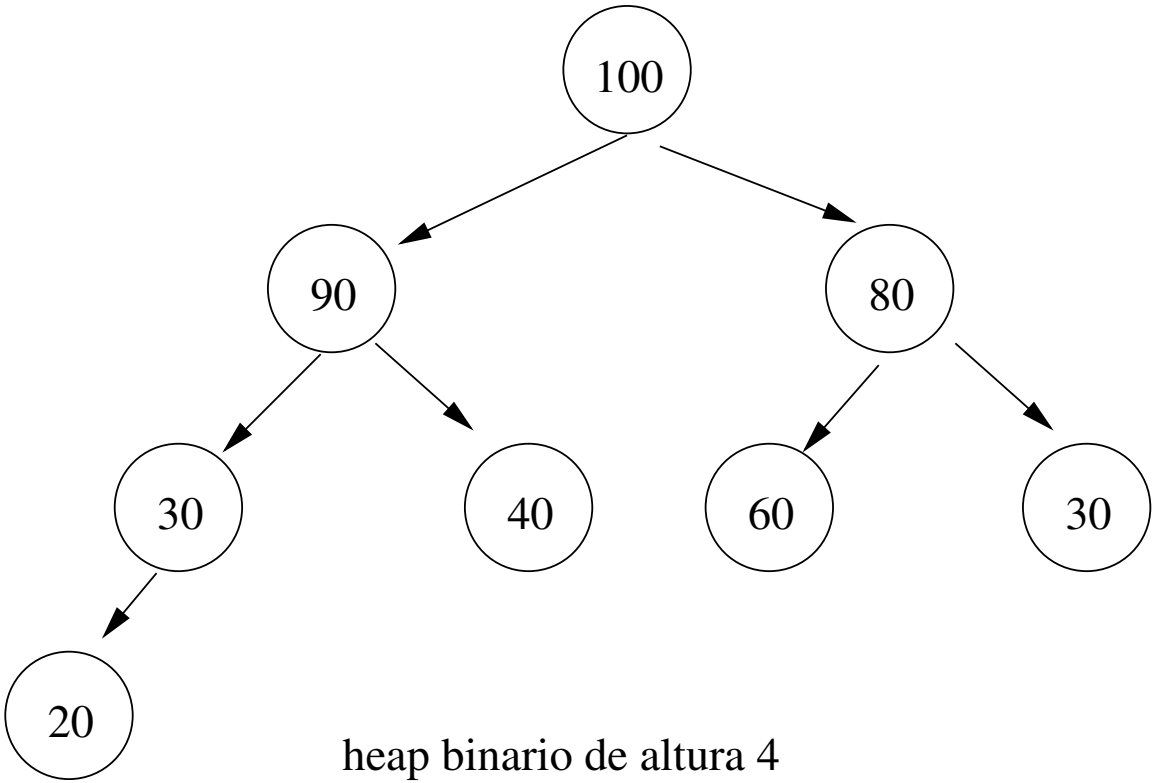
O maior (menor) valor do heap está sempre no índice 0.

Inserção no heap binário

Um novo nó é sempre inserido no final do heap e depois seu valor é comparado com o valor do pai, trocando eles sempre que for maior que o valor do pai, subindo no heap até que a propriedade de heap seja satisfeita novamente.

Inserção no heap binário

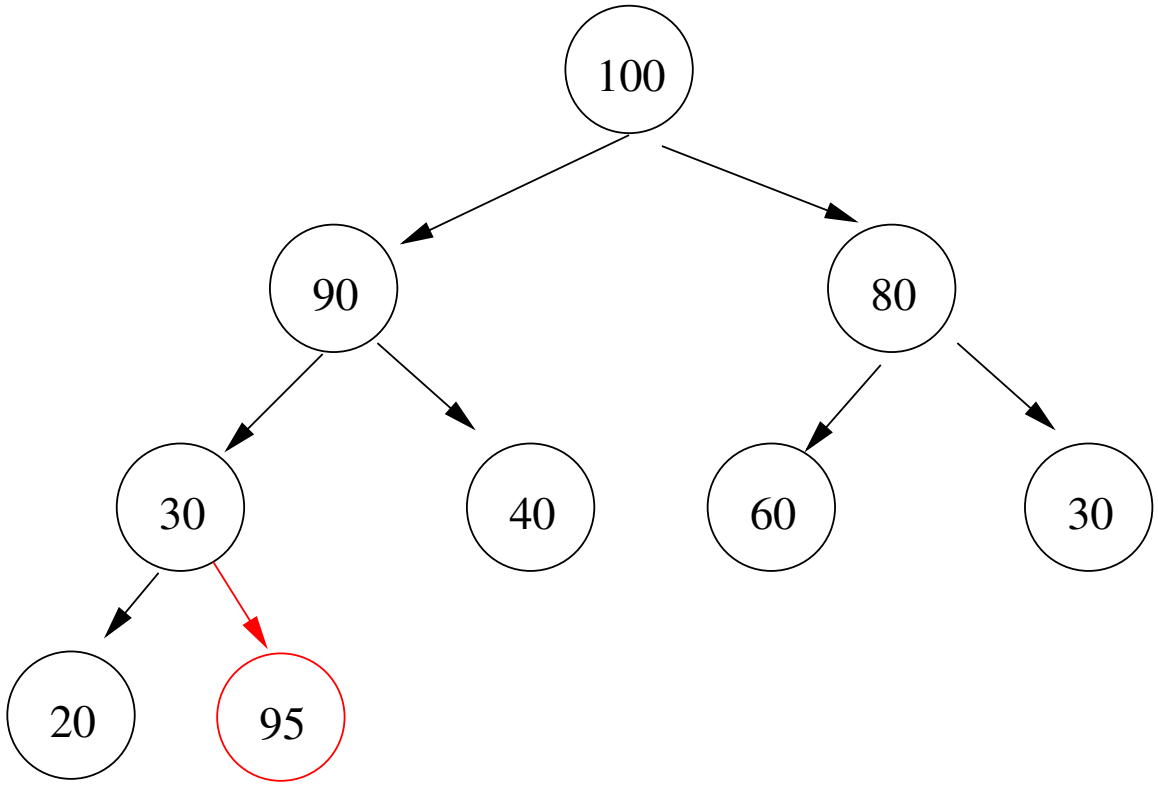
Por exemplo, considere a inserção do nó de valor 95.



heap binario de altura 4

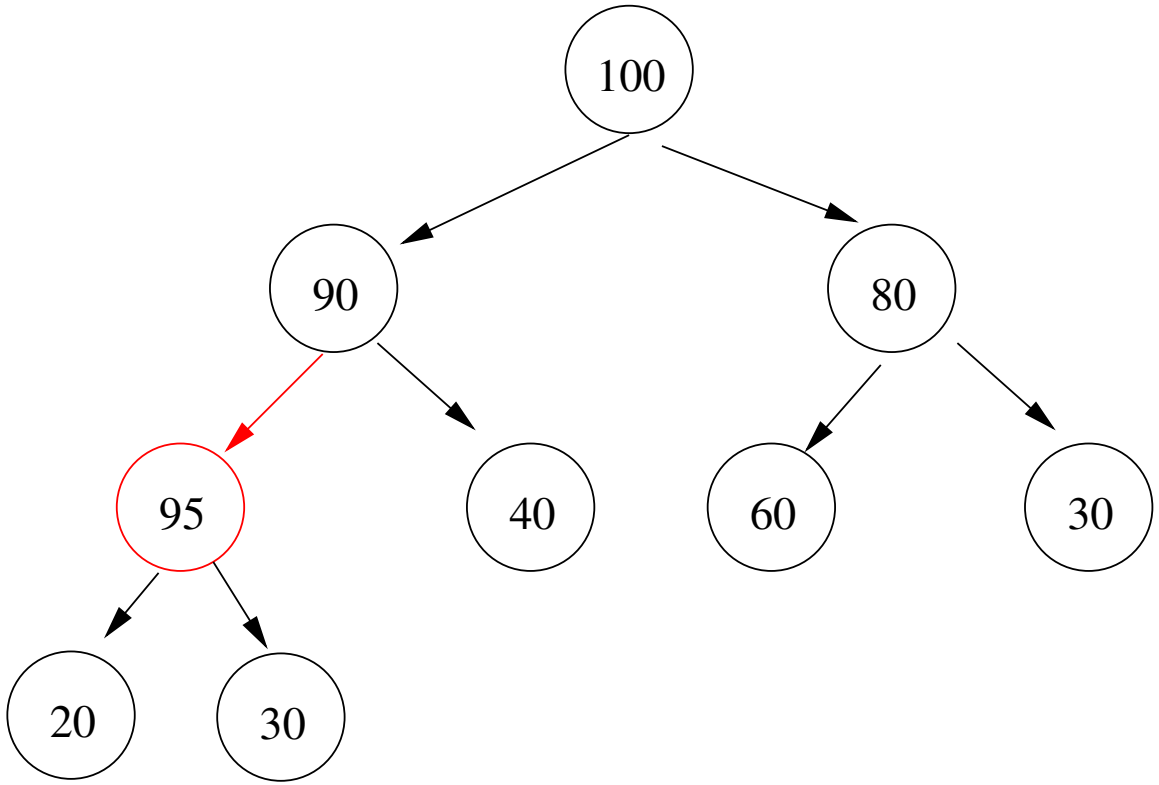
Inserção no heap binário

Por exemplo, considere a inserção do nó de valor 95.



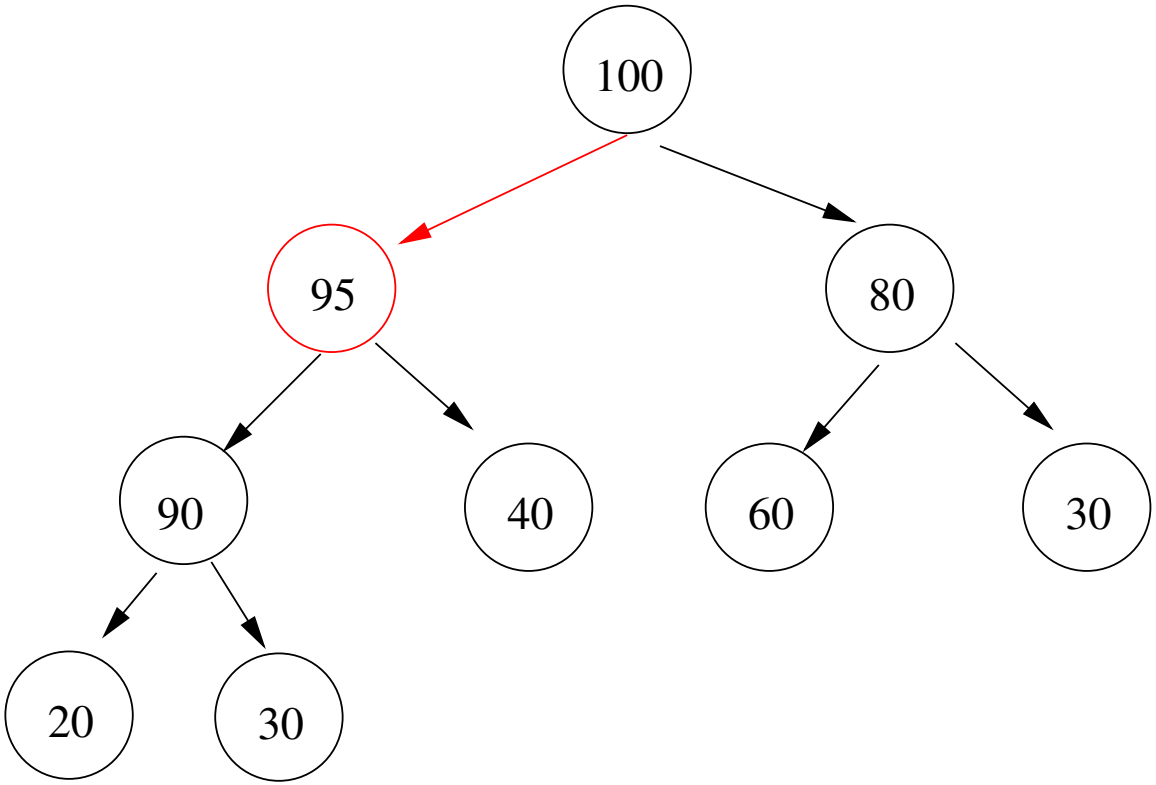
Inserção no heap binário

Por exemplo, considere a inserção do nó de valor 95.



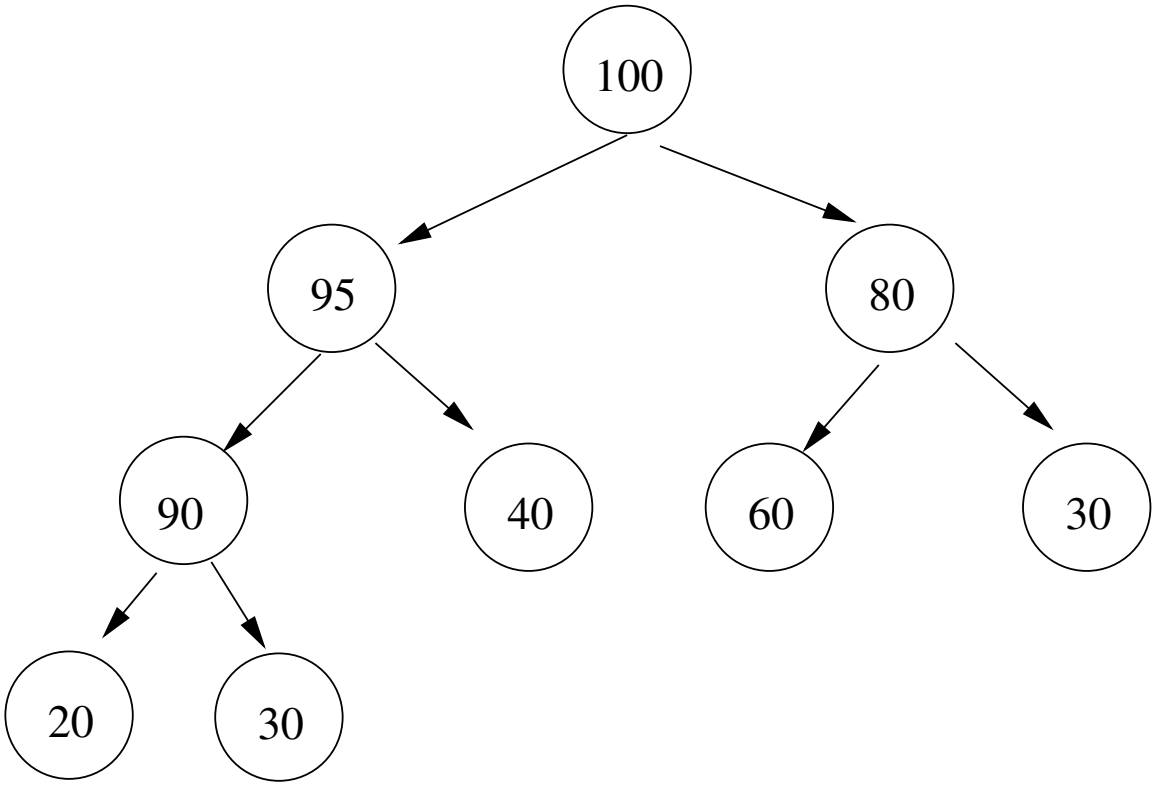
Inserção no heap binário

Por exemplo, considere a inserção do nó de valor 95.



Inserção no heap binário

Por exemplo, considere a inserção do nó de valor 95.



Remoção no heap binário

A remoção do nó com maior valor é feita trocando-o pelo último nó armazenado no heap,

subtraindo de um o número de nós armazenados, e

depois realizando o seguinte procedimento de descida.

Remoção no heap binário

A remoção do nó com maior valor é feita trocando-o pelo último nó armazenado no heap,

subtraindo de um o número de nós armazenados, e

depois realizando o seguinte procedimento de descida.

- Compara-se o valor da raiz com os de seus filhos à esquerda e à direira,

Remoção no heap binário

A remoção do nó com maior valor é feita trocando-o pelo último nó armazenado no heap,

subtraindo de um o número de nós armazenados, e

depois realizando o seguinte procedimento de descida.

- Compara-se o valor da raiz com os de seus filhos à esquerda e à direira,
- troca a raiz pelo filho de maior valor e

Remoção no heap binário

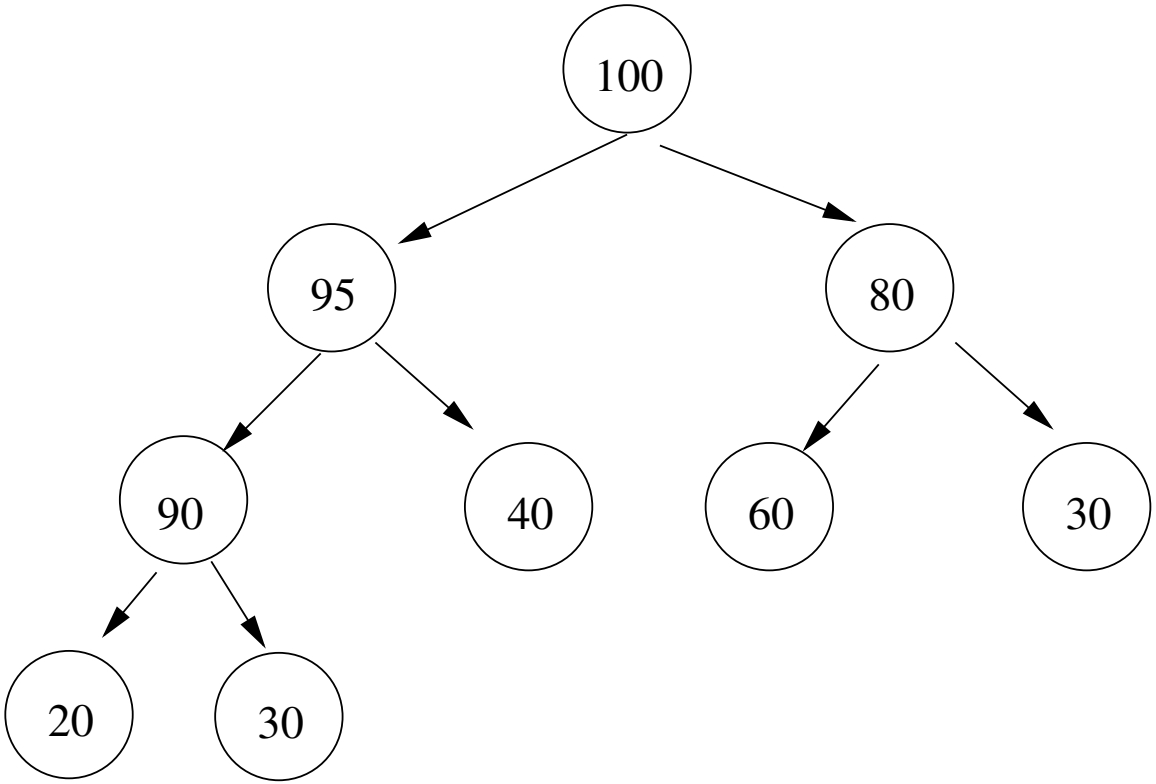
A remoção do nó com maior valor é feita trocando-o pelo último nó armazenado no heap,

subtraindo de um o número de nós armazenados, e

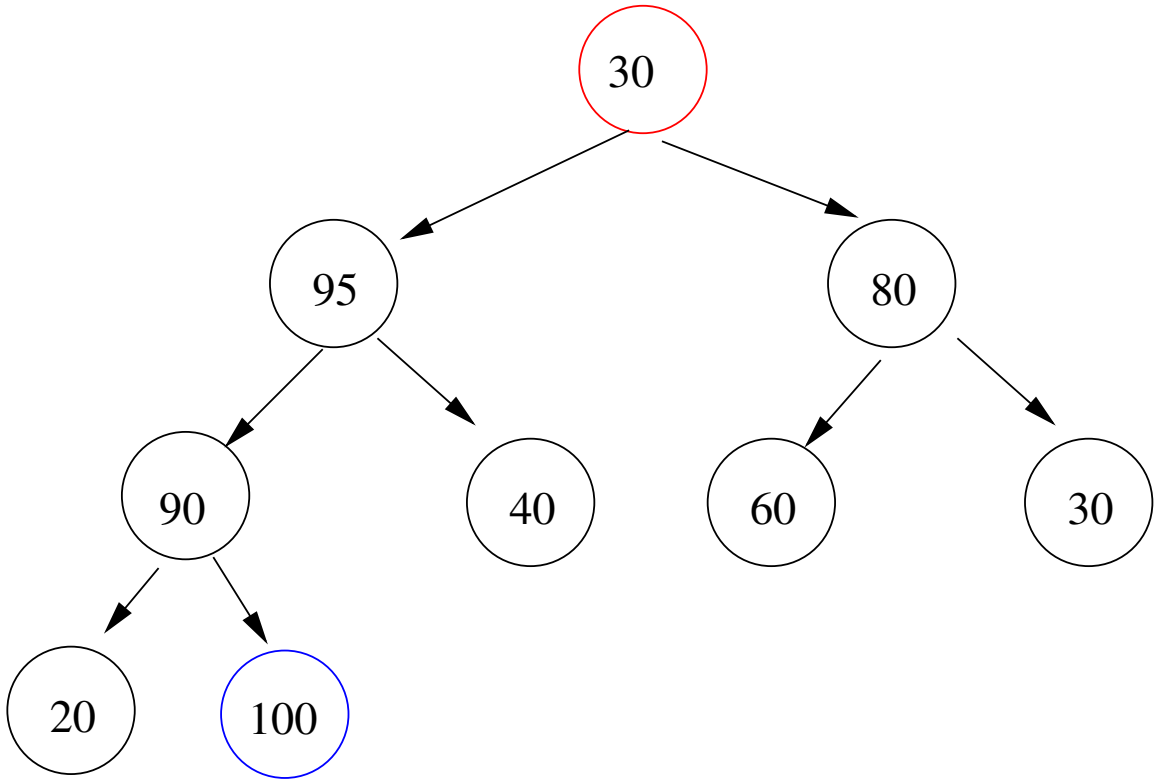
depois realizando o seguinte procedimento de descida.

- Compara-se o valor da raiz com os de seus filhos à esquerda e à direira,
- troca a raiz pelo filho de maior valor e
- repete este processo em sua subárvore até que a propriedade de heap seja novamente satisfeita.

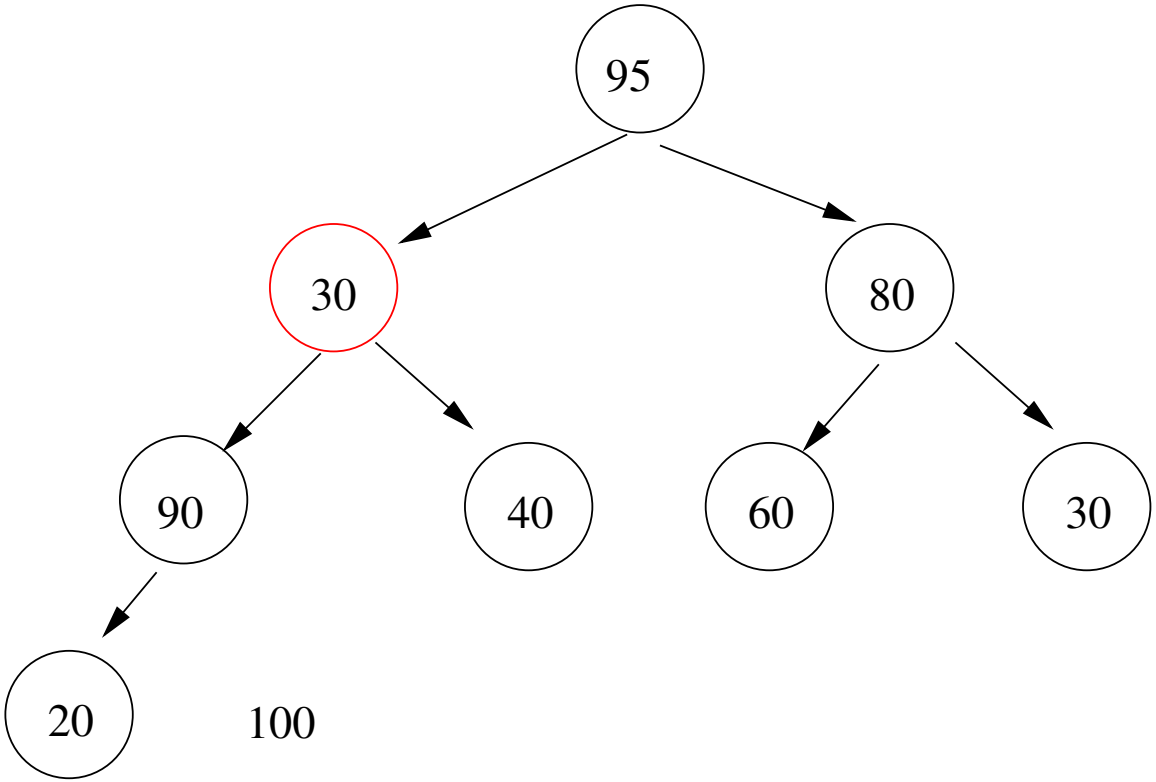
Remoção no heap binário



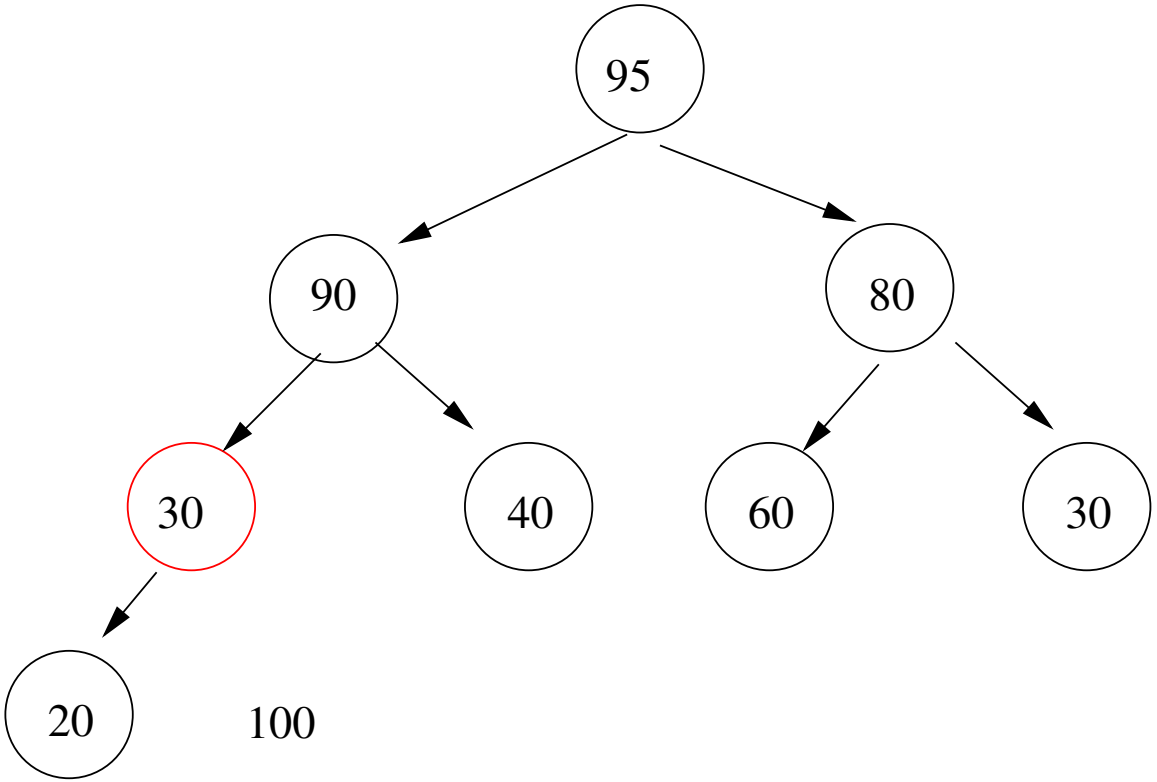
Remoção no heap binário



Remoção no heap binário



Remoção no heap binário



Heapsort

- Um vetor com n valores armazenados pode ser transformado em um heap binário ao assumirmos inserções (procedimento de subida) subsequentes do nó da posição $i = 1$ até o nó da posição $i = n - 1$. Esta operação terá custo $O(n \log_2^n)$.

Heapsort

- Um vetor com n valores armazenados pode ser transformado em um heap binário ao assumirmos inserções (procedimento de subida) subsequentes do nó da posição $i = 1$ até o nó da posição $i = n - 1$. Esta operação terá custo $O(n \log_2^n)$.

- Podemos reduzir este custo para $O(n)$, se aplicarmos o procedimento de descida para todo nó de $i = pai(n - 1)$ até $i = 0$, pois isso já seleciona o maior entre pai e filhos, realizando mais resultados a cada iteração.

Heapsort

- Um vetor com n valores armazenados pode ser transformado em um heap binário ao assumirmos inserções (procedimento de subida) subsequentes do nó da posição $i = 1$ até o nó da posição $i = n - 1$. Esta operação terá custo $O(n \log_2^n)$.

- Podemos reduzir este custo para $O(n)$, se aplicarmos o procedimento de descida para todo nó de $i = pai(n - 1)$ até $i = 0$, pois isso já seleciona o maior entre pai e filhos, realizando mais resultados a cada iteração.

- As n remoções subsequentes deste heap irão colocar seus nós no próprio vetor em ordem crescente de valores.