Tutorato Lezione 5

Alberi binari e BST

Tutor: Maria Angela Pellegrino
mapellegrino@unisa.it

Albero - 1

Realizzare delle funzioni per:

- determinare l'altezza di un albero binario;
- il numero di nodi di un albero binario;
- realizzare una visita per livelli di un albero binario.

Albero - 1

altezza: 3

nodi: 9

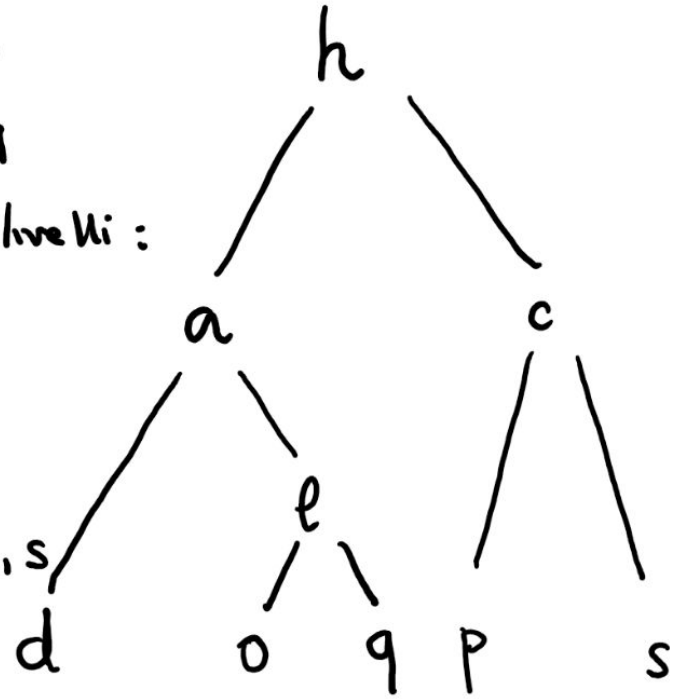
visita per livelli:

1 h

2 a, c

3 d, e

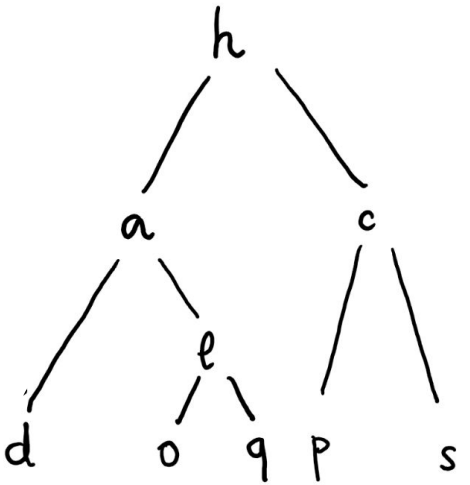
4 o, q, p, s



Albero - 2

Realizzare le tre visite (pre-order, in-order, post-order) dell'albero binario in **maniera iterativa**, con l'uso di uno stack.

Albero - 2



Pre order:

h, a, d, e, o, q, c, p, s

Post order:

d, o, q, e, a, p, s, c, h

In order / Simmetrica:

d, a, o, e, q, h, p, c, s

Esercizio 1

Contare il numero di elementi distinti

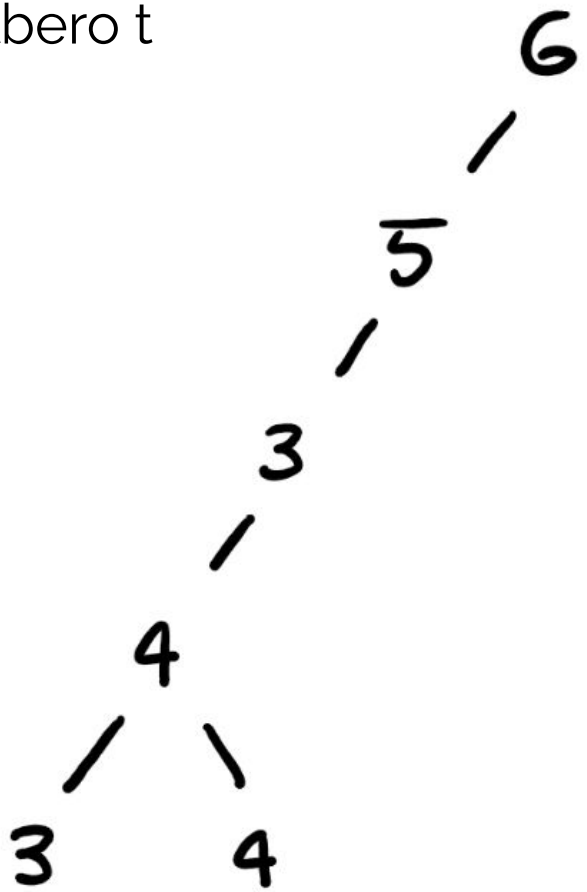
Pre condizioni
La funzione prende in ingresso un albero binario

Post condizioni
La funzione conta il numero di elementi distinti tramite una lista.

Esercizio 1

Input → Albero t

Output: 4



Esercizio 1

Soluzione

```
int countDistinti(BTree b){
    int nodes = nodesCount(b), count, pos;
    Stack s = newStack();
    BTree bTmp;
    List l = newList();
    push(s, b);
    for(count = 0; count != nodes; count++){
        bTmp = top(s);
        if(searchItem(l , bTmp->value, &pos) == NULL)
            addHead(l, bTmp->value);
        pop(s);
        if(!isEmptyTree(bTmp->right))
            push(s, bTmp->right);
        if(!isEmptyTree(bTmp->left))
            push(s, bTmp->left);
    }
    return sizeList(l);
}
```

Esercizio 2

Scrivere una funzione che, dato un BST, restituisca una lista ordinata dei suoi elementi.

Pre condizioni
La funzione prende in ingresso un BST

Post condizioni
La funzione restituisce una lista contenente gli elementi ordinati contenuti nel BST.

Esercizio 2

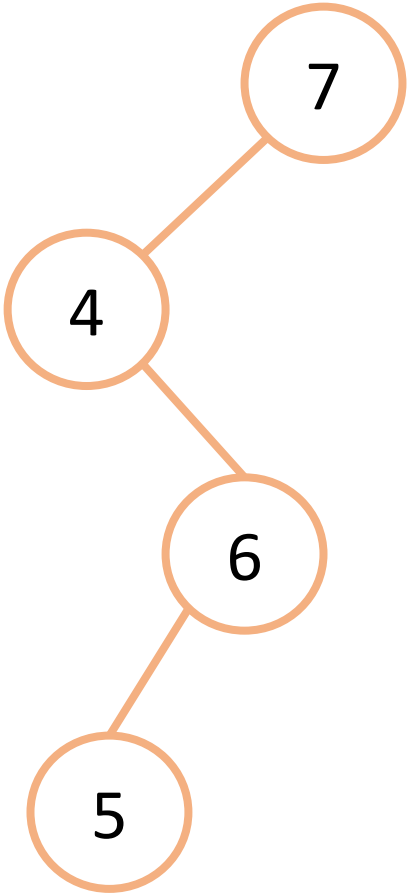
```
list toOrderedList(BST T){
        list lista = newList();
        if(emptyBST(T)) return lista;
        if(figlioSX(T)!=NULL ) {
                list listaSX = toOrderedList(figlioSX(T));
                for(int i=0; i<sizeList(listaSX); i++){
                        lista = insertList(lista, 0, getItemList(listaSX, i));
                }
        }
        lista = insertList(lista, 0, getItem(T));
        if(figlioDX(T)!=NULL ) {
                list listaDX = toOrderedList(figlioDX(T));
                for(int i=0; i<sizeList(listaDX); i++){
                        lista = insertList(lista, 0, getItemList(listaDX, i));
                }
        }
        return reverseList(lista);
}
```

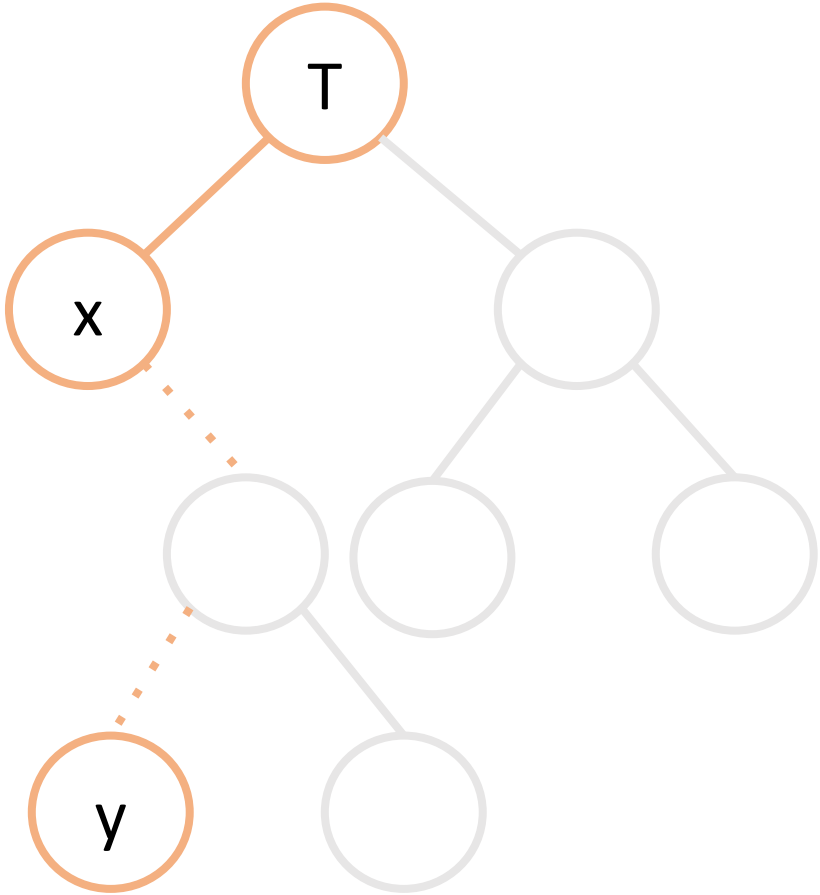
Esercizio 3

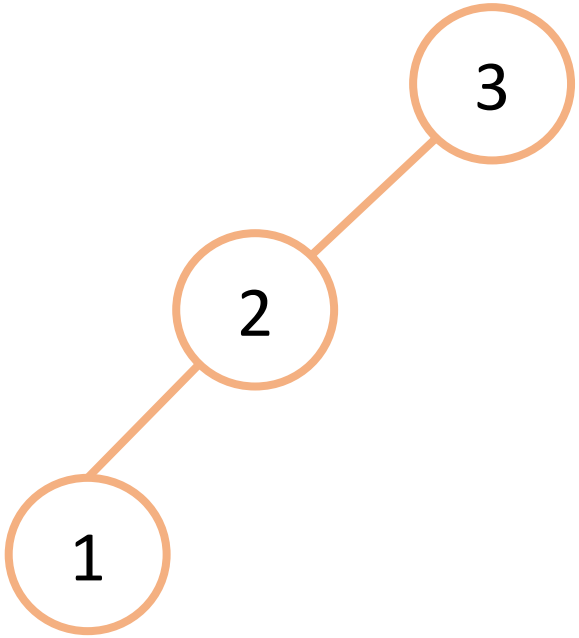
Scrivere una funzione che prende un BST T che restituisce i due valori più piccoli di T.

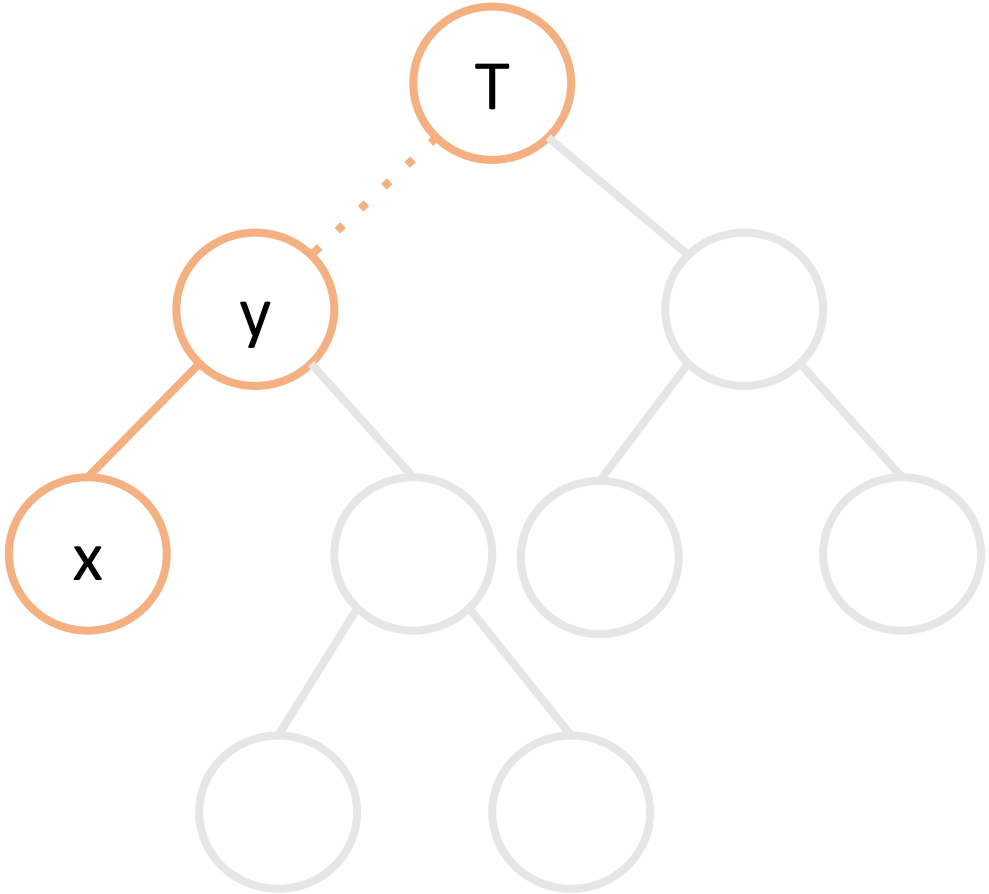
Pre condizioni
La funzione prende in ingresso un BST.

Post condizioni
La funzione restituisce un array di lunghezza contenente le 2 entrate con chiave più piccola di T.
Se T è vuoto o contiene meno di 2 entrate la funzione deve restituire null.


7
4
6
5


T
x
y


3
2
1


T
y
x

```
item* primiDueElementi (BST T){
	if(emptyBST(T) || countNode(T)<2) return NULL;
	static item elem[2];
	BST padre;

	while(figlioSX(T)!=NULL){
		padre = T;
		T = figlioSX(T);
	}

	elem[0]=getItem(T);
	if(figlioDX(T)!=NULL){
		T=figlioDX(T);
		while(figlioSX(T)!=NULL)
			T=figlioSX(T);
	}
	else
		T =padre;

	elem[1]=getItem(T);
	return elem;

}
```