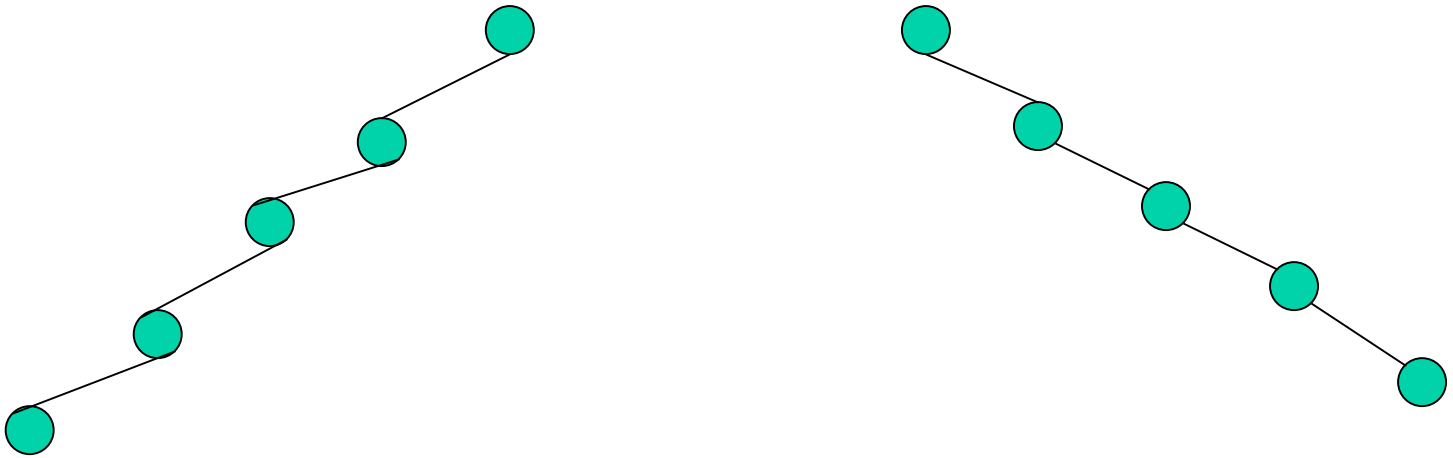
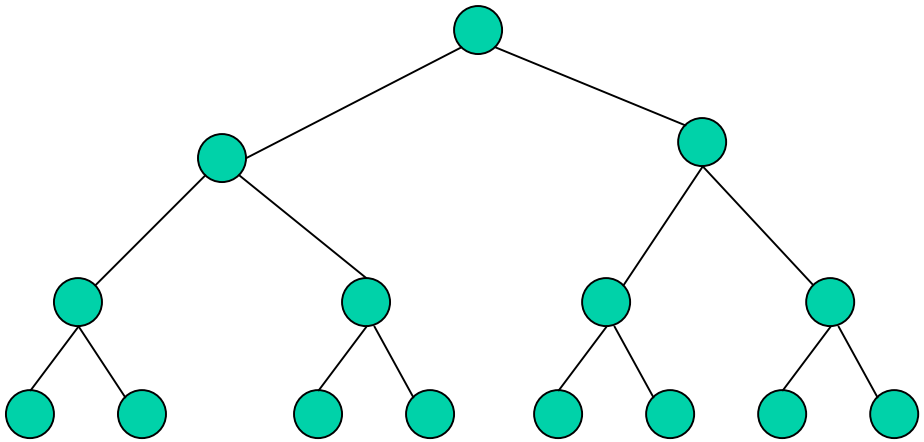
alberi completamente sbilanciati



Il numero dei nodi al livello i è 1, per 0≤ i ≤ altezza

L'altezza h di un albero completamente sbilanciato con n nodi è h = n-1

Un albero **completamente bilanciato** o **pieno** (full)



Definizione: Un albero è pieno se tutte le foglie sono sullo stesso livello e ogni nodo non foglia ha due figli.

Il numero dei nodi al livello i è 2^i, per $0 \leq i \leq$ altezza

Il numero dei nodi n di un albero pieno di altezza h è $n = 2^{h+1} - 1$

Per un albero binario qualsiasi di altezza h e numero di nodi n vale $\log_2(n) < h+1 \leq n$

$$\log_2(n) < h+1 \leq n$$

Per ottenere questi limiti per l'altezza di un albero binario in funzione del numero dei nodi osserviamo che:

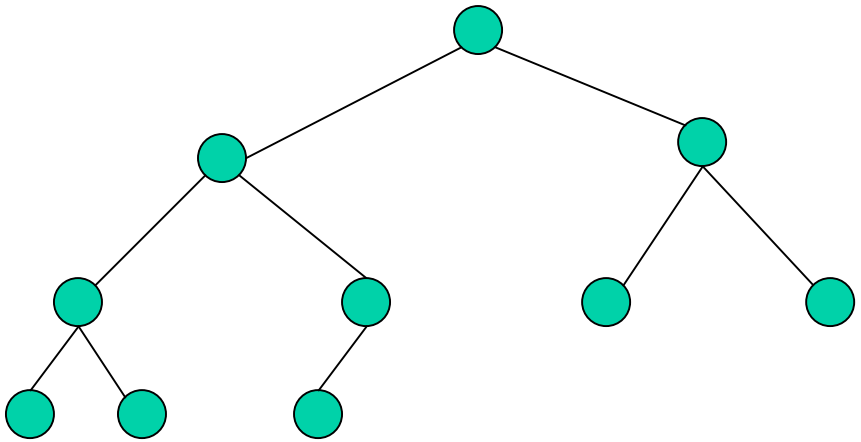
1. **l'albero completamente sbilanciato con n nodi ha altezza massima a parità di nodi. In altre parole è il più alto albero binario con n nodi, infatti ha un solo nodo per ogni livello. L'altezza è $h = n-1$.**

Quindi qualsiasi albero binario con n nodi ha altezza $h \leq n-1$, da cui $h+1 \leq n$.

2. **l'albero pieno ha il maggior numero di nodi fra gli alberi binari della stessa altezza. Il numero di nodi è $n = 2^{h+1} - 1$.**

Quindi qualsiasi albero binario di altezza h ha un numero di nodi $n \leq 2^{h+1} - 1$, da cui $n < 2^{h+1}$, e passando ai logaritmi si ha $\log_2(n) < h+1$

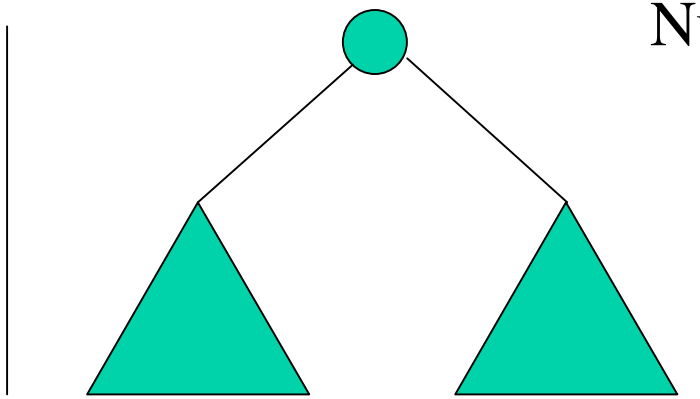
Un albero binario completo



Definizione: un albero binario è completo se tutti i livelli, tranne al più il massimo, hanno il massimo numero di nodi e le foglie sul livello massimo sono più a sinistra possibile.

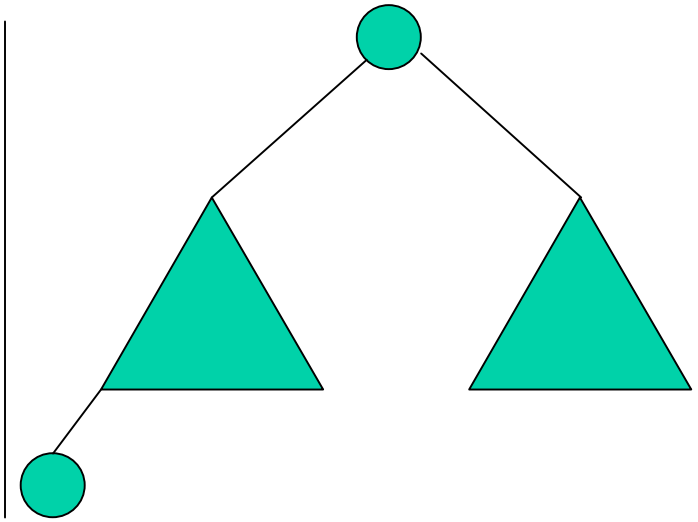
Per ogni n esiste un unico albero binario completo con n nodi

Alt = h-1



Numero nodi $n = 2^h - 1$

Alt = h

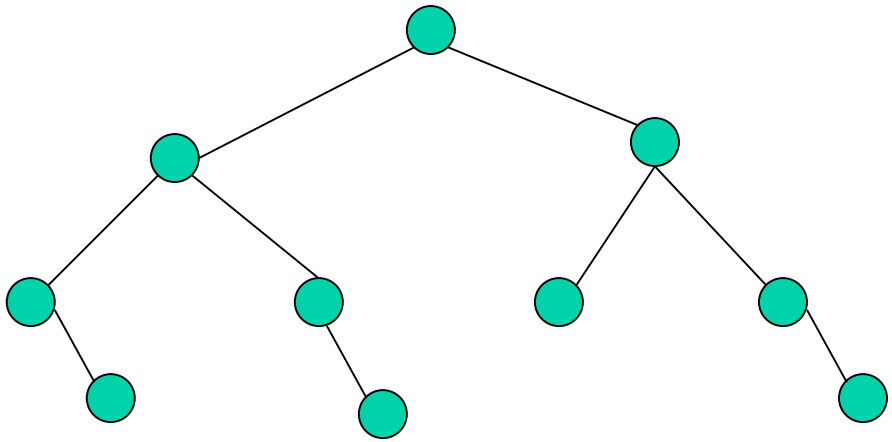


Numero nodi $n = 2^h$

posso aggiungere altri $2^h - 1$ nodi, senza aumentare l'altezza.

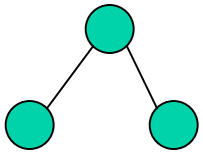
$2^h \leq n \leq 2^{h+1} - 1$, da cui
$h = \lfloor \log_2(n) \rfloor$, $n \geq 1$.

alberi n-bilanciati

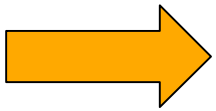


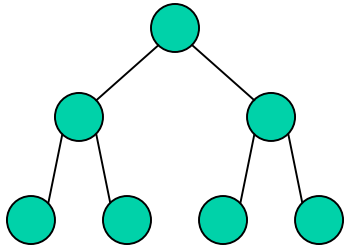
Definizione: Un albero è bilanciato nel Numero dei Nodi, brevemente n-bilanciato, quando, per ogni sottoalbero t radicato in un suo nodo, il numero dei nodi del sottoalbero sinistro di t meno il numero dei nodi del sottoalbero destro di t è in valore assoluto al più 1.

Anche per gli alberi n-bilanciati $h = \lfloor \log_2(n) \rfloor$, $n \geq 1$.

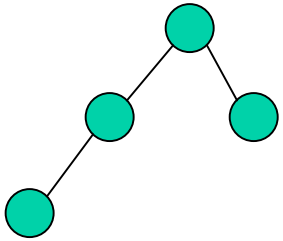


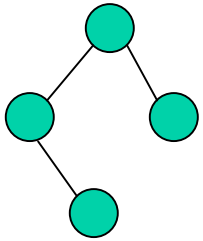
$n = 2^2-1$ nodi

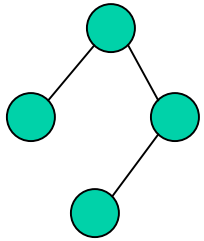


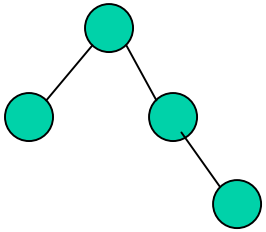


$n = 2^3-1$ nodi

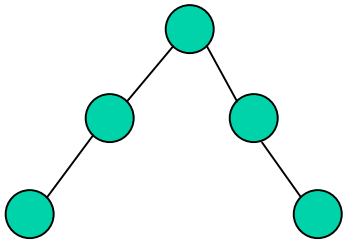






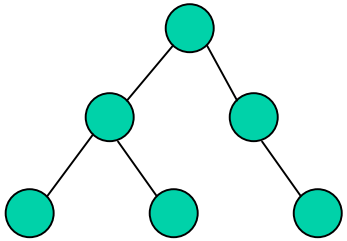


$n = 2^2$ nodi



…

$n = 2^2+1$ nodi

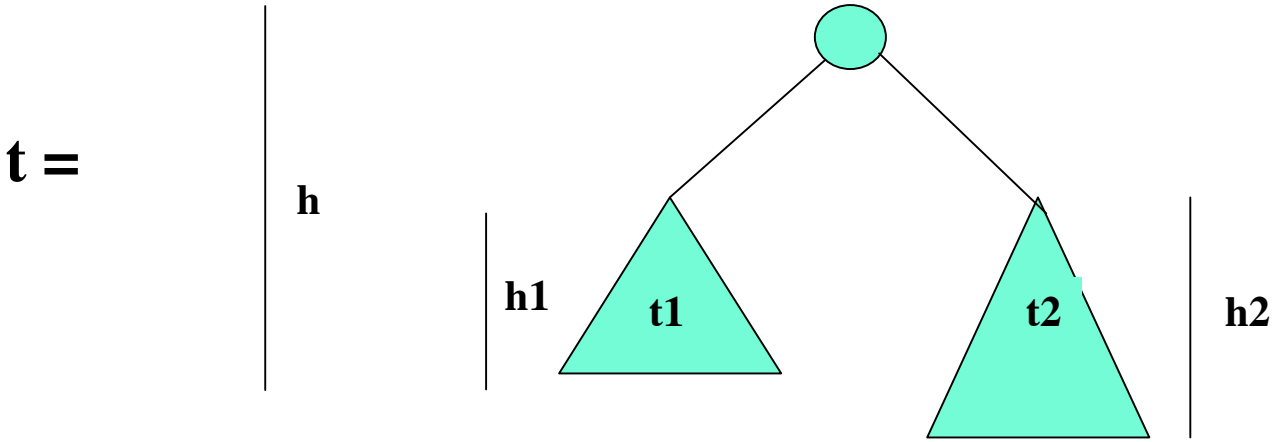


…

$n = 2^2+2$ nodi

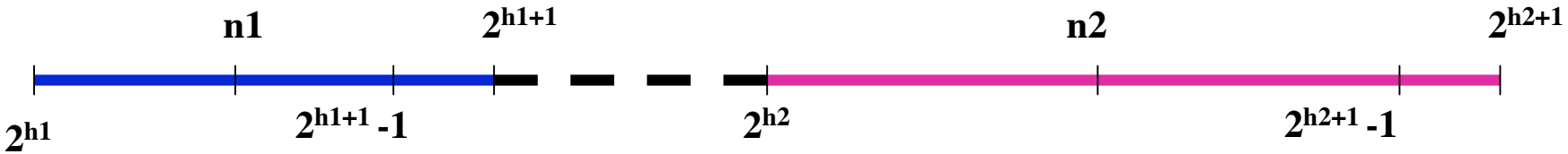
Se t è un albero n-bilanciato con numero dei nodi n e altezza h,vale $2^h \leq n \leq 2^{h+1} - 1$, per ogni n≥1.
Dimostriamolo per induzione sul numero dei nodi.

Per n = 1 è banalmente vero (l'altezza è 0).
Sia vero per tutti gli alberi con meno di n nodi e sia t un albero con n nodi.



n = n1 + n2 + 1
h = max(h1,h2) +1

Per ipotesi induttiva $2^{h1} \leq n1 \leq 2^{h1+1} - 1$ e $2^{h2} \leq n2 \leq 2^{h2+1} - 1$, graficamente:



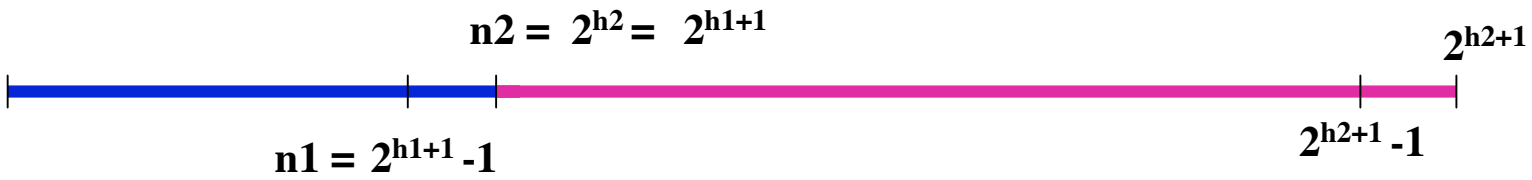
Ma $|n1-n2| \leq 1$!!

Poichè $|n1-n2| \leq 1$, possiamo avere i seguenti casi, assumendo, senza ledere la generalita', che $n2 \geq n1$:

Caso 1: n1 = n2, per cui h1 = h2

Caso 2: n2 = n1+1 e entrambi cadono nello stesso intervallo, per cui h1 = h2

Caso 3: n2 = n1+1 e cadono in intervalli diversi ma successivi, per cui h2 = h1 +1



$n = n1 + n2 + 1$
$h = \max(h1,h2) + 1$

Caso 1: n1 = n2, per cui h1 = h2

Per ipotesi induttiva $2^{h1} \le n1=n2 \le 2^{h1+1} -1$, quindi

$2*2^{h1} \le 2*n1 \le 2*2^{h1+1} -2$ da cui

$2^{h1+1} \le 2*n1+1 \le 2^{h1+2} -1$ da cui la tesi essendo $n = 2*n1+1$ e $h = h1+1$

Caso 2: n2 = n1+1 e entrambi cadono nello stesso intervallo, per cui h1 = h2

Per ipotesi induttiva $2^{h1} \le n1 \le 2^{h1+1} -1$, e $2^{h1} \le n2 \le 2^{h1+1} -1$ quindi

$2*2^{h1} \le n1+n2 \le 2*2^{h1+1} -2$ da cui

$2^{h1+1} \le n1+n2+1 \le 2^{h1+2} -1$ da cui la tesi essendo $n = n1+n2+1$ e $h = h1+1$

Caso 3: n2 = n1+1 e cadono in intervalli diversi ma successivi, per cui h=h2 = h1 +1

Per ipotesi induttiva $2^{h1} \le n1 \le 2^{h1+1} -1$, e $2^{h2} \le n2 \le 2^{h2+1} -1$, ma allora deve essere

$n1 = 2^{h2} -1$ e $n2 = 2^{h2}$ da cui

$n1+n2 = 2^{h2} -1 + 2^{h2}$

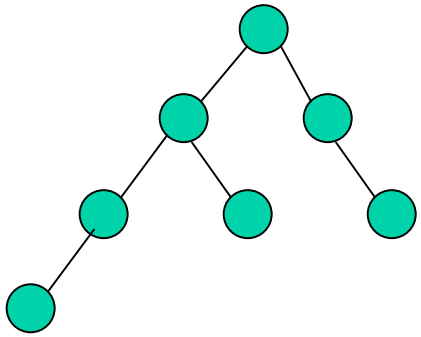
da cui $n1+n2+1 = 2^{h2} + 2^{h2} = 2^{h2+1}$

Analogamente si dimostra che in un albero n-bilanciato le altezze dei sottoalberi sinistro e destro dell'albero radicato in ogni nodo differiscono in valore assoluto al più di uno.

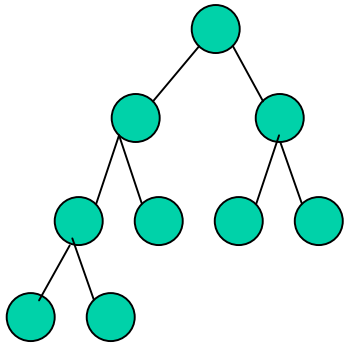
Definizione: Un albero è bilanciato in altezza , brevemente h-bilanciato, quando, per ogni sottoalbero t radicato in un suo nodo, l'altezza del sottoalbero sinistro di t meno l'altezza del sottoalbero destro di t è in valore assoluto al più 1.

Quindi possiamo dire che ogni albero bilanciato nel numero dei nodi è anche bilanciato in altezza.

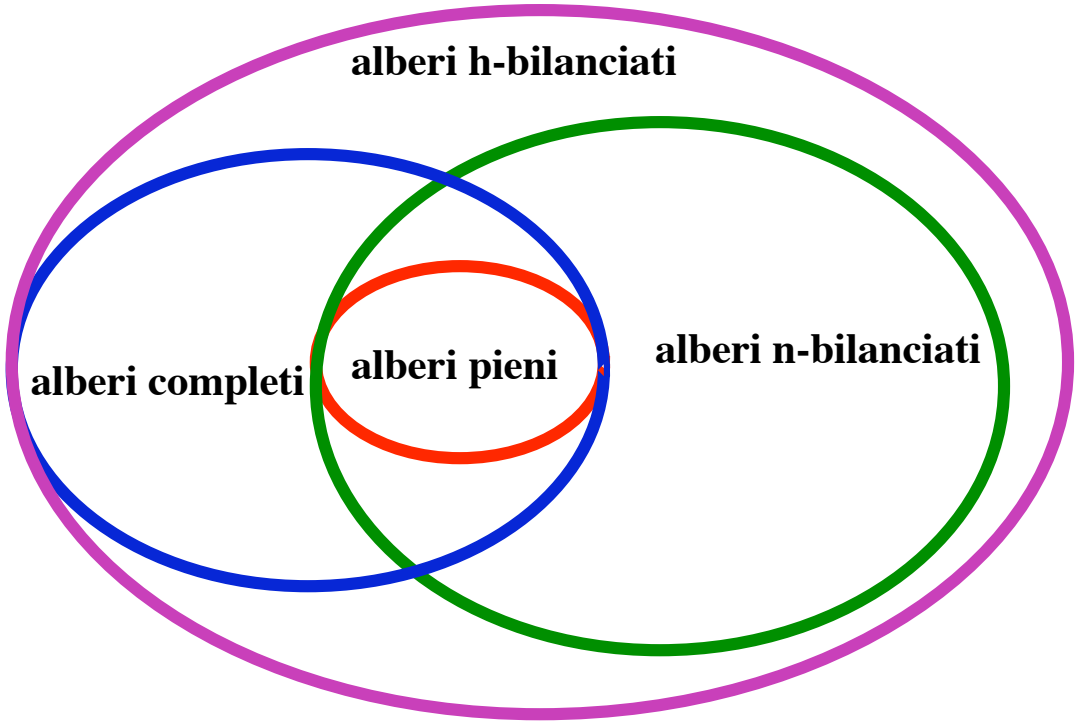
Che rapporto c'è tra le classi di albero introdotte fino ad ora?



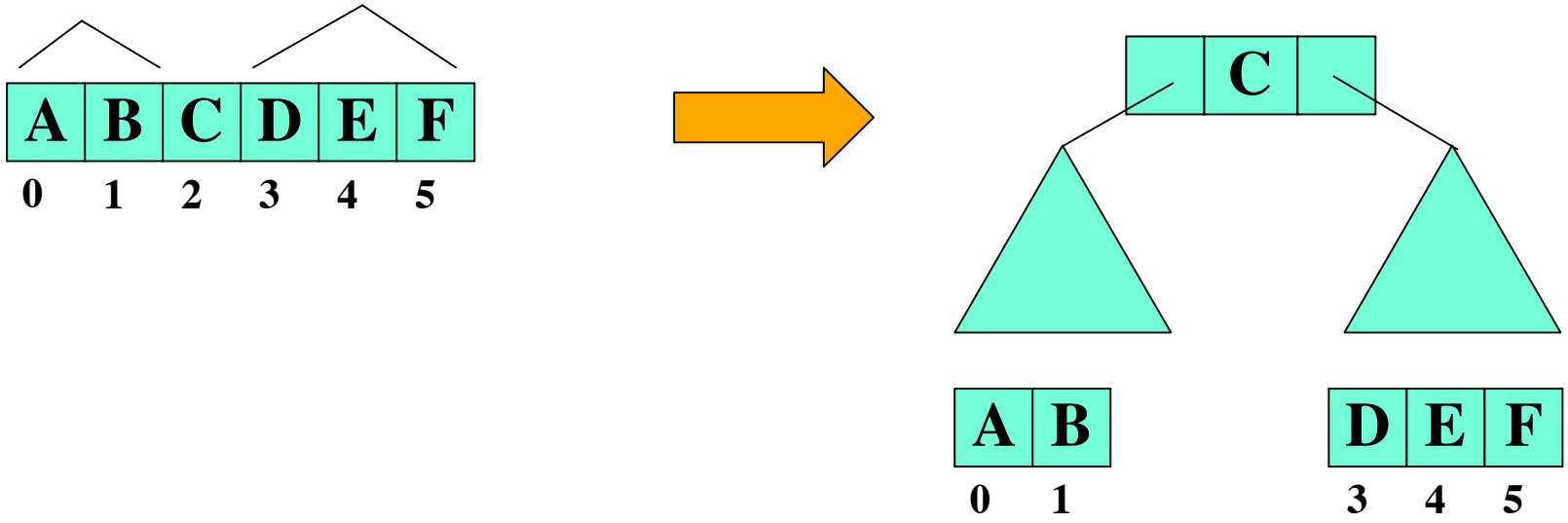
esempio di albero bilanciato in altezza e non bilanciato nel numero dei nodi



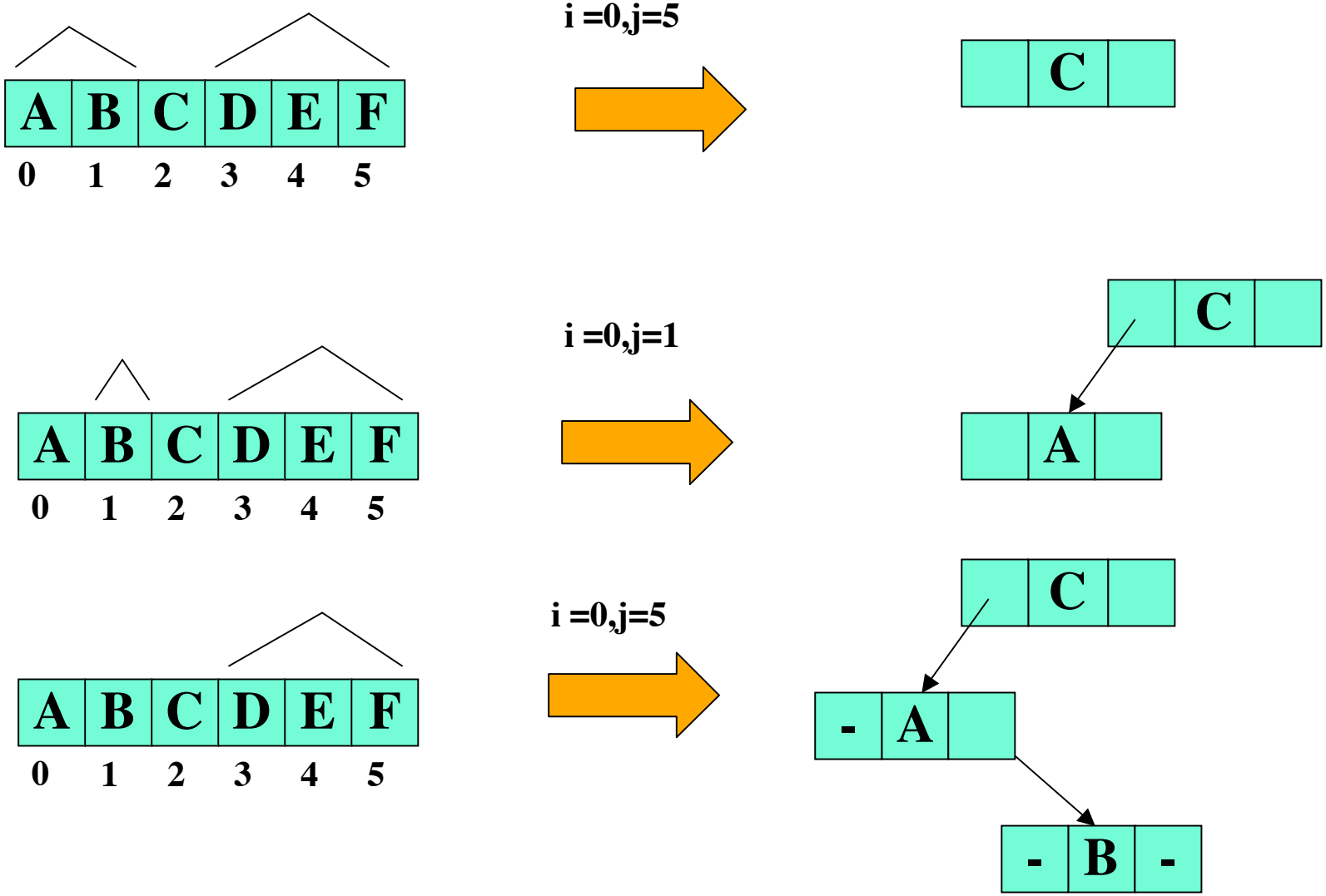
Questo e' un esempio di albero completo non bilanciato nel numero dei nodi

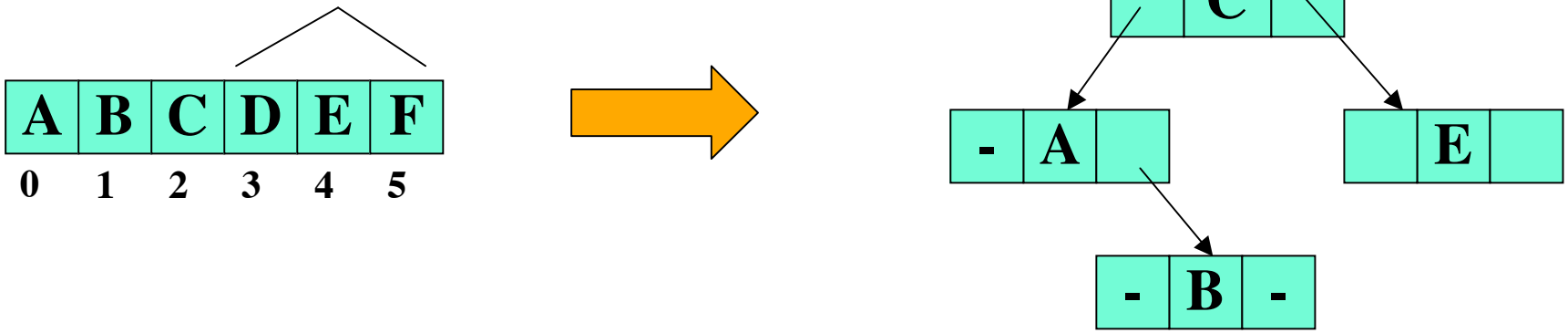


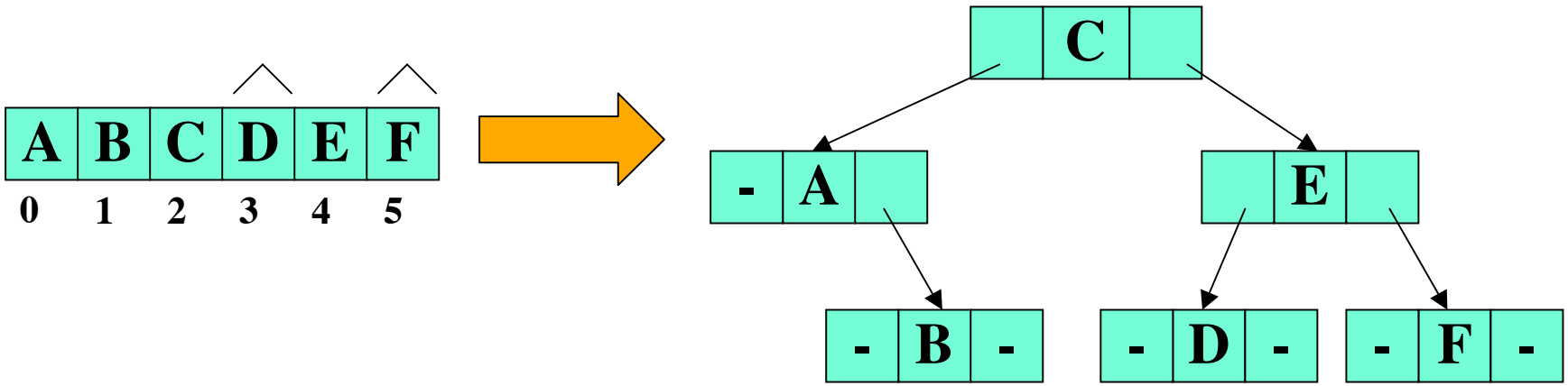
A partire da un vettore di interi possiamo costruire ricorsivamente un albero bilanciato nel numero di nodi selezionando a ogni chiamata una radice alla metà del vettore su cui si effettua la chiamata.



```
TreePtr AlbBinBil(int * vett, int i, int j)
/*costruisce un albero n-bilanciato a partire da un vettore  di indici i e j
* prec: vett!=NULL && i ≥0 && j≥0
*postc: t punta alla radice di un albero n-bilanciato che contiene vett[i],…,vett[j]*/
{int m;
TreePtr t;
if (i <= j)
  {m =  (i+j)/2;
    t = malloc(sizeof(Node));
    if (t)   t -> elem = vett[m];
    else
 /* malloc ha restituito un puntatore nullo */
fprintf(stderr,"%d non inserito, memoria non disponibile per  AlbBibBil.\n",vett[m]);
t->lPtr = AlbBinBil(vett,i,m-1);
t->rPtr = AlbBinBil(vett,m+1,j);
    return t;
  }
return NULL;
}
```


A B C D E F
0 1 2 3 4 5
i =0,j=5
C
A B C D E F
0 1 2 3 4 5
i =0,j=1
C
A
A B C D E F
0 1 2 3 4 5
i =0,j=5
C
- A
- B -


A B C D E F
0 1 2 3 4 5
C
- A
E
- B -


A B C D E F
0 1 2 3 4 5
C
- A
E
- B -
- D -
- F -

Esercizio 1: **si definisca una funzione che dovrà restituire "vero" se l'albero è bilanciato nel numero dei nodi, "falso" altrimenti.**

Esercizio 2: **si definisca una funzione che dovrà restituire "vero" se l'albero è bilanciato in altezza, "falso" altrimenti.**