ALGORITMI E STRUTTURE DATI

Prof. Manuela Montangero

A.A. 2022/23

STRUTTURE DATI: alberi

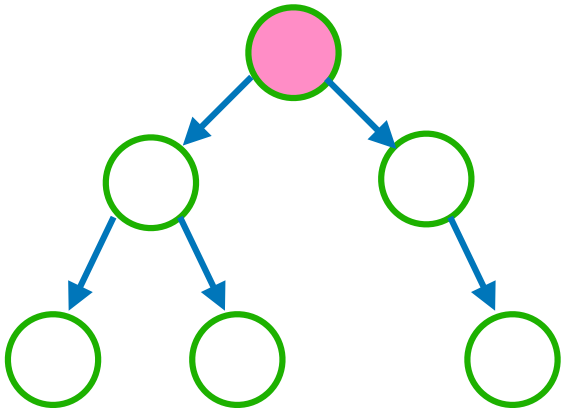
UNIMORE
UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

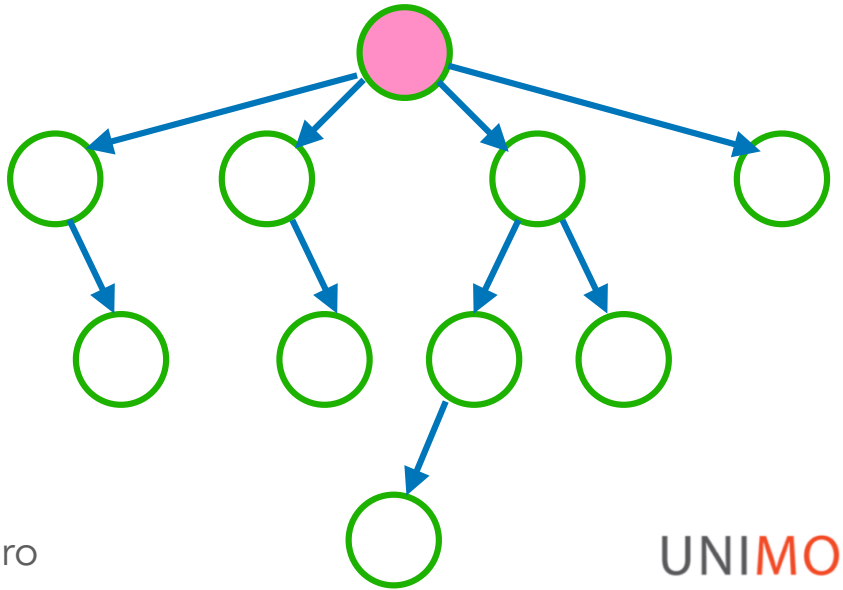
Albero Radicato (Rooted Tree)

DEFINIZIONE:

Un **albero radicato** $T = (V, E)$ è una coppia di insiemi con le seguenti proprietà :

- V è un insieme di **nodi**
- E è un insieme di **archi orientati** che connette coppie di nodi di V
- Un nodo di V è la **radice** r dell'albero
- Ogni nodo, a parte la radice r, ha esattamente un arco entrante
- Esiste un cammino unico dalla radice ad ogni nodo



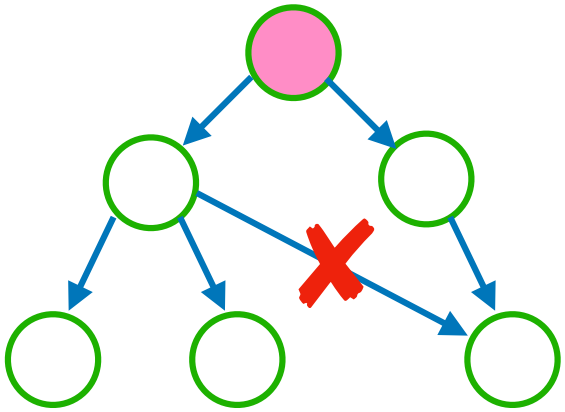


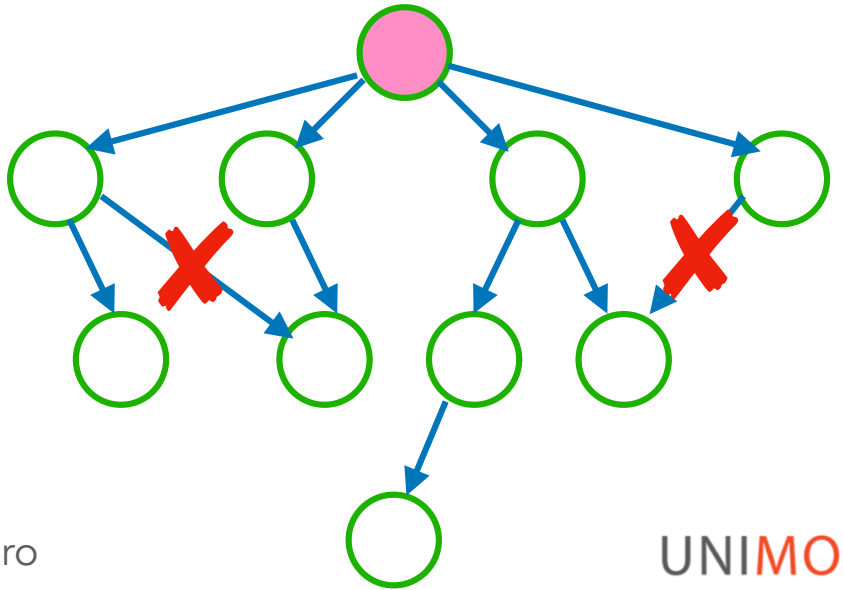
Albero Radicato (Rooted Tree)

DEFINIZIONE:

Un **albero radicato** $T = (V, E)$ è una coppia di insiemi con le seguenti proprietà :

- V è un insieme di **nodi**
- E è un insieme di **archi orientati** che connette coppie di nodi di V
- Un nodo di V è la **radice** r dell'albero
- Ogni nodo, a parte la radice r, ha esattamente un arco entrante
- Esiste un cammino unico dalla radice ad ogni nodo





Albero Radicato (Rooted Tree)

DEFINIZIONE RICORSIVA:

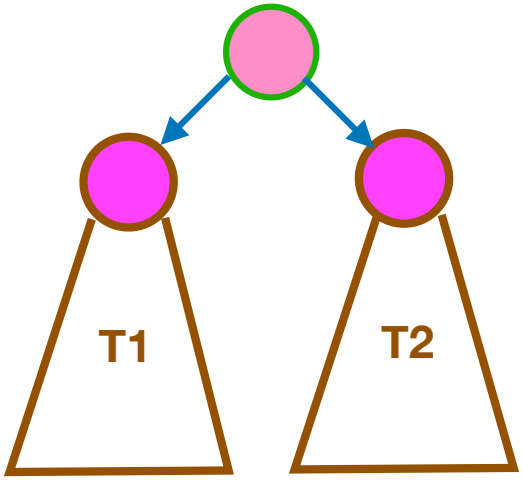
Un **albero radicato** è dato da:

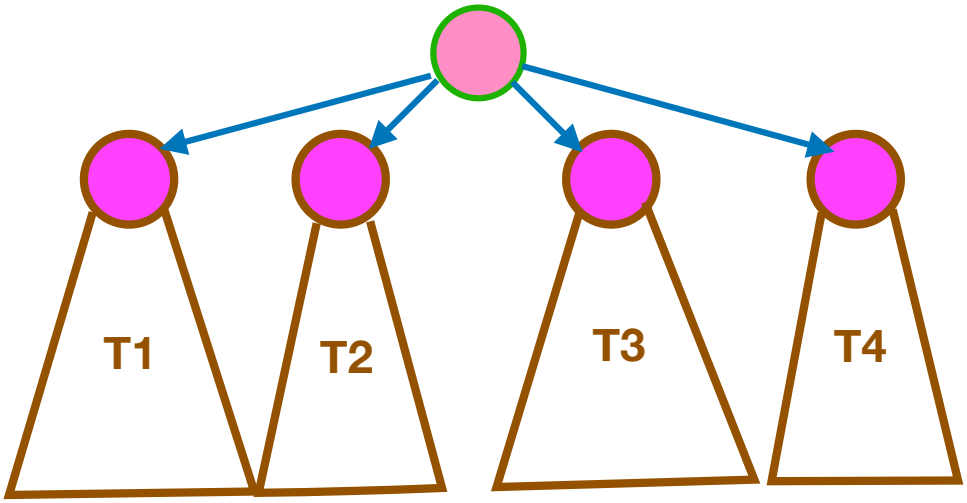
- un insieme **vuoto**

OPPURE

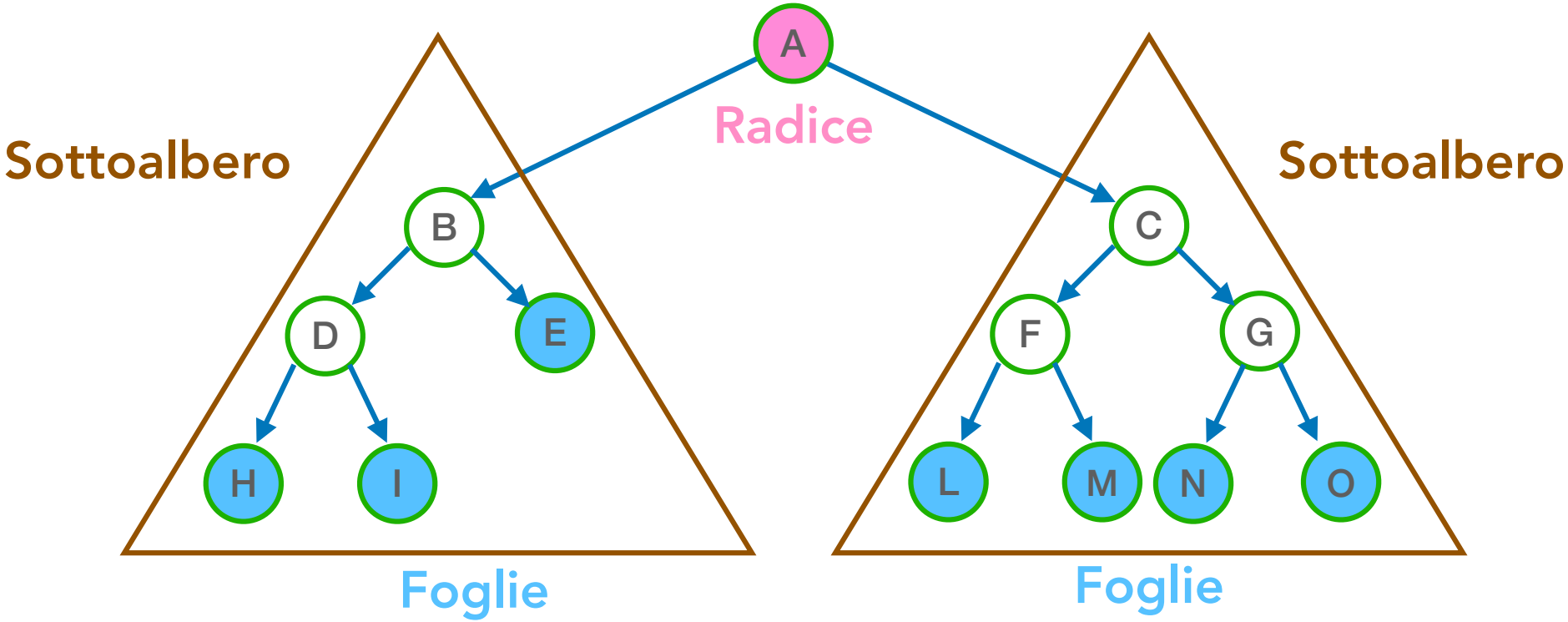
- un nodo **radice** e uno o più **sottoalberi**:
  - ogni sottoalbero è un albero
  - la **radice** e' connessa alla **radice** di ogni sottoalbero da un **arco orientato**

E' perfetto per scrivere algoritmi ricorsivi!!



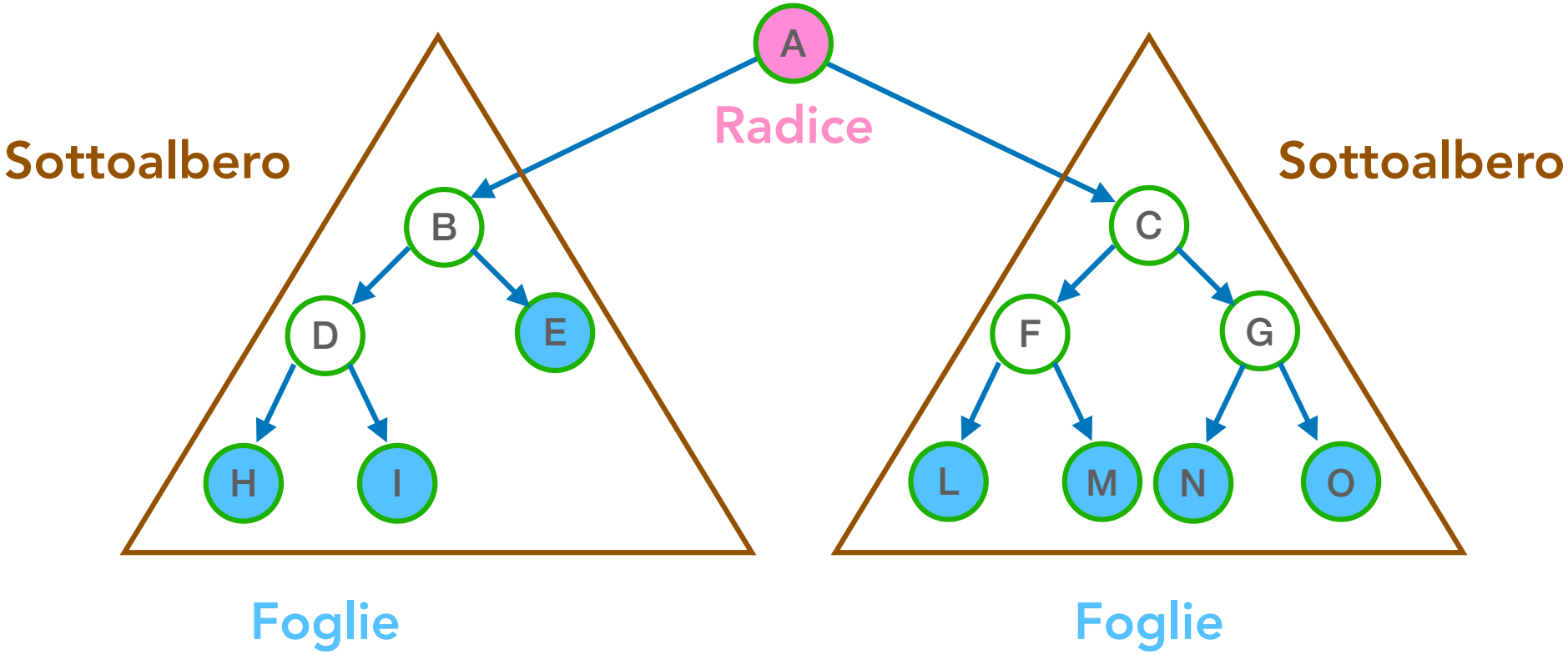


Alberi: nomenclatura (terminologia)



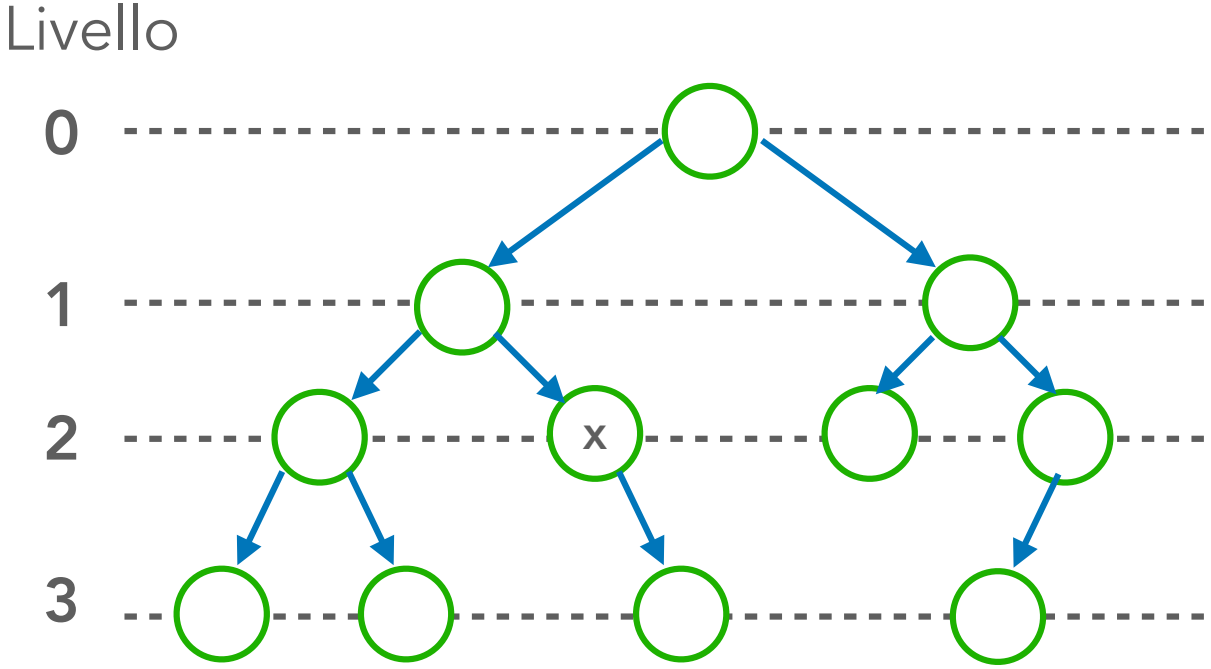
- A è la **radice** (root)
- A è il **padre** (father) di B e C
- B e C sono **figli** (children) di A
- B e C sono **fratelli** (siblings)
- B è un **antenato** (ancestor) di H
- H è un **discendente** (descendent) di B
- B è radice del **sottoalbero** di sx
- H, I, E, L, M, N e O sono **foglie** (leaves), non hanno figli.
- A,B,D,C,F e G sono **nodi interni** (internal nodes), hanno almeno un figlio.

Alberi: nomenclatura



- **father**(x) è il nodo parde di x —> father(D) = B
- **children**(x) è l'insieme di nodi figli di x
  —> children(F) = {L,M}
  —> children(H) = {}
- **sibling**(x) è l'insieme dei fratelli di x —> sibling(D) = {E}

Alberi: nomenclatura



La radice ha profondità zero

Il nodo x ha profondità 2
Il nodo x sta a livello 2

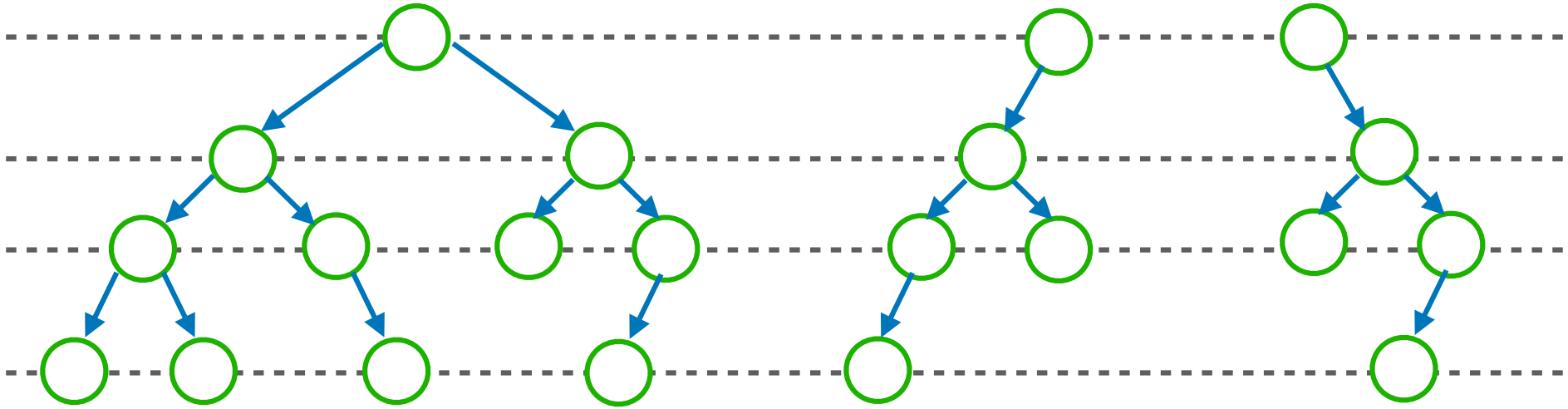
L'albero ha altezza 3

- **PROFONDITÀ** (**Depth**) o **LIVELLO** (**Level**) di un nodo:
  lunghezza del cammino semplice dalla radice al nodo
  (lunghezza misurata come numero di archi)
- **ALTEZZA** (**Height**) dell'albero:
  profondità massima delle foglie

Alberi binari

DEFINIZIONE:

Un **albero binario** è un albero in cui ogni nodo ha **zero**, **uno** o **due figli** (identificati come **figlio sinistro** e **figlio destro**).



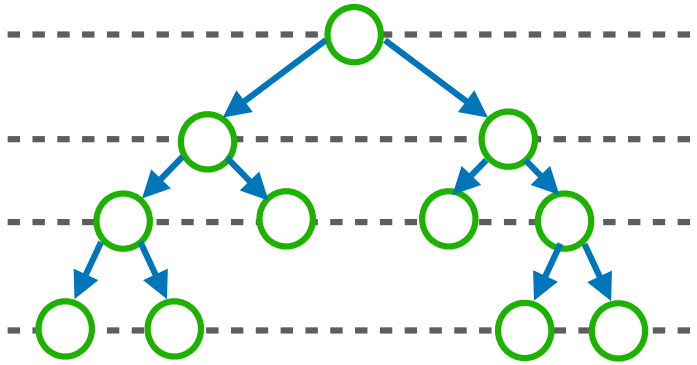
OSSERVAZIONE:

Il figlio sinistro e il figlio destro sono definiti dalla struttura dell'albero e non possono essere "scambiati"

Alberi binari

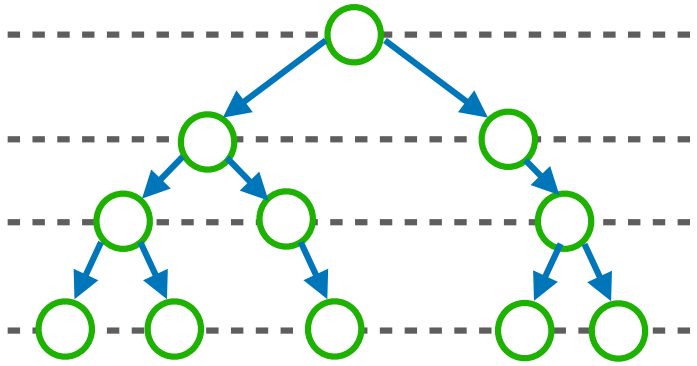
DEFINIZIONE:

Un albero binario è **COMPLETO** se ogni nodo ha zero figli oppure due figli.



DEFINIZIONE:

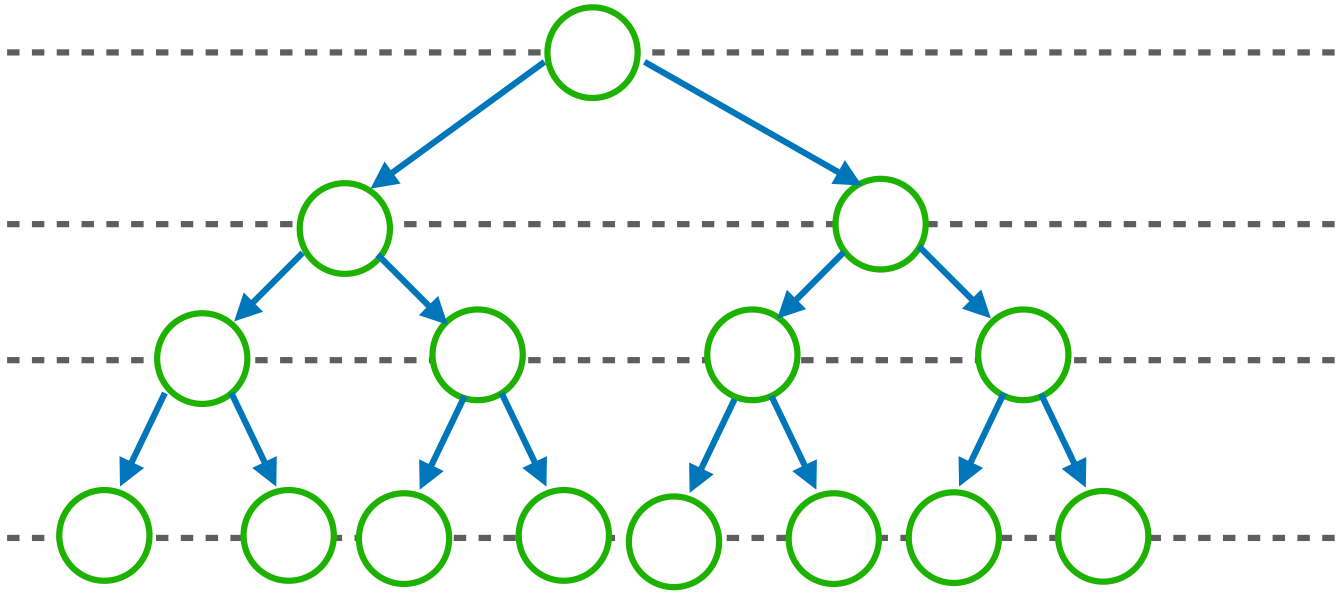
Un albero binario è **PERFETTAMENTE BILANCIATO** se le foglie sono tutte allo stesso livello.



Alberi binari

Un albero binario può essere

COMPLETO e **PERFETTAMENTE BILANCIATO**



Alberi binari

PROPOSIZIONE:

Un **albero binario completo e perfettamente bilanciato** di altezza $h \geq 0$ ha esattamente $n = 2^{h+1} - 1$ nodi di cui 2^h foglie.

DIMOSTRAZIONE per INDUZIONE (per il **numero di foglie**):

CASO BASE: Un albero di altezza zero ha $h = 0$ e un solo nodo (la radice) —> $1 = 2^0 = 1$

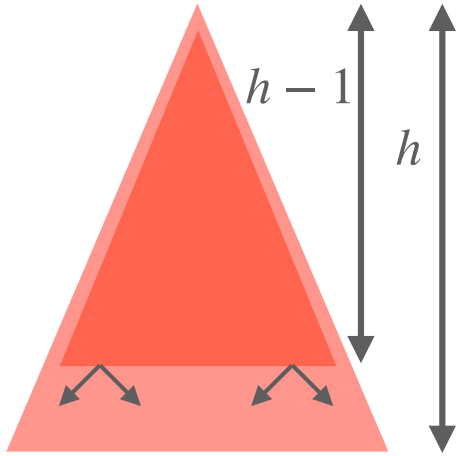
IPTESI INDUTTIVA: Un albero di altezza $h - 1 \geq 1$ ha 2^{h-1} foglie

PASSO INDUTTIVO:

Dimostriamo che: un albero di altezza h ha 2^h foglie

Dimostrazione:

- L'albero di altezza h contiene un albero di altezza $h - 1$: le sue foglie sono i nodi del penultimo livello dell'albero
- Ogni nodo del penultimo livello ha esattamente due figli, che sono le foglie dell'albero
- Il numero di foglie e' $2^{h-1} \cdot 2 = 2^h$



Alberi binari

PROPOSIZIONE:

Un **albero binario completo e perfettamente bilanciato** di altezza $h \geq 0$ ha esattamente $n = 2^{h+1} - 1$ nodi di cui 2^h foglie.

DIMOSTRAZIONE (per il numero di nodi):

Per ogni $i = 0, \ldots, h$, il numero di nodi a livello i e' uguale al numero di foglie di un albero binario completo perfettamente bilanciato di altezza i. Quindi, il numero totale di nodi e'

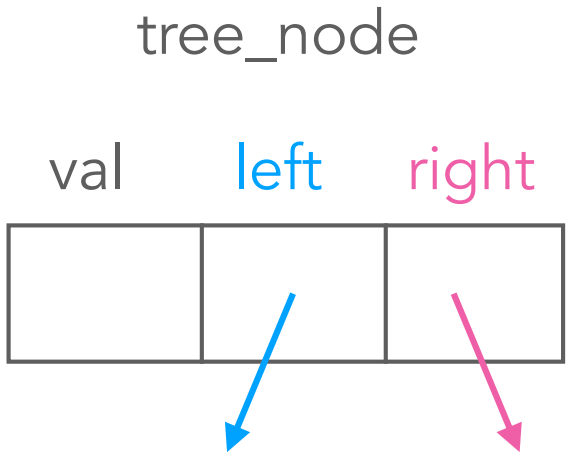
$$\sum_{i=0}^{h} 2^i = 2^{h+1} - 1$$

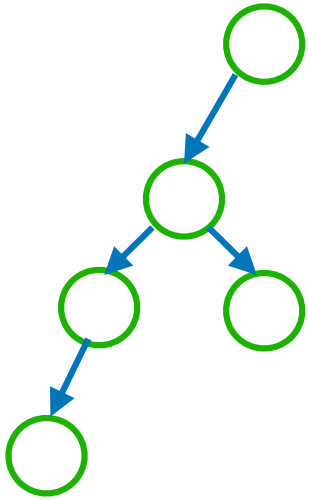
COROLLARIO:

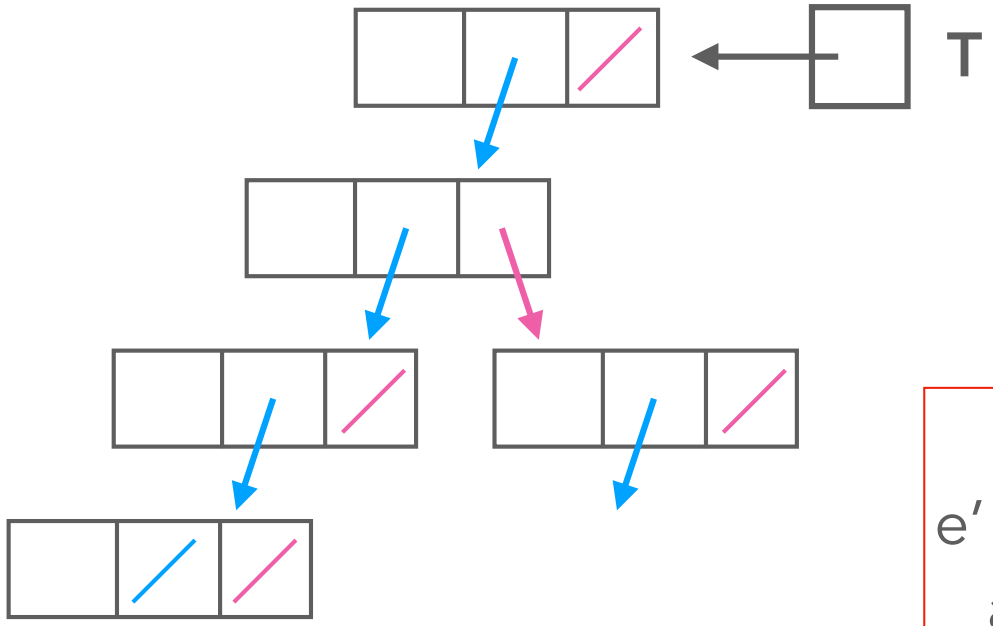
Un albero binario completo e perfettamente bilanciato di n nodi ha altezza $h = \log(n+1) - 1 \in \Theta(\log n)$

Alberi binari: realizzazione

NODO di un albero binario (1):
- campo **LEFT**: puntatore al figlio sinistro
- campo **RIGHT**: puntatore al figlio destro
- campo **VAL**: eventuale valore associato al nodo



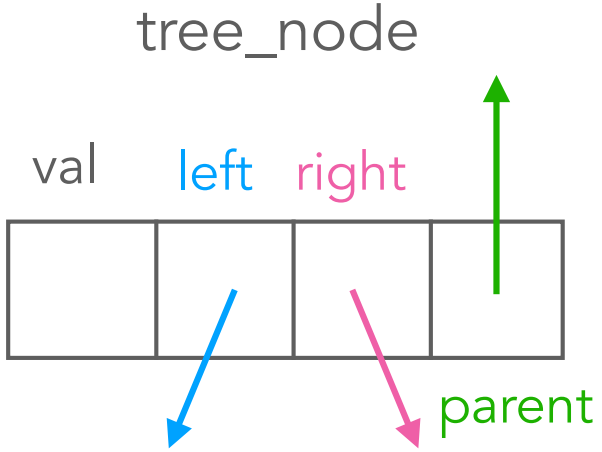


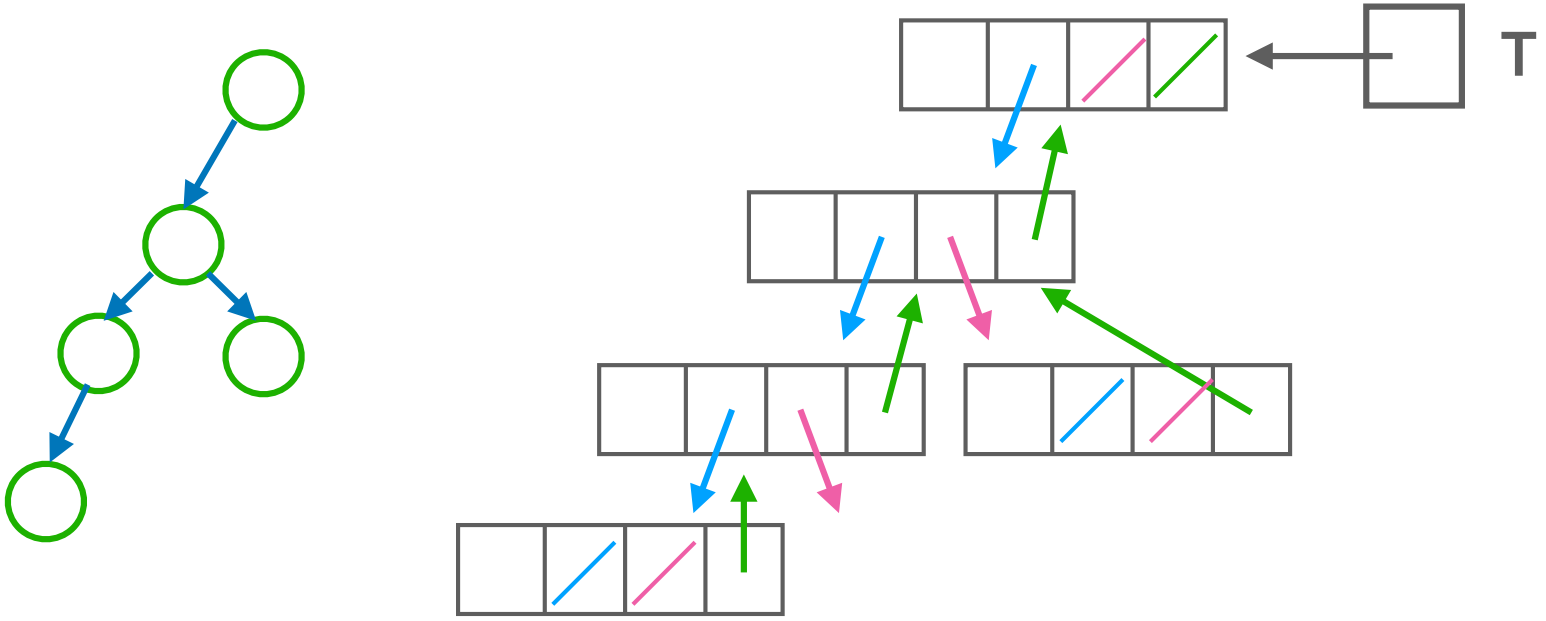


L'albero **T** e' il puntatore alla radice

Alberi binari: realizzazione

NODO di un albero binario (2):
- campo **LEFT**: puntatore al figlio sinistro
- campo **RIGHT**: puntatore al figlio destro
- campo **PARENT**: puntatore al nodo padre
- campo **VAL**: eventuale valore associato al nodo





L'albero **T** e' il puntatore alla radice

Esempio: come usare la ricorsione

PROBLEMA: contare il numero di nodi di un albero binario T

- **caso base**: albero vuoto. Il numero di nodi è pari a zero
- **caso ricorsivo**: albero non vuoto. Il numero di nodi è uguale al numero di nodi nel sottoalbero sinistro, più il numero di nodi nel sottoalbero destro, più uno (la radice)

Scriviamo una funzione che prende in input il puntatore ad un nodo radice di un albero e restituisce il numero di nodi dell'albero radicato in quel nodo

```
contaNodi(t)
if t = NIL
 then return 0
 else return 1 + contaNodi(t.right) + contaNodi(t.left)
```

chiamata principale: contaNodi(T)

Esempio: come usare la ricorsione

PROBLEMA: contare il numero di nodi di un albero binario T

- **caso base**: l'albero è una foglia. Il numero di nodi è pari a uno
- **caso ricorsivo**: albero non vuoto. Il numero di nodi è uguale al numero di nodi nel sottoalbero sinistro (se esiste), più il numero di nodi nel sottoalbero destro (se esiste), più uno (la radice)

Scriviamo una funzione che prende in input il puntatore ad un nodo radice di un albero e restituisce il numero di nodi dell'albero radicato in quel nodo

```
contaNodi(t)
if t.right = t.left = NIL
 then
  return 1
 else
  n_s:= 0, n_d:= 0
  if t.left NOT= NIL then n_s:= contaNodi(t.left)
  if t.right NOT= NIL then n_d:= contaNodi(t.right)
  return 1 + n_s + n_d
```

chiamata principale: contaNodi(T)

ESERCIZI

1. Fare un esempio di un albero binario con almeno 10 nodi e simulare l'esecuzione delle due funzioni contaNodi, mostrando l'ordine delle chiamate ricorsive, del ritorno dalle chiamate e i valori che vengono restituiti dalle chiamate.
2. Scrivere lo pseudo-codice di una funzione ricorsiva che, dato in input il puntatore al nodo radice di un albero T, restituisce l'altezza dell'albero. Se ne dia una versione in cui il caso base è l'albero vuoto, e una in cui il caso base è la foglia.