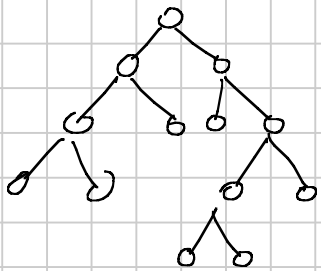
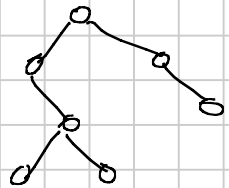
Es 1 : Verificare se un albero binario è completo



SI



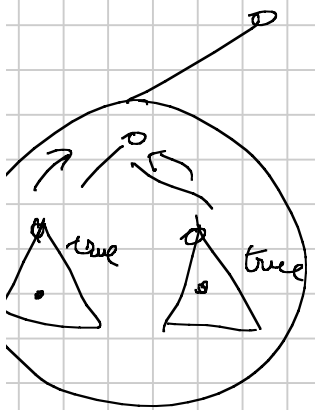
NO

Ogni nodo di un albero completo ha sul figliosinistro che figliodestro $\neq \emptyset$

```
Completo(u):
if (u == NULL || (u.left == NULL && u.right == NULL))
        return true;

if (u.left == NULL || u.right == NULL) return false;

return Completo(u.sx) && Completo(u.dx);
```




```
Perf_Bil (u)                                   < bil , alt >
if (u == NULL)   return   < true , -1 >
   else {
< bil_sx , alt_sx > = Perf_Bil (u.sx);
< bil_dx , alt_sx > = Perf_Bil (u.dx);
bil = ( bil_sx && bil_dx && ( alt_sx = alt_dx ) );
alt = max (alt_sx , alt_dx) + 1;
return ( bil , alt );
}
```

$$T(n) = T(n_s) + T(n_d) + \Theta(1)$$

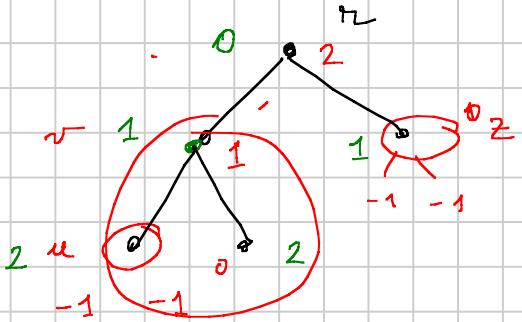
$$T(n) \leq 2\,T\left(\frac{n}{2}\right) + \Theta(1)$$

$$\Theta(n)$$

```
contafoglie (u) :
if u == NULL return ∅;
if (u.sx == NULL && u.dx = NULL) return 1;
return contafoglie (u.sx) + contafoglie (u.dx);
```

nodo u cardine se $h(u) = p(u)$ altezza = profondità


```
Cardine (u, p)   1ª chiamata Cardine (u, 0)
if (u == NULL) return -1;
else {
• altezza_sx = Cardine (u.sx, p+1);
• altezza_dx = Cardine (u.dx, p+1);
altezza = max (altezza_sx, altezza_dx) + 1
if (p == altezza) print u.key;
return altezza;
}
```



v è cardine

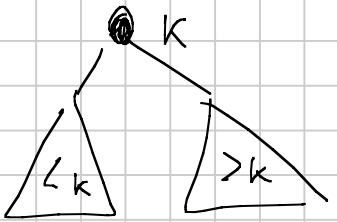
$\Theta(n)$

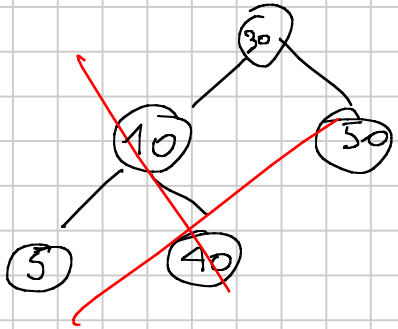
Titolo nota 17/03/2016

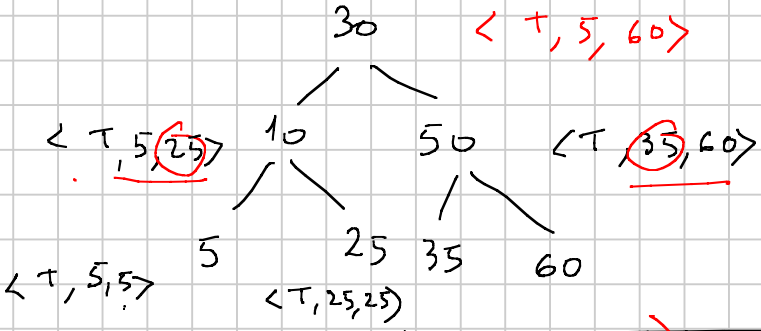
ABR

Verificare se un albero binario è un albero binario di ricerca

Definizione







Sol 1 IN-VISITA: memorizzando in array A + controllo

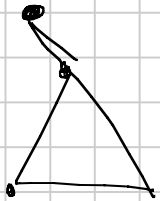
Tempo: $\Theta(n)$, $\Theta(n)$ spazio

Titolo nota 17/03/2016

Sol. 2 : Invisita con controllo locale ricordandosi l'ultimo visitato $\Theta(n)$ tempo

Sol. 3 :

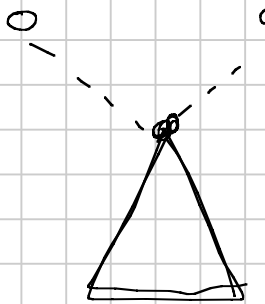
<min, max>



< min, max >

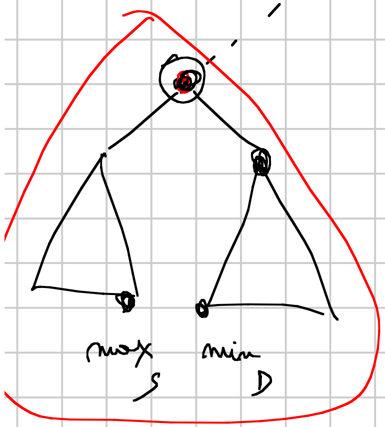
u. key > max

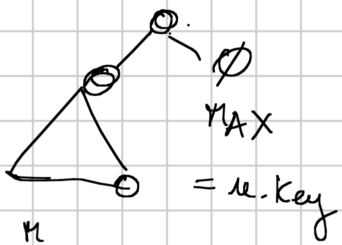
u. key < min



< min, max >

Usando puntatore al padre

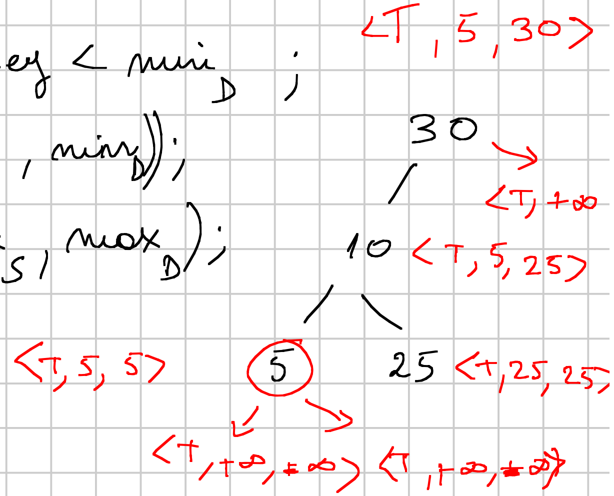




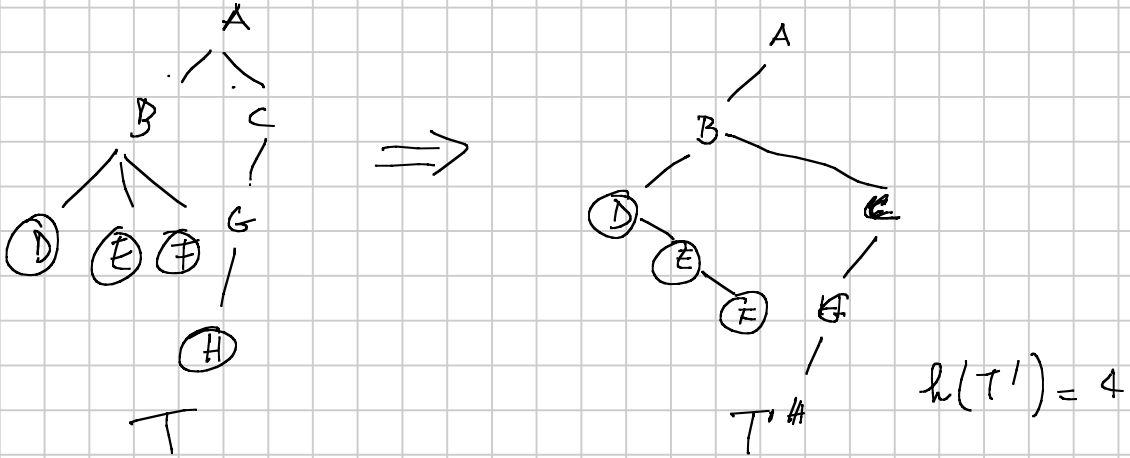

```
ABR(u):                                  <isABR, min, max>
if (u == NULL)   return  ( true, +∞, -∞ > ;

<isABR_S, min_S, max_S> = ABR(u.sx);
<isABR_D, min_D, max_D> = ABR(u.dx);

isABR = isABR_S && isABR_D && max_S < u.key < min_D ;
min = Math.min(u.key, Math.min(min_S, min_D));
max = Math.max(u.key, Math.max(max_S, max_D));
return <is_ABR, min, max>;
```



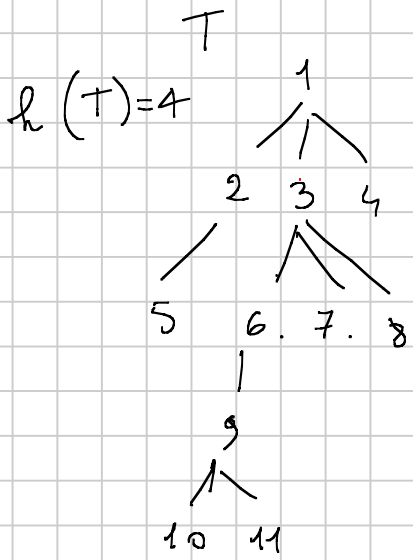
Contare le foglie di un albero dalla sua immagine binaria

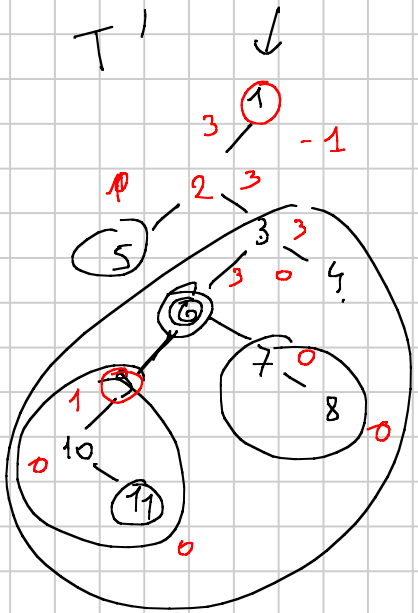


foglia di T
se figli$_{sx}$ = $\emptyset$

$h(T') = 4$

Calcolare l'altezza di T da T'





$h(T') = 6$

$= 4$

0

$\max(h_{SIN} + 1, h_{DES})$

Calcolo l'altezza della radice di T.

$\max(-1+1, 0)$

```
Alt(u):
if (u == NULL) return -1;
alt_SIN = Alt(u.sx);
alt_DES = Alt(u.dx);
alt = max(alt_SIN + 1, alt_DES);
return alt;
```