Informatica Generale
Homework 3: Alberi Binari

Ivano Salvo – Sapienza Università di Roma

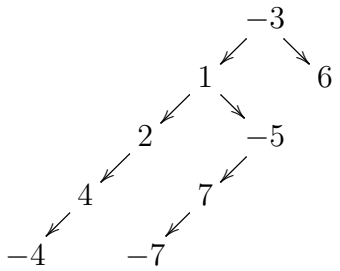
Esercizio 1 La distanza $d(u, v)$ tra due nodi u, v in un albero binario B è la lunghezza del cammino che li unisce (contando il numero di archi). Il diametro $D(B)$ di un albero binario, è la massima distanza tra due nodi arbitrari. Formalmente $D(B) = \max_{u,v \in B} d(u, v)$. Il diametro dell'albero vuoto è -1.

Scrivere una funzione C di prototipo:

```
int diameter(binTree B);
```

che preso come parametro di input un albero binario B ne calcola il diametro.

Esempio: attenzione che non sempre il cammino più lungo tra due nodi che realizza il diametro di un albero passa per la radice dell'albero. Ad esempio considerate il seguente albero:



Il diametro è 6 che corrisponde alla distanza tra i nodi -7 e -4 (nel sottoalbero radicato nel nodo etichettato 1), mentre la distanza tra i nodi -4 e 6 è 5.

Esercizio 2 Presi due nodi u, v in un albero binario B il *minimo antenato comune* (*low common ancestor*) è il nodo più profondo t, tale che u, v sono entrambi nell'albero radicato in t. Alternativamente, possiamo dire che t è la radice del più piccolo sottoalbero di B che contiene sia u che v. Se u e/o v non fossero presenti in B, il minimo antenato comune sarebbe l'albero vuoto.

Scrivere una funzione C di prototipo:

```
binTree lowCommonAncestor(binTree B, int u, int v)
  /* PREC: B contiene etichette tutte distinte */
```

che restituisce un puntatore alla radice del sottoalbero più piccolo che contiene le etichette u e v. Assumere le etichette dell'albero tutte distinte.

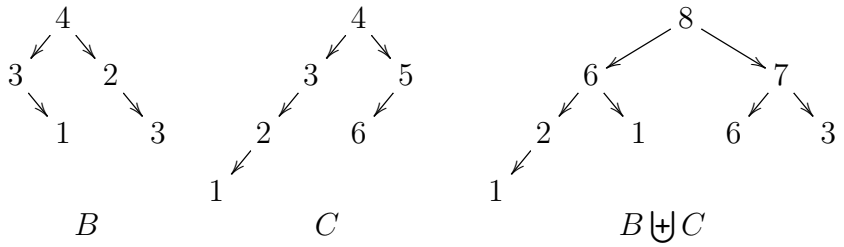
ESEMPI: Nell'albero mostrato nell'esempio dell'**Esercizio 1**, l'antenato comune di -7 e -4 è l'albero radicato in 1. L'antenato comune tra -5 e -7 è il sottoalbero radicato in -5.

Esercizio 3 Scrivere una funzione `C` di prototipo:

```
binTree sumUnionTree(binTree B, binTree C)
```

che, ricevendo come parametri di input due alberi binari di interi B e C, ritorna come risultato un *nuovo* albero la cui struttura contiene la struttura di entrambi gli alberi e nei nodi comuni contiene come etichetta la somma delle etichette dei nodi corrispondenti negli alberi in input. Nei nodi non comuni, vengono semplicemente ricopiate le etichette presenti nell'albero che la contiene.

ESEMPIO: Dati gli alberi B e C come in figura, l'albero che deve calcolare la funzione è l'albero $B \uplus C$ a destra in figura.



Note Pratiche: Potete usare le funzioni del file `binTree.c` e **dovete** usare le definizioni di tipo in `binTree.h`. In `binTree.c` troverete anche le funzioni `binTree buildTree(int[], int[], int)` che costruisce un albero binario da una visita in preordine e da una visita inordine memorizzate in un array, e una funzione `void printBinTree(binTree)` che dà in output una stampa parentesizzata dell'albero. Il loro utilizzo dovrebbe essere chiaro dai files `hw3-main-*.c` forniti. Potrebbero essere utili in fase di debugging.

Attenzione. Nel vostro file dovrà esserci un `#include "binTree.h"` e **non** dovete ridefinire le funzioni di `binTree.c`.