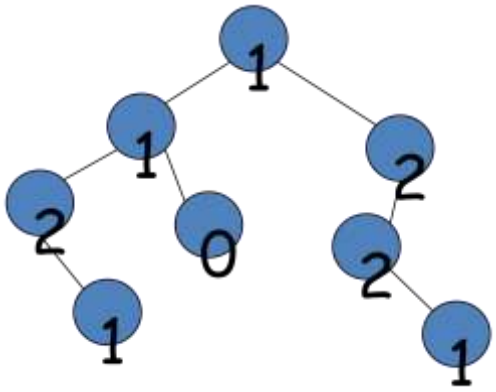
Esercizio 3 del 22/5/2015 da consegnare per il 3/6

Abbiamo un albero binario r un valore intero y e k>=0 e vogliamo sapere se in r c'è un **cammino che si estende dalla radice fino ad una foglia** che contiene esattamente k nodi con campo info uguale a y. Nel seguito i cammini su un albero binario saranno rappresentati con una sequenza di 0/1 terminante con -1 a indicare appunto la fine del cammino. Quindi il cammino -1 è il cammino vuoto che coincide con la radice.

Esempio: se il nostro albero r fosse il seguente:



e k=2 e y=1 allora un cammino con esattamente due 1 sarebbe il cammino 01-1. C'è anche un altro cammino con esattamente due 1 ed è il cammino 101-1. Invece i cammini 0-1 e 00-1, che avrebbero due 1 non andrebbero bene perché non arrivano ad una foglia. Se invece k=1 e y=2 il cammino 001-1 soddisferebbe la richiesta e lo stesso cammino andrebbe bene per k=3 e y=1. Invece nessun cammino di albero(r) soddisfa k=0, y=1, visto che la radice contiene 1. Si osservi che un albero vuoto non ha cammini, quindi tanto meno ha cammini con una data proprietà. Questo giustifica la pre-condizione di cerca_cam che segue.

Per risolvere il problema appena descritto, si chiede di realizzare una funzione ricorsiva:

bool cerca_cam(nodo*r, int k, int y, int*C) che soddisfa la seguente pre- e post-condizione:

PRE_cerca=(albero(r) è corretto e non vuoto, k>=0 e y valore qualsiasi, C ha almeno tanti elementi quanta è l'altezza di albero(r))

POST_cerca=(restituisce true sse in r esiste un cammino da r ad una foglia con esattamente k nodi con campo info=y e false altrimenti, in caso restituisca true, C contiene una sequenza (anche vuota) di 0/1 che termina con -1 e che individua il cammino più a sinistra in albero(r) con esattamente k y).

Correttezza: fare la prova induttiva di correttezza di cerca_cam.

Viene fornito un main che esegue le letture, la costruzione dell'albero su cui viene invocata cerca_cam e l'output finale. Il main assume che dim>0 e che quindi l'albero R costruito da rep_ins non possa essere vuoto. Quindi la prima invocazione di trova_cam soddisfa la pre-condizione di trova_cam.