

# LAB 6

- **Error dict::insertElem(const Key k, const Value v, Dictionary& d):**

Se il nodo corrente e' vuoto, alloco nuovo nodo e gli assegno i valori k e v passati come argomento; il figlio destro e sinistro sono vuoti; **altrimenti** procedo discendendo nell'albero mediante chiamate ricorsive e nel momento in cui troviamo un nodo vuoto durante la ricerca, lo sostituiamo con il puntatore al nodo appena creato. A questo punto il nuovo nodo sara' nel posto giusto, ovvero nel punto in cui la funzione search lo trovera'.

Se il nodo corrente non e' vuoto, cerchiamo la chiave k con un algoritmo ricorsivo analogo a quello della search; nel caso in cui la chiave ci sia, non dobbiamo re-inserirla e restituiamo FAIL; **Se** l'elemento da inserire ha chiave minore della chiave del nodo corrente, lo inserisco a sinistra con una chiamata ricorsiva a insertElem, **altrimenti** inserisco a destra.

- **Value dict::search(const Key k, const Dictionary& d):**

Abbastanza simile al lab 5 a livello logico ma l'unica differenza che qua siamo in alberi binari di ricerca, quindi dobbiamo scendere a sinistra o a destra confrontando la chiave dell'elemento, usando la ricorsione ci spostiamo o a sinistra o a destra.

- **Delete min:**

Questa è una funzione ausiliare della deleteElem, il suo compito è quello di rimuovere l'elemento con chiave minore in un albero non vuoto e restituisce il dictionary Elem (la coppia (key,value) corrispondente all'elemento rimosso).

Allora partiamo con la if, **SE** il figlio sinistro di d è vuoto vuol dire che d contiene l'elemento del dizionario con chiave minima, lo deve eliminare.

Quello che vado a fare è inanzitutto dichiararmi una variabile temporanea (tmpElem) che contiene la coppia chiave valore del nodo corrente.

Dichiaro un puntatore tmpNode che punta al figlio destro.

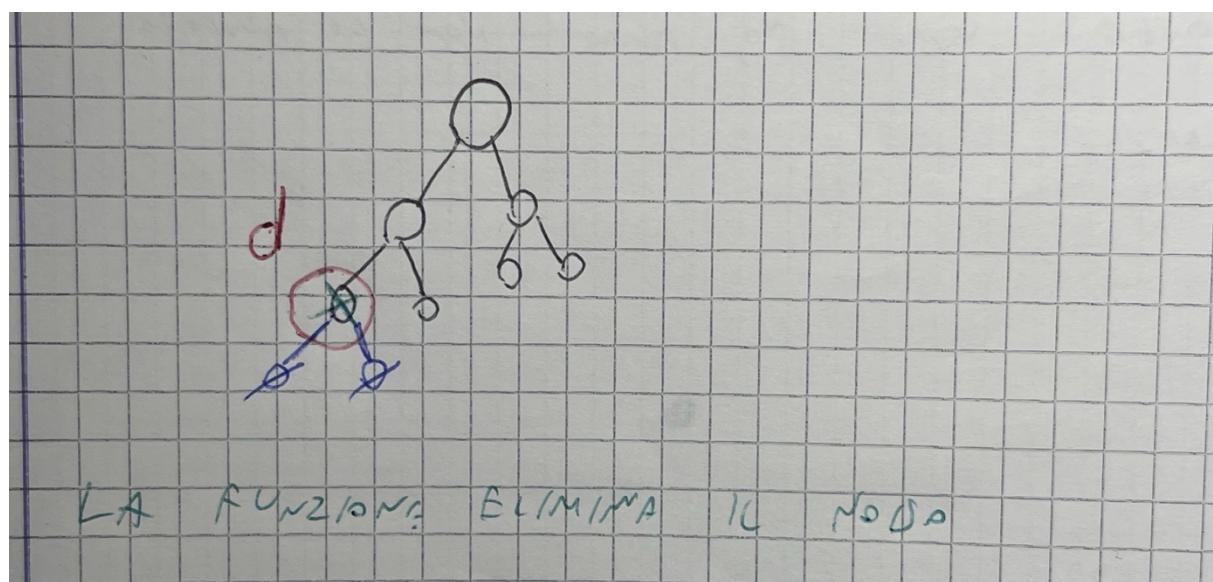
Ora posso eliminare il nodo corrente, rimpiazzo d con il suo figlio destro e restituisco la variabile tmpElem.

**ELSE** effettuo una sorta di ricerca, praticamente ricorsivamente scorro tutto il nodo sinistro.

- **deleteElem:**

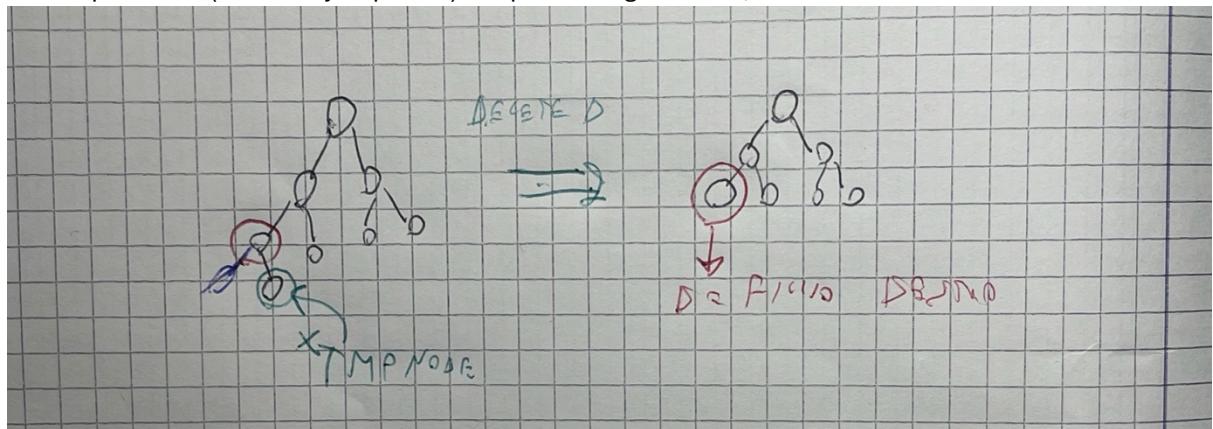
Abbiamo 4 casi da considerare:

1) entrambi i figli di d sono vuoti: devo soltanto deallocare (delete d) e porre d = emptyNode.



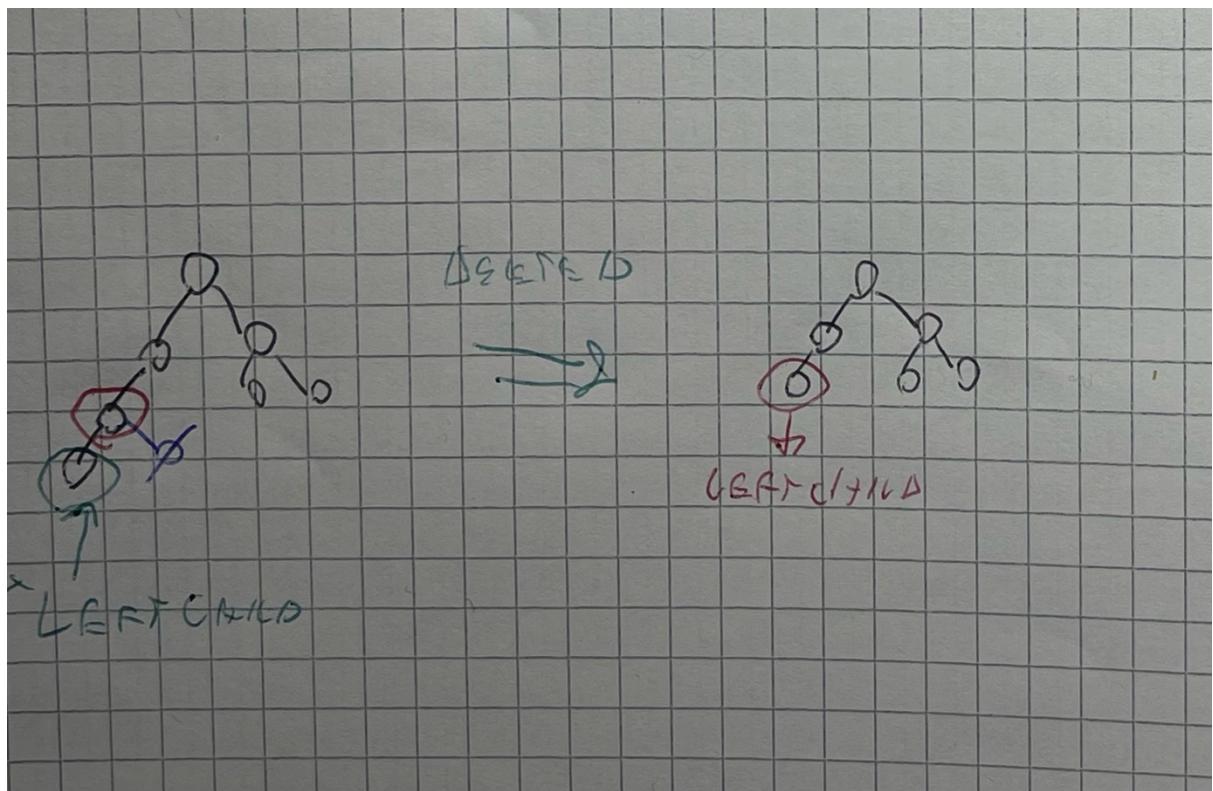
2) Il figlio sinistro di d è vuoto ma il figlio destro non lo è: rimpiazzo d con il figlio destro.

Uso un puntatore (Dictionary tmpNode) e lo punto al figlio destro, elimino d .

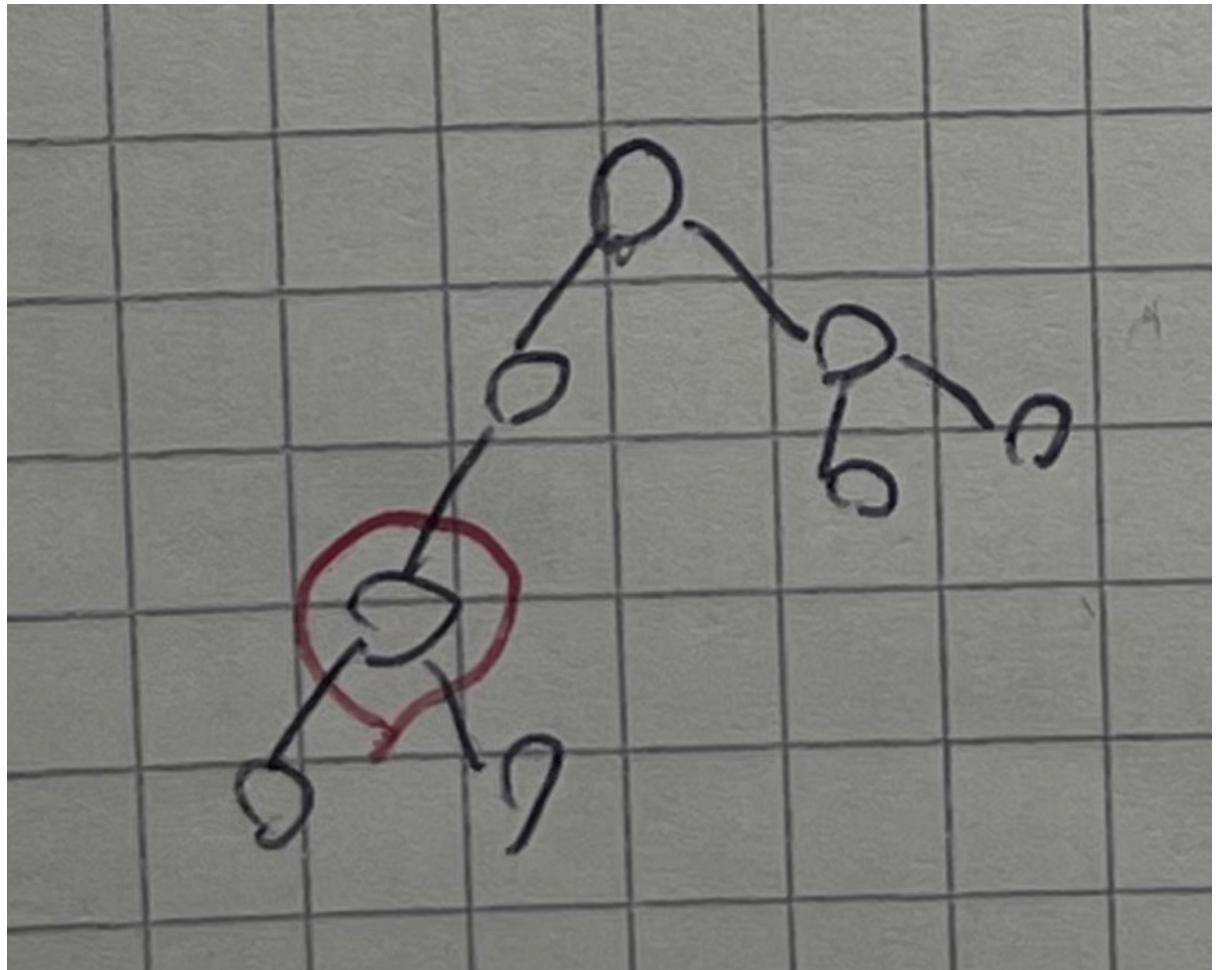


3) Il figlio destro di d è vuoto mentre il figlio sinistro no: rimpiazzo d con il figlio sinistro.

Procedimento analogo a quello sopra ma ragionando col figlio sinistro.



4) nessuno dei figli d è vuoto: chiamo delete Min:



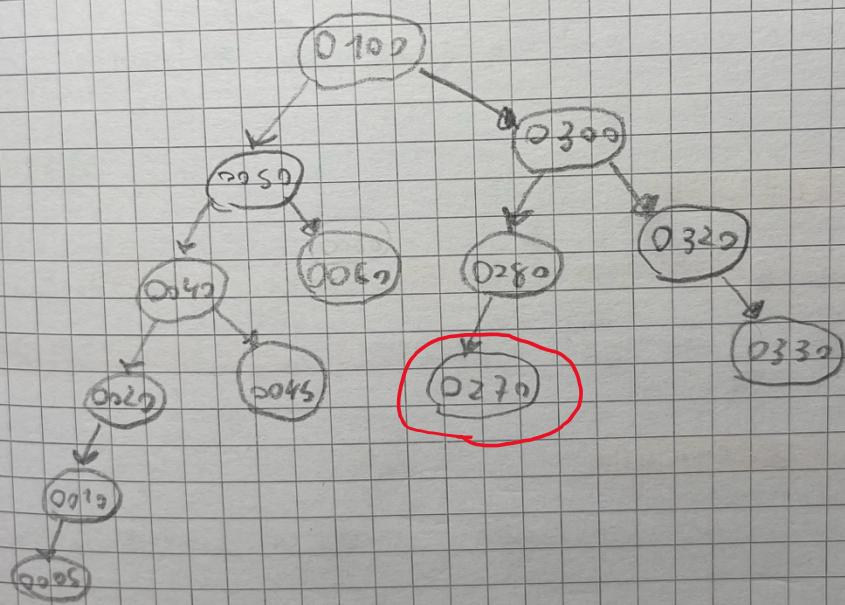
Esempio pratico(preso anche da un esame passato)

DeleteElem(100):

DOMANDA ESAME

RST (Aula WEB)

1)



2)

Si rappresenta il caso persone per mezzo di un diagramma  
di A, con N variabili dell'albero, per insieme S  
TETA(A) che appunto rappresenta il caso persone.

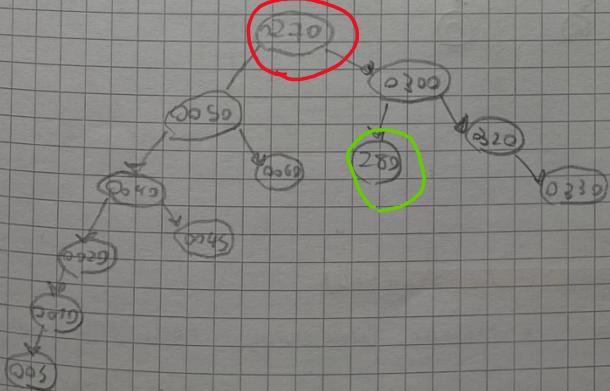
2)

CAPITATE UNA volta un nodo, per tollolo si può  
dare una o più future posizioni.

Dove usano la furtoare austriaca determinare che ci  
sono attualmente le nolu più piccole.

Troviamo il più piccolo dei campi, quindi in questo caso

determiniamo gli altri 270 e parla la sua chiave viene a  
determinare e bisognerebbe soprattutto /k/ 100 con > 70



non siamo nei campi perché da lì andiamo fino al fondo  
l'albero