

Consideriamo alberi binari i cui nodi hanno 2 campi interi (vedi programma dato): il solito campo info ed un nuovo campo num che indica per ciascun nodo r dell'albero il numero di nodi dell'albero radicato in r (r compreso). Il percorso infisso (a sinistra, radice, a destra) di un albero binario, determina un ordine totale dei suoi nodi. Assumiamo che se i nodi di un albero sono m (campo num della radice è m), allora l'ordine dei nodi va da 1 fino a m , dove 1 è il nodo che si raggiunge andando il più possibile verso sinistra e m quello che si raggiunge andando il più possibile verso destra.

Vogliamo una funzione ricorsiva che, dato un albero r ed un intero $n > 0$, restituisca il nodo n -esimo dell'albero (secondo l'ordine dell'attraversamento infisso), se c'è, e altrimenti, restituisca 0. Il campo num dei nodi dell'albero può essere molto utile per sapere se il nodo cercato è presente nell'albero e, in caso ci sia, per guidare la ricerca verso questo nodo senza mai visitare sottoalberi inutilmente.

Esempio: sia dato l'albero BST $[9,4]([3,2](_,[4,1](_,_)), [10,1](_,_))$, dove ogni nodo è rappresentato dai suoi 2 campi, info e num. Per esempio la radice è $[9,4]$ che indica che il campo info della radice è 9 e il campo num è 4. Infatti l'albero intero contiene 4 nodi. Si osservi che le foglie hanno ovviamente num=1. Se ora vogliamo cercare il nodo $n=1$ di quest'albero, dobbiamo restituire un puntatore al nodo $[3,2]$. Per il nodo $n=2$ dobbiamo restituire un puntatore al nodo $[4,1]$, per $n=3$ restituiamo un puntatore a $[9,4]$ e per $n=4$ a $[10,1]$. Per $n > 4$ dobbiamo restituire 0 in quanto ci sono solo 4 nodi.

Funzione da fare: si chiede di realizzare la funzione **ricorsiva** `nodo* trova(nodo*r, int n)` che soddisfa la seguente coppia di PRE e POST condizioni:

PRE=(albero(r) è benformato, $n > 0$)

POST=(se albero(r) contiene almeno n nodi, restituisce il puntatore al nodo numero n nell'ordinamento determinato dalla visita infissa, altrimenti restituisce 0)

Correttezza: dimostrare induttivamente la correttezza della funzione `trova`.

Attenzione: si osservi che PRE e POST devono valere per ogni invocazione. POST è specialmente interessante in quanto suggerisce che in ogni invocazione ricorsiva su un sottoalbero r , n deve essere il numero del nodo da cercare nel sottoalbero con radice r .

