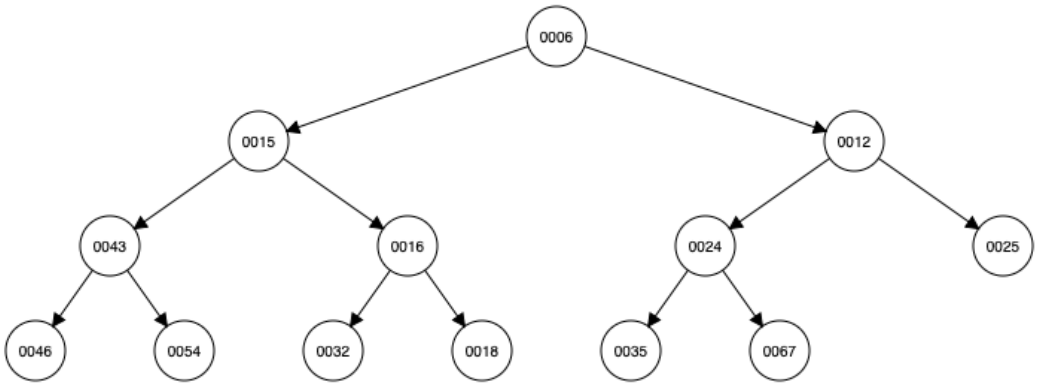
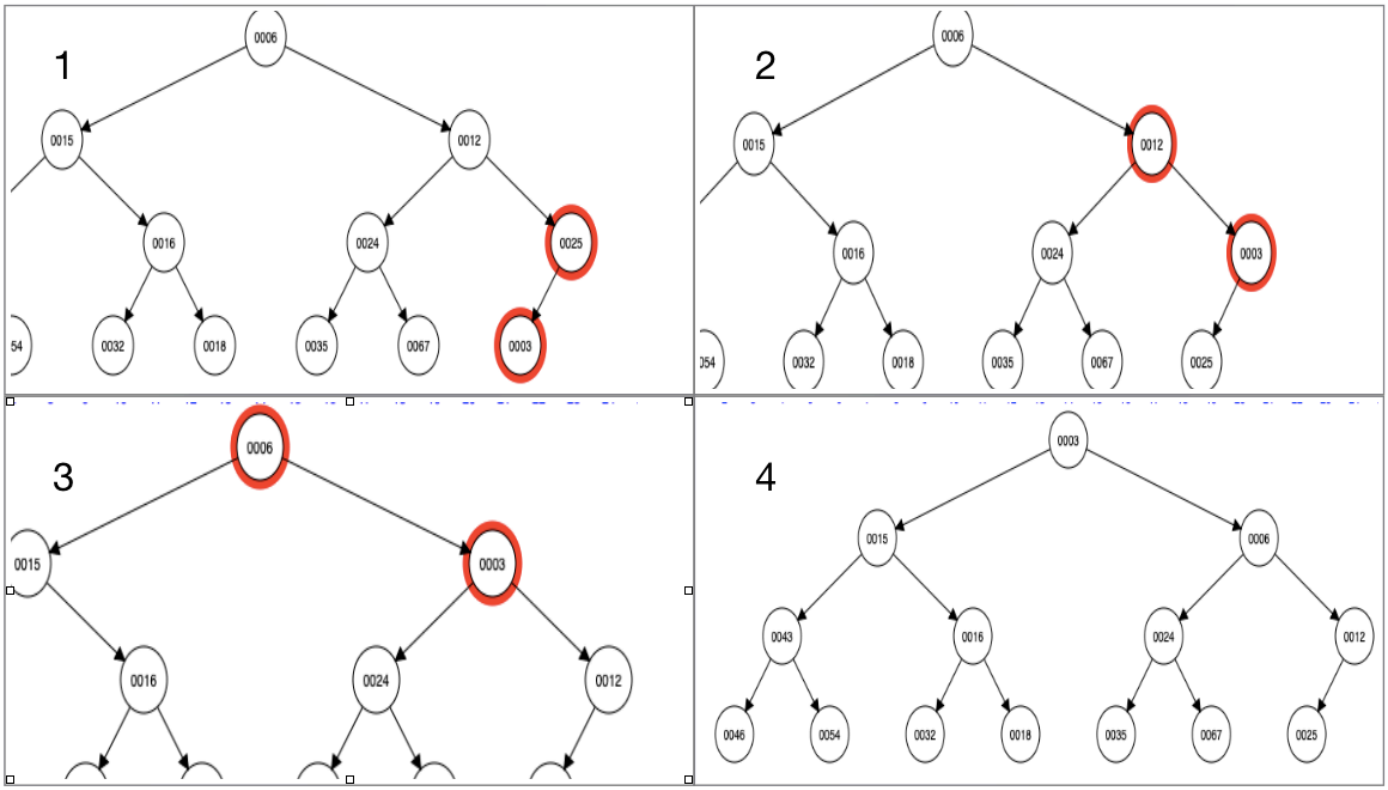
# Soluzione Heap Binari

## Heap di tipo Min: insert




> Si consideri lo Heap Binario di tipo min disegnato sopra, che indicheremo con **A**.
>
> **[1/3 del punteggio]** Assumendo che le chiavi siano numeri interi, si disegni come viene modificato lo heap **A** dopo la seguente chiamata (senza fornire alcuna spiegazione dei passaggi: disegnate solo il risultato): `insert(3, “elem”, A)`;
>
> La chiamata insert(3, “elem”, A) rappresenta il caso peggiore della operazione insert sullo heap **A**, rispetto alla complessità temporale? Motivare la risposta.

Il nuovo nodo viene **aggiunto come foglia nel primo posto disponibile** ossia come figlio sinistro di 25. Da li parte la procedura che “sistema” i nodi, quindi il nuovo nodo viene fatto “risalire” fino alla radice. ( 25 → 12 → 6 ). Quindi avremo:



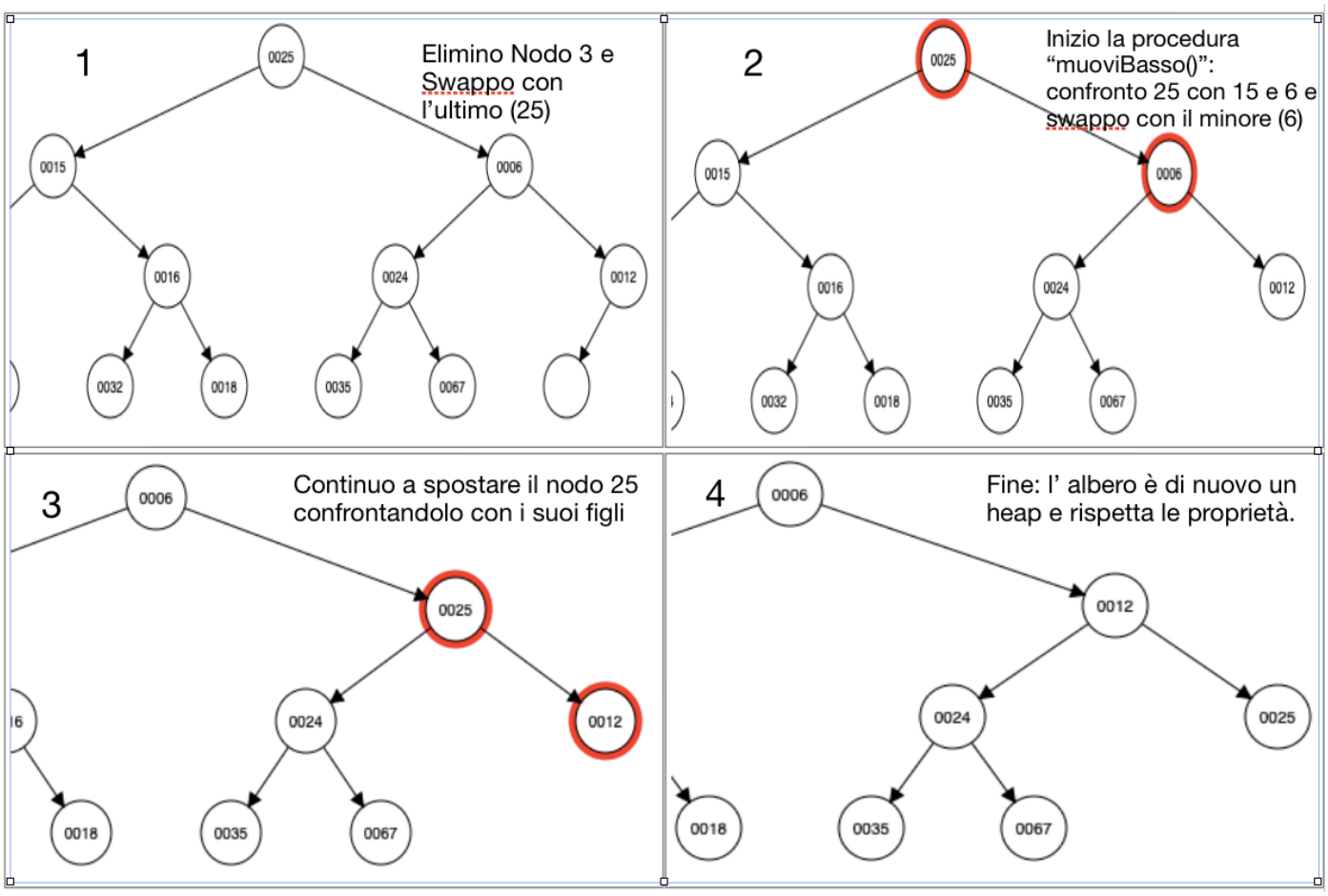

Dato che dobbiamo risalire l’heap, ricadiamo nel caso peggiore per l’inserimento che vale $\Theta(logn)$.

## Heap di tipo Min: delete

> **[2/3 del punteggio]** Si spieghino dettagliatamente, mediante disegni chiari e autoesplicativi, i passaggi principali della chiamata `deleteMin(A);` effettuata sullo heap **A** modificato a seguito dell'inserimento dell'elemento con chiave 3.

Si sostituisce l'**ultimo nodo** dell'albero con la **radice**. Questa operazione compromette le proprietà strutturali dell' heap in questione, infatti verranno eseguite diverse chiamate di funzione `muoviBasso()` che "sistema" i nodi dell'heap in modo tale da rispettare le proprietà.

Nel nostro caso, ricadiamo ancora una volta nel caso peggiore, ossia $O(log\ n)$ in quanto dovremo scorrere tutto un lato dell' heap per tornare ad una situazione di stabilità.




!! A pagina 21-22 degli appunti ci sono altri esempi