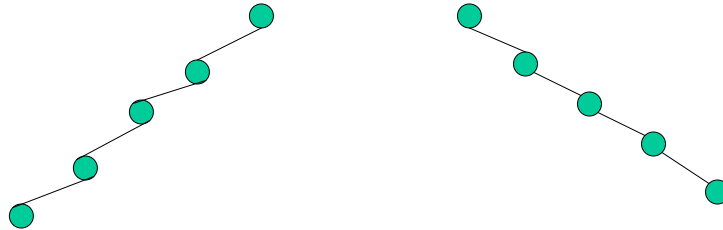


alberi completamente sbilanciati

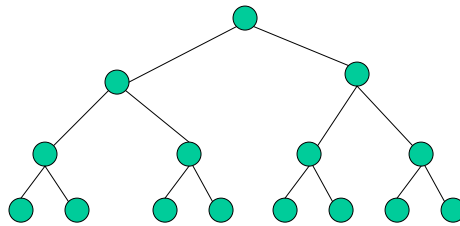


Il numero dei nodi al livello i è 1, per $0 \leq i \leq$ altezza

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

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

2

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

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 $n-1$.

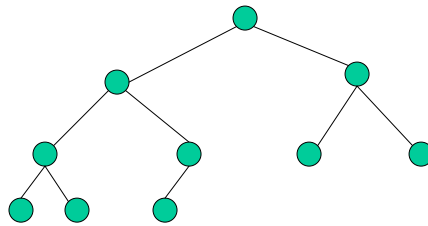
Quindi qualsiasi albero binario con n nodi ha altezza $h \leq n-1$, da cui $h+1 \leq n$.

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

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 $\log_2(n) < h+1$

3

Un albero binario completo

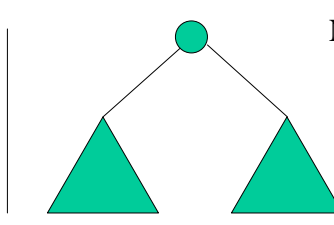


Definizione: un albero binario è **completo** se tutti i livelli, tranne al più il massimo, hanno il massimo numero di nodi e le foglie sul livello massimo sono più a sinistra possibile.

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

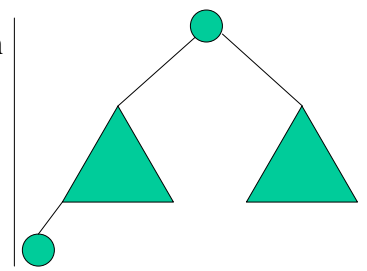
4

Alt = h-1



Numero nodi $n = 2^h - 1$

Alt = h



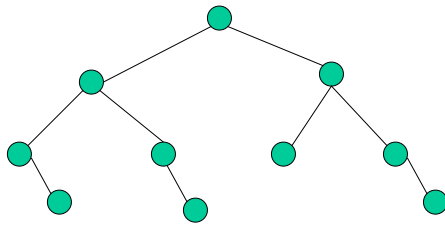
Numero nodi $n = 2^h$

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

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

5

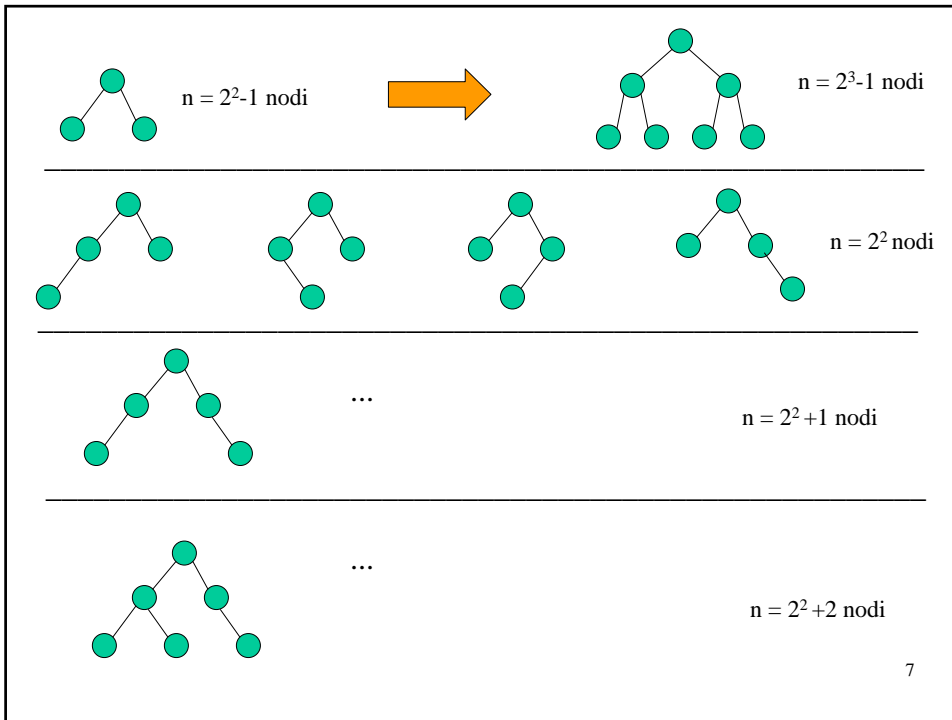
alberi n-bilanciati



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** $h = \lfloor \log_2(n) \rfloor, n \geq 1$.

6



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 $n \geq 1$.
 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.

$n = n_1 + n_2 + 1$
 $h = \max(h_1, h_2) + 1$

Per ipotesi induttiva $2^{h_1} \leq n_1 \leq 2^{h_1+1} - 1$ e $2^{h_2} \leq n_2 \leq 2^{h_2+1} - 1$, graficamente:

Ma $|n_1 - n_2| \leq 1$!!

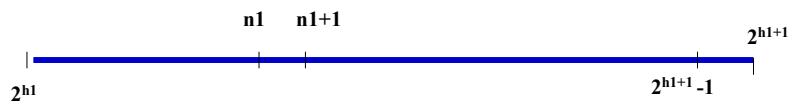
8

Poichè $|n_1 - n_2| \leq 1$, possiamo avere i seguenti casi, assumendo, senza ledere la generalità, che $n_2 \geq n_1$:

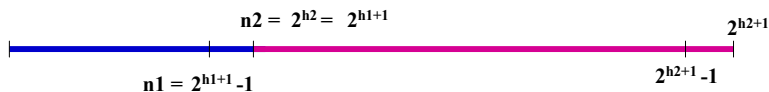
Caso 1: $n_1 = n_2$, per cui $h_1 = h_2$



Caso 2: $n_2 = n_1 + 1$ e entrambi cadono nello stesso intervallo, per cui $h_1 = h_2$



Caso 3: $n_2 = n_1 + 1$ e cadono in intervalli diversi ma successivi, per cui $h_2 = h_1 + 1$



9

$$\begin{aligned} n &= n_1 + n_2 + 1 \\ h &= \max(h_1, h_2) + 1 \end{aligned}$$

Caso 1: $n_1 = n_2$, per cui $h_1 = h_2$

Per ipotesi induttiva $2^{h_1} \leq n_1 = n_2 \leq 2^{h_1+1} - 1$, quindi

$$2 * 2^{h_1} \leq 2 * n_1 \leq 2 * 2^{h_1+1} - 2 \text{ da cui}$$

$$2^{h_1+1} \leq 2 * n_1 + 1 \leq 2^{h_1+2} - 1 \text{ da cui la tesi essendo } n = 2 * n_1 + 1 \text{ e } h = h_1 + 1$$

Caso 2: $n_2 = n_1 + 1$ e entrambi cadono nello stesso intervallo, per cui $h_1 = h_2$

Per ipotesi induttiva $2^{h_1} \leq n_1 \leq 2^{h_1+1} - 1$, e $2^{h_1} \leq n_2 \leq 2^{h_1+1} - 1$ quindi

$$2 * 2^{h_1} \leq n_1 + n_2 \leq 2 * 2^{h_1+1} - 2 \text{ da cui}$$

$$2^{h_1+1} \leq n_1 + n_2 + 1 \leq 2^{h_1+2} - 1 \text{ da cui la tesi essendo } n = n_1 + n_2 + 1 \text{ e } h = h_1 + 1$$

Caso 3: $n_2 = n_1 + 1$ e cadono in intervalli diversi ma successivi, per cui $h = h_2 = h_1 + 1$

Per ipotesi induttiva $2^{h_1} \leq n_1 \leq 2^{h_1+1} - 1$, e $2^{h_2} \leq n_2 \leq 2^{h_2+1} - 1$, ma allora deve essere

$$n_1 = 2^{h_2} - 1 \text{ e } n_2 = 2^{h_2} \text{ da cui}$$

$$n_1 + n_2 = 2^{h_2} - 1 + 2^{h_2}$$

$$\text{da cui } n_1 + n_2 + 1 = 2^{h_2} + 2^{h_2} = 2^{h_2+1}$$

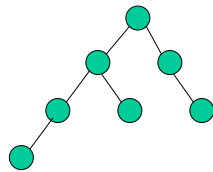
10

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.

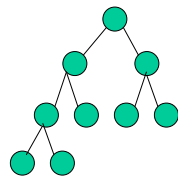
Quindi possiamo dire che ogni albero bilanciato nel numero dei nodi è anche bilanciato in altezza.

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

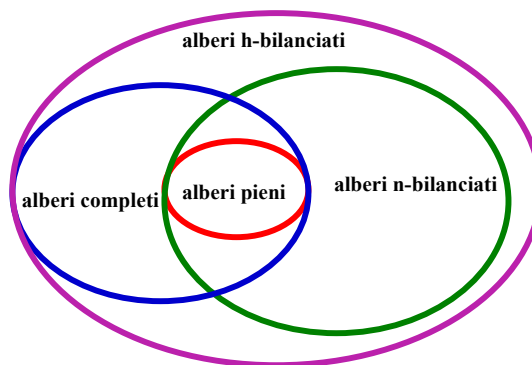


esempio di albero bilanciato in altezza e non bilanciato nel numero dei nodi

11

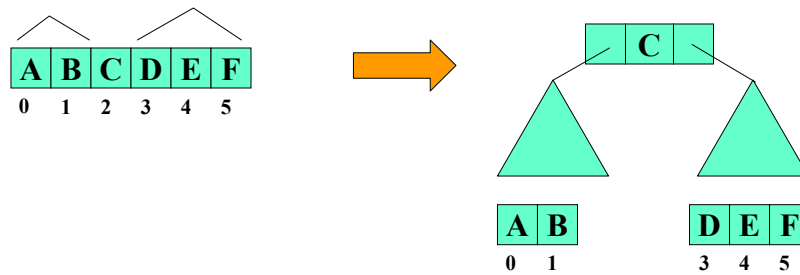


Questo è un esempio di albero completo non bilanciato nel numero dei nodi



12

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.



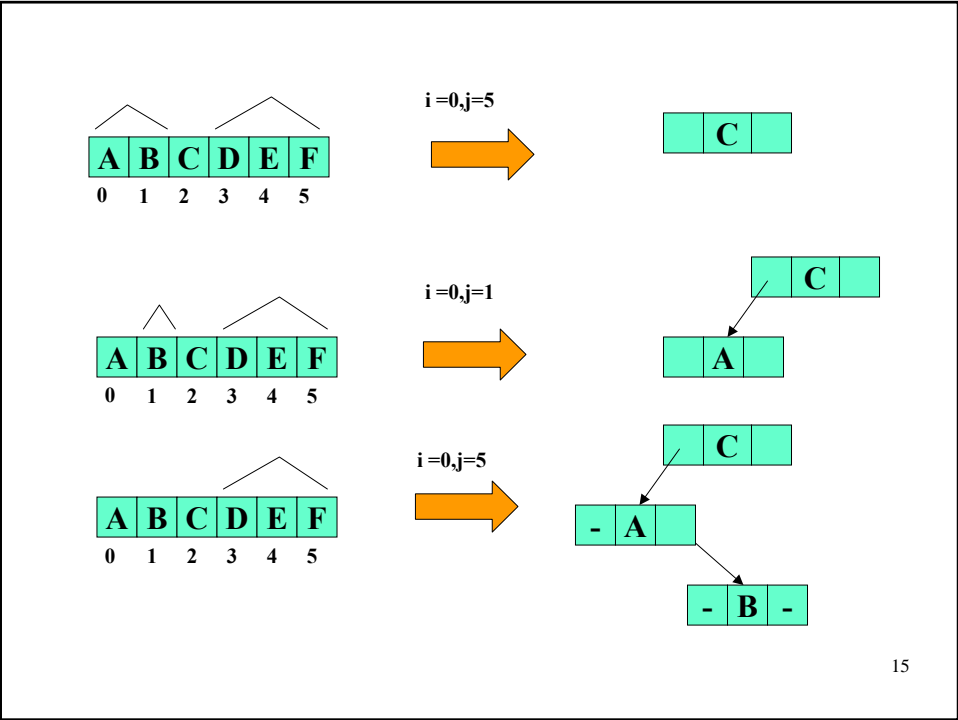
13

```

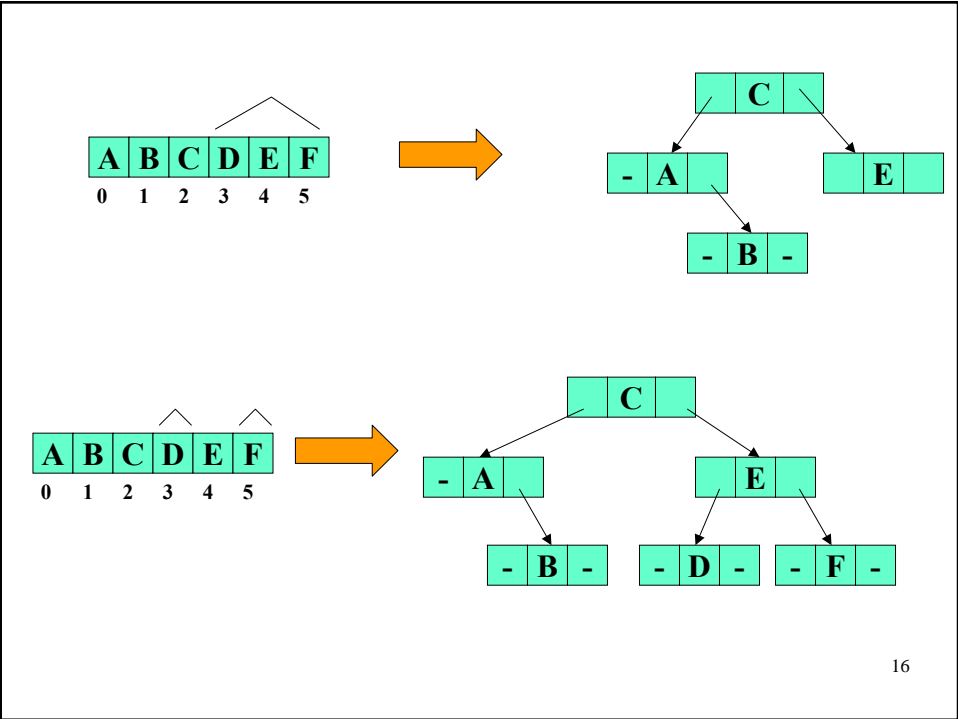
TreePtr AlbBinBil(int * vett, int i, int j)
/*costruisce un albero n-bilanciato a partire da un vettore di indici i e j
* prec: vett!=NULL && i≥0 && j≥0
*postc: t punta alla radice di un albero n-bilanciato che contiene vett[i],...,vett[j]*/
{int m;
TreePtr t;
if (i <= j)
{m = (i+j)/2;
t = malloc(sizeof(Node));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

```

14



15



16

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.