# Algoritmi e Strutture di Dati

## Alberi radicati

*m.patrignani*

# Nota di copyright

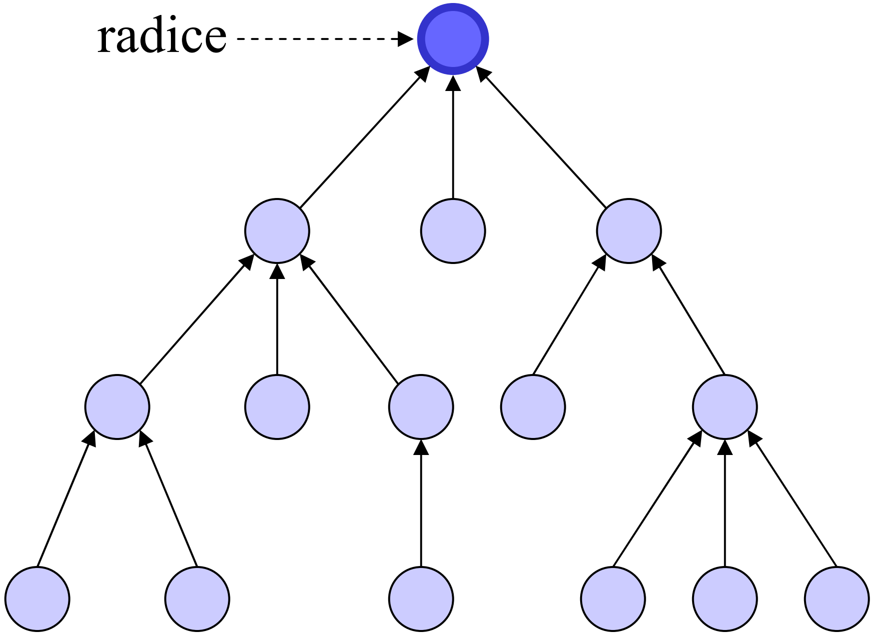
# Sommario

- Alberi radicati
  - definizione e uso
- Strutture di dati per rappresentare alberi
  - alberi binari, alberi di grado arbitrario
- Esercizi sugli alberi

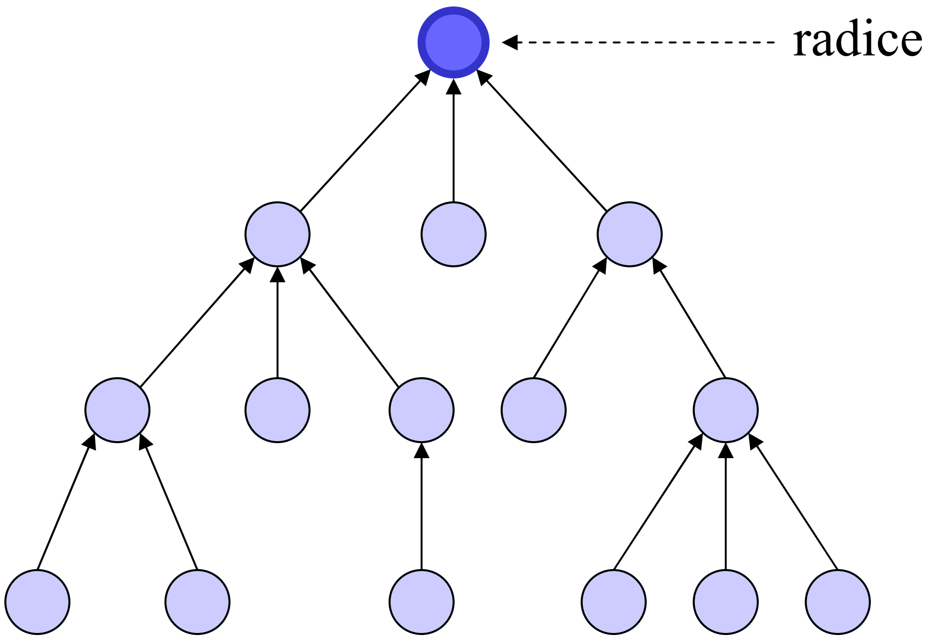
# Definizione di albero radicato (rooted tree)

- Un *albero radicato* è un insieme di nodi, su cui è definita una relazione binaria "$x$ è figlio di $y$" (oppure "$y$ è genitore di $x$") tale che:
  1. ogni nodo ha un solo genitore, con l'eccezione della radice che non ha genitori
  2. c'è un cammino diretto da ogni nodo alla radice
     - l'albero, cioè, è connesso



# Esempio di albero radicato




- Un albero può essere costruito a partire dalla radice aggiungendo ogni volta un nodo *x* come figlio di un nodo *y* già esistente
  - ciò giustifica il fatto che, se l'albero ha *n* nodi, allora ci sono *n*-1 relazioni genitore/figlio

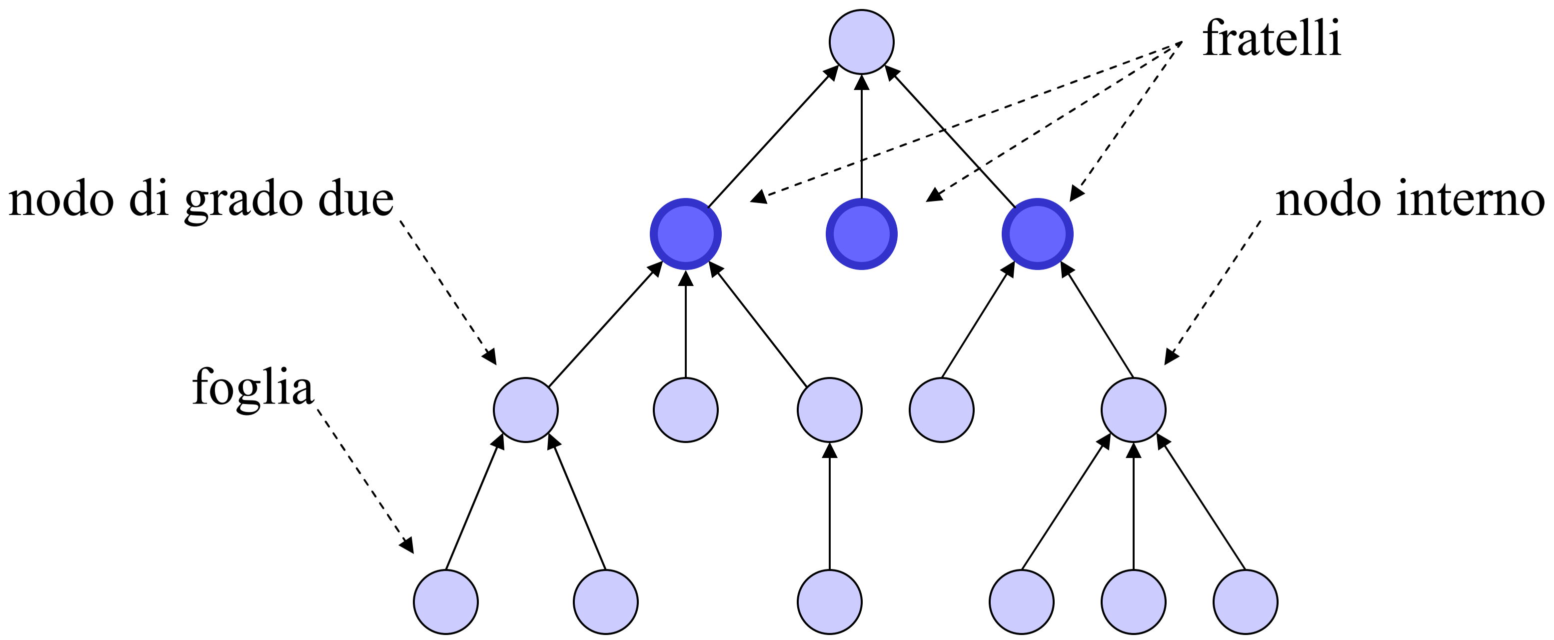
 

# Numerose applicazioni usano alberi

- I rapporti di ereditarietà determinano alberi
  – alberi genealogici o filogenetici
  – ereditarietà di classi nella programmazione ad oggetti
- I rapporti gerarchici sono alberi
  – gerarchie organizzative, di controllo, di responsabilità
- I rapporti di contenimento formano alberi
  – la classificazione scientifica degli organismi (tassonomie)
  – le directory del filesystem
  – i cammini minimi da una sorgente a tutti i nodi di una rete
- La struttura sintattica di una frase è un'albero
  – alberi sintattici
- …

# Alberi: definizioni




- Due nodi che hanno lo stesso genitore si dicono *fratelli*
- Il numero di figli di un nodo è il suo *grado*
- I nodi di grado zero sono *foglie*
- Un nodo non foglia è detto *nodo interno*

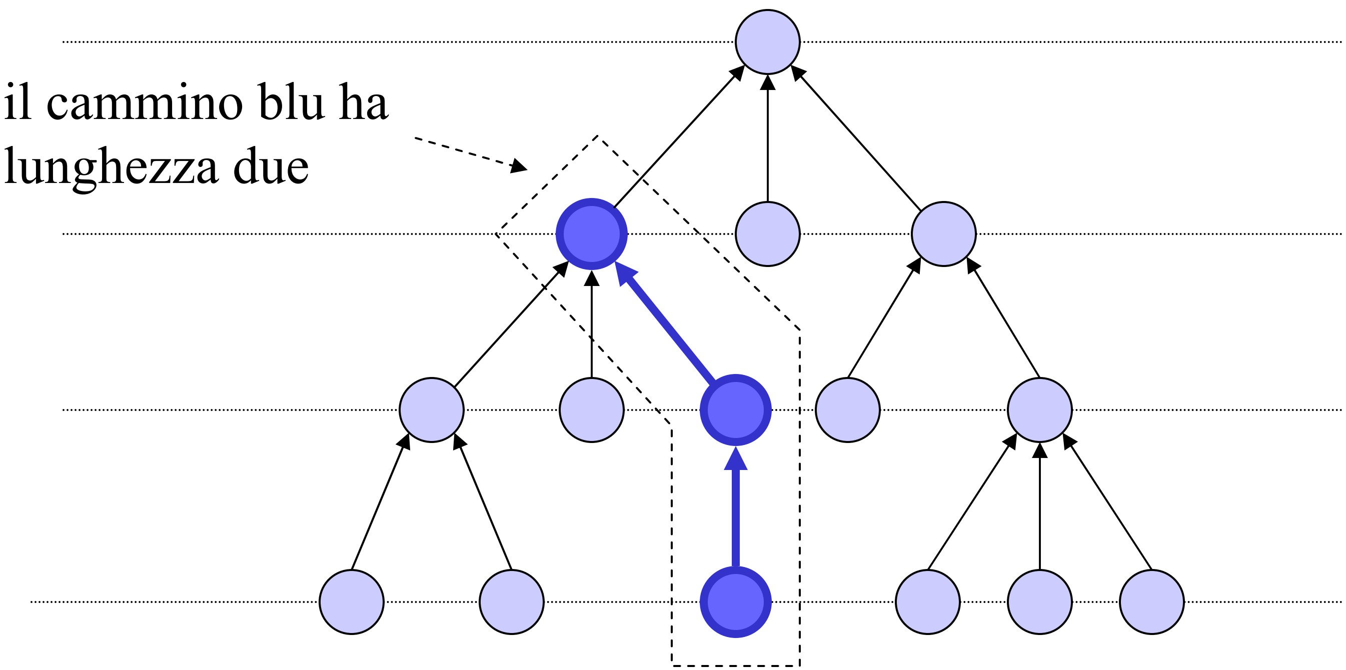
 

# Tipi di alberi

- Alberi binari
  - ogni nodo può avere solamente un figlio sinistro e un figlio destro
    - l'ordine dei figli è generalmente significativo
    - si distingue tra avere il solo figlio sinistro e avere il solo figlio destro
- Alberi di grado arbitrario
  - non è noto a priori il numero massimo dei figli di un nodo
    - l'ordine dei figli generalmente non è significativo

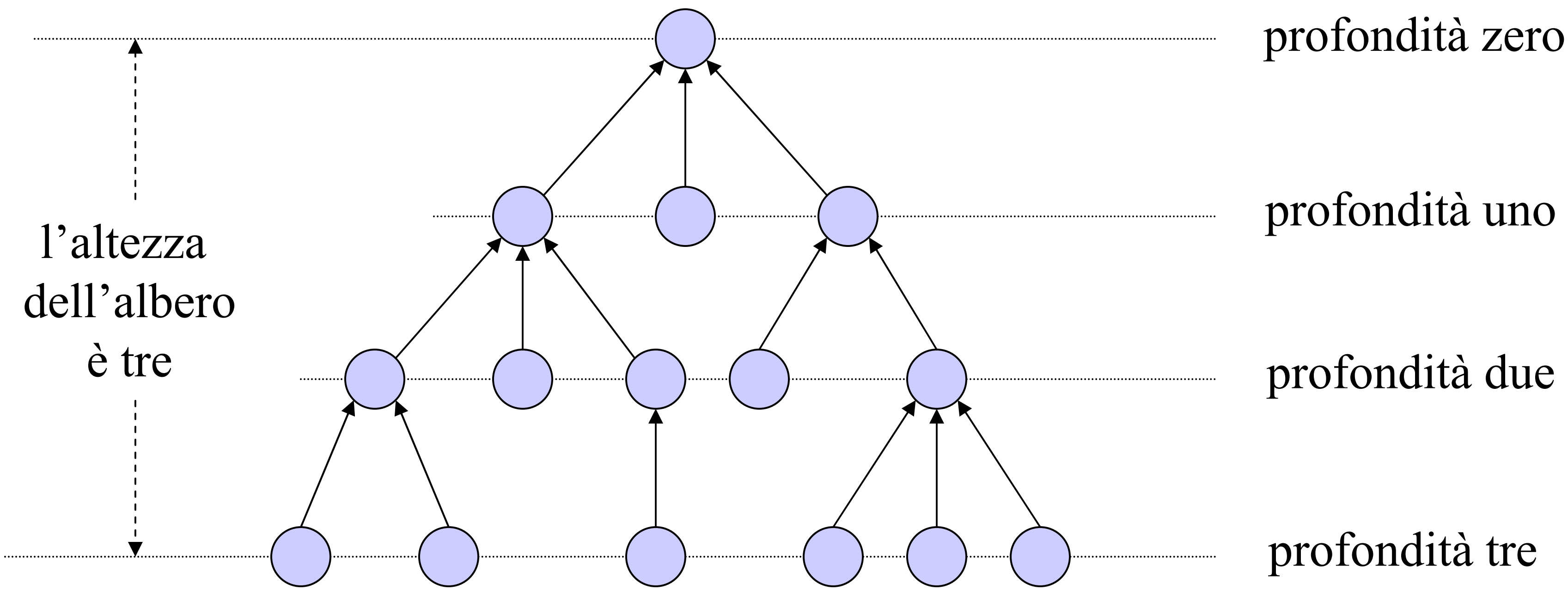
# Alberi: definizioni




- Una sequenza di nodi tali che uno è il genitore del successivo è detta *cammino*
  - il cammino percorre gli archi alla rovescia rispetto alla figura qui sopra
- Il numero degli archi di un cammino è la sua *lunghezza*

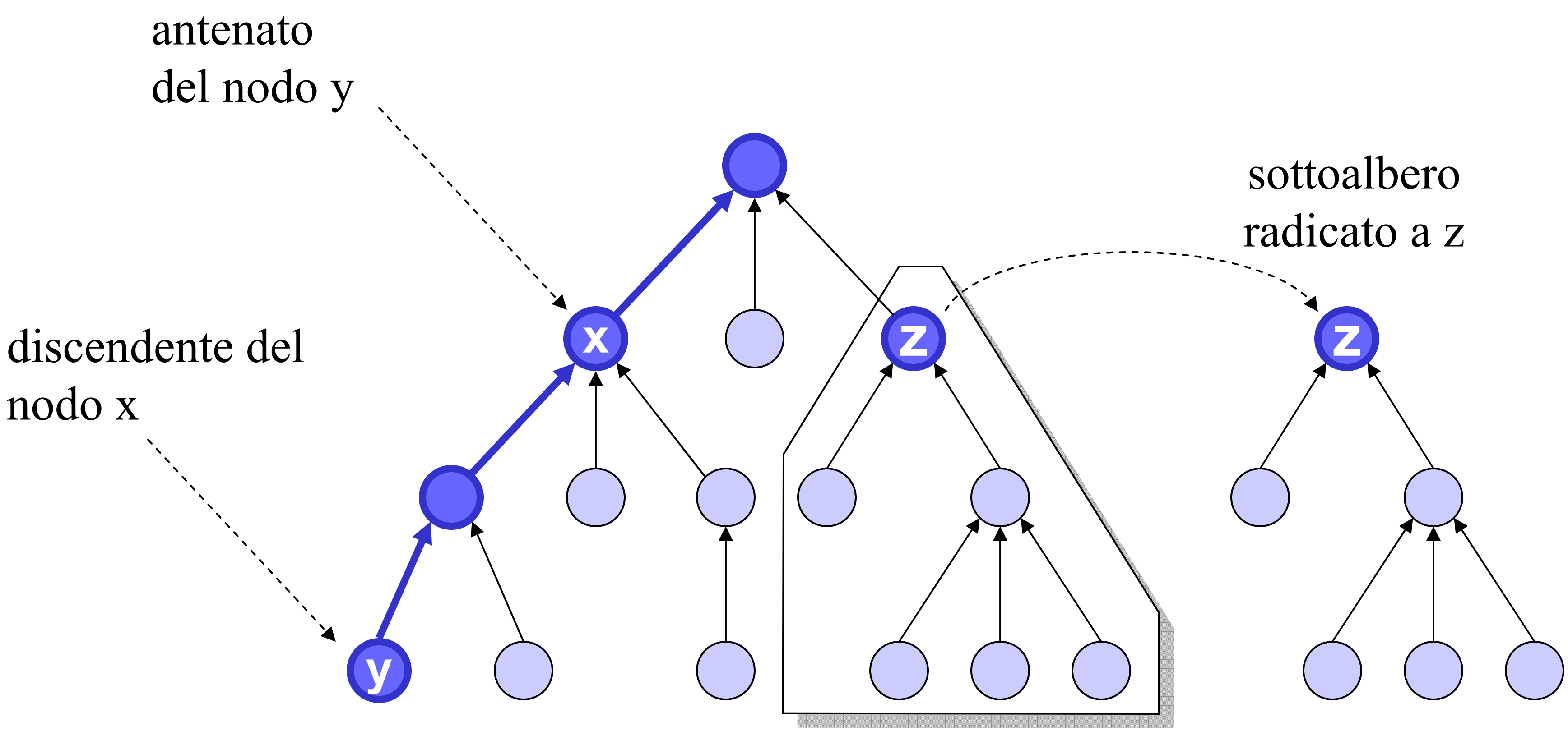
 

# Alberi: definizioni




- La *profondità* di un nodo è la lunghezza del cammino dal nodo alla radice
- La profondità del nodo più profondo è *l'altezza* dell'albero

# Alberi: definizioni

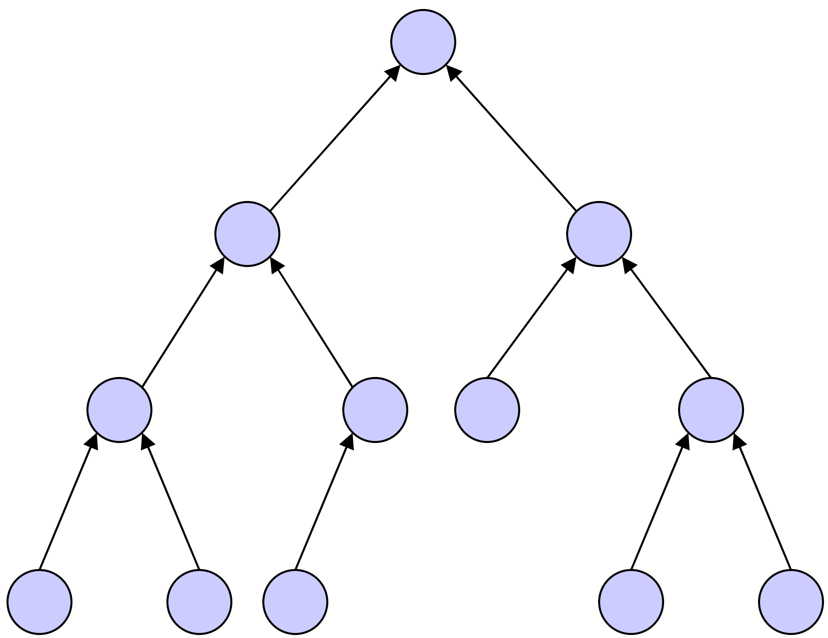



- Qualunque nodo *x* sul cammino (unico) dalla *y* alla radice è un *antenato* di *y*, mentre *y* è un *discendente* di *x*;
- L'insieme costituito da un nodo *z* e da tutti i suoi discendenti è il *sottoalbero radicato a z*

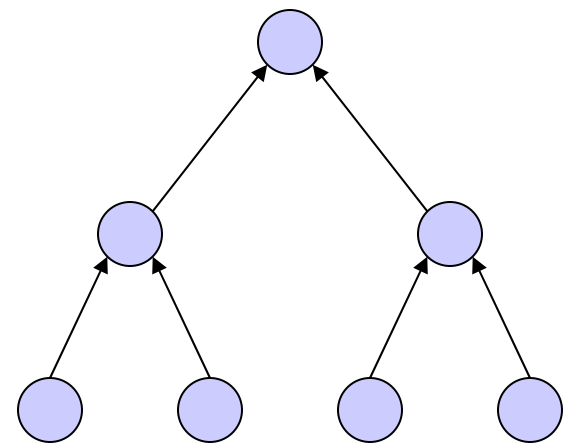
 

# Alberi: definizioni

albero binario



albero binario completo



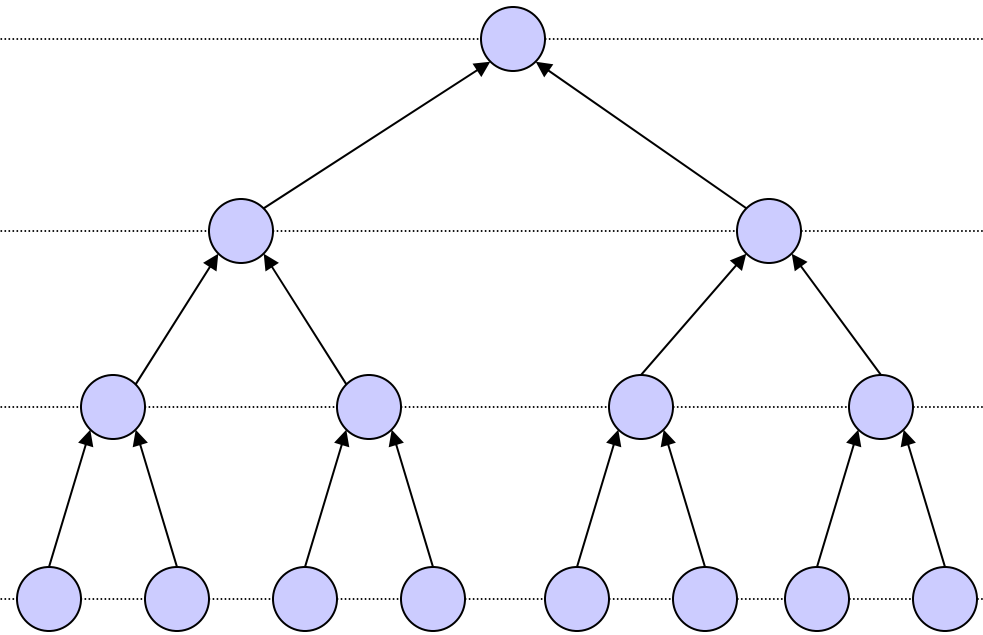
- Un albero *ordinato* è un albero per il quale l'ordine dei figli di ogni nodo è significativo (non possono essere permutati)
- Un albero *binario* è un albero ordinato in cui i nodi hanno grado al più due
- Un albero binario è *completo* se ogni livello presenta tutti i nodi possibili

# Alberi: definizioni

| profondità | nodi | nodi totali |
|---|---|---|
| 0 | 1 | 1 |
| 1 | 2 | 3 |
| 2 | 4 | 7 |
| 3 | 8 | 15 |

albero binario completo



- Un albero binario completo di altezza h
  - ha $2^h$ foglie, dunque $h = \log_2$(numero foglie)
  - ha $2^h$-1 nodi interni
  - ha $2^{h+1}$-1 nodi

# Il tipo astratto albero

- Tipo astratto albero di interi
  - domini
    - il dominio di interesse è l’insieme degli alberi di interi
    - dominio di supporto: i riferimenti R che identificano le posizioni nell’albero
    - dominio di supporto: gli interi Z = {0, 1, -1, 2, -2, …}
    - dominio di supporto: i booleani B = {true, false}
  - costanti
    - l’albero vuoto

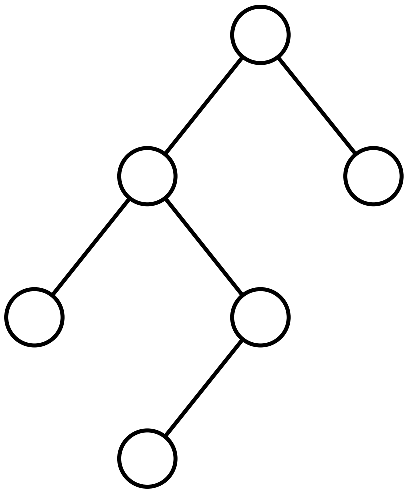
# Operazioni del tipo astratto albero

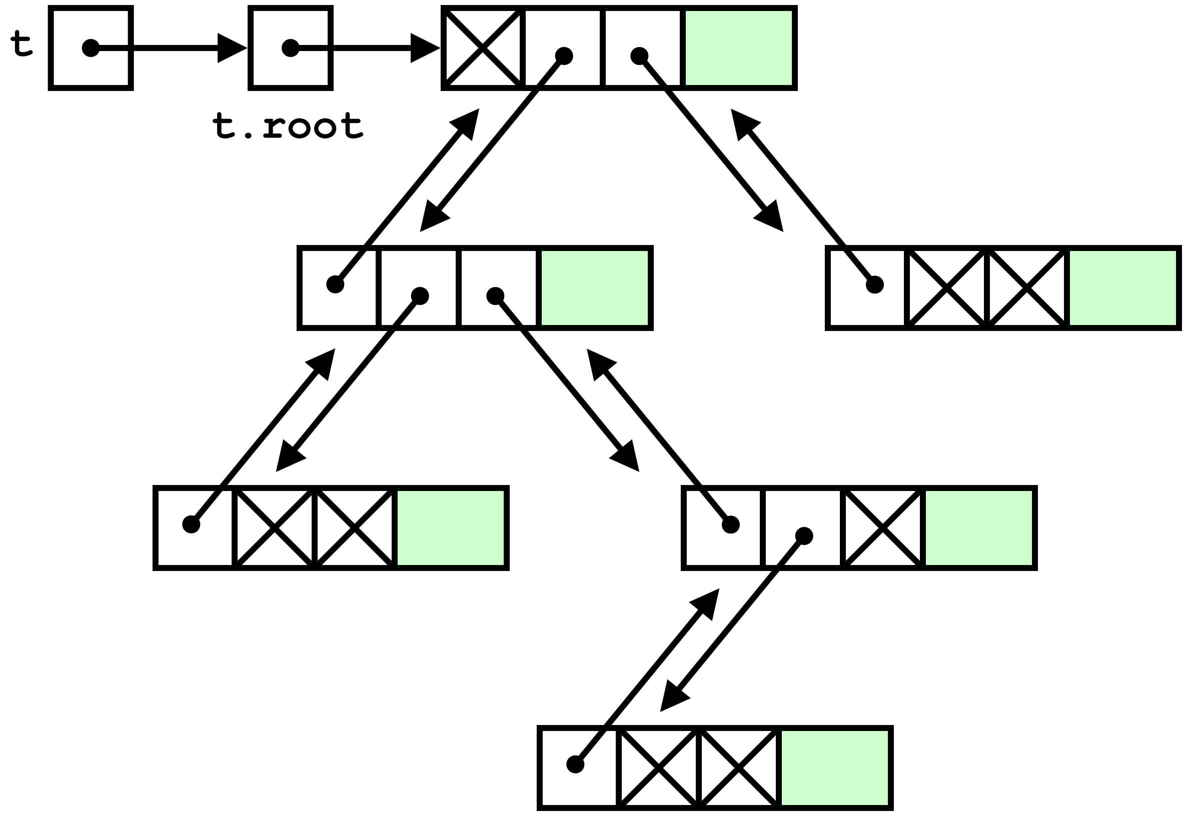
- Operazioni sugli alberi di interi
  - ritorna il riferimento alla radice: ROOT: $T \rightarrow R$
  - ritorna il riferimento al figlio sinistro: LEFT: $T \times R \rightarrow R$
  - ritorna il riferimento al figlio destro: RIGHT: $T \times R \rightarrow R$
  - ritorna l'intero nel nodo specificato: INFO: $T \times R \rightarrow Z$
  - verifica se un albero è vuoto: IS_EMPTY: $T \rightarrow B$
  - aggiunge un nodo come radice: ADD_ROOT: $T \times Z \rightarrow T$
  - aggiunge un nodo come figlio sinistro: ADD_LEFT: $T \times R \times Z \rightarrow T$
  - aggiunge un nodo come figlio destro: ADD_RIGHT: $T \times R \times Z \rightarrow T$
  - elimina una foglia: DELETE_LEAF: $L \times R \rightarrow L$
  - cerca un nodo: SEARCH: $T \times Z \rightarrow R$
  - svuota l'albero: EMPTY: $T \rightarrow T$
  - conta i nodi dell'albero: SIZE: $T \rightarrow Z$
  - …

# Rappresentazione di alberi binari

- Analogamente alle liste, gli alberi binari possono essere rappresentati mediante oggetti e riferimenti

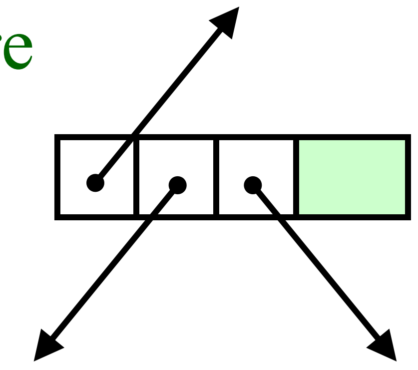




# Rappresentazione di alberi binari

- Un nodo dell'albero binario è un oggetto con i quattro campi
  - `parent`: riferimento al nodo genitore
  - `left`: riferimento al figlio sinistro
  - `right`: riferimento al figlio destro
  - `info`: dati satellite



- Un albero binario è un oggetto con un solo campo `root` che è un riferimento al nodo radice

# Operazioni sugli alberi binari

- `NEW_TREE()`
  - restituisce una struttura rappresentante l'albero vuoto
  - questa funzione rappresenta la costante
- `IS_EMPTY(t)`
  - restituisce TRUE se l'albero è vuoto
- `ROOT(t)`
  - restituisce il riferimento alla radice dell'albero (`NULL` se t è vuoto)
- `LEFT(t,n)`
  - restituisce il riferimento (può essere `NULL`) al figlio sinistro del nodo *n*
- `RIGHT(t,n)`
  - restituisce il riferimento (può essere `NULL`) al figlio destro del nodo *n*
- `INFO(t,n)`
  - restituisce le informazioni (dati satellite) memorizzate nel nodo *n*
- …

# Esercizi sugli alberi binari

1. Scrivi lo pseudocodice delle funzioni
   `NEW_TREE()`
   `IS_EMPTY(t)`
   `ROOT(t)`
   `LEFT(t,n)`
   `RIGHT(t,n)`
   `INFO(t,n)`
   descritte nella slide precedente
2. Scrivi lo pseudocodice della funzione `TWO_CHILDREN(n)` che ritorna `TRUE` se il nodo `n` ha due figli, `FALSE` altrimenti

# Esercizi sugli alberi binari

3. Scrivi lo pseudocodice della procedura `ADD_ROOT(t,z)` che aggiunga il nodo radice con valore `z` all'albero binario `t`
   – assumi che `t` sia vuoto
4. Scrivi lo pseudocodice delle procedure `ADD_LEFT(t,n,z)` e `ADD_RIGHT(t,n,z)` che aggiungono il figlio sinistro e destro al nodo `n`, contenente il valore `z`
5. Scrivi lo pseudocodice della funzione `ONLY_LEFT(t)` che restituisce `TRUE` se tutti i nodi dell'albero binario `t` hanno solamente il figlio sinistro (o nessun figlio), `FALSE` altrimenti
   • se l'albero è vuoto restituisci `TRUE`

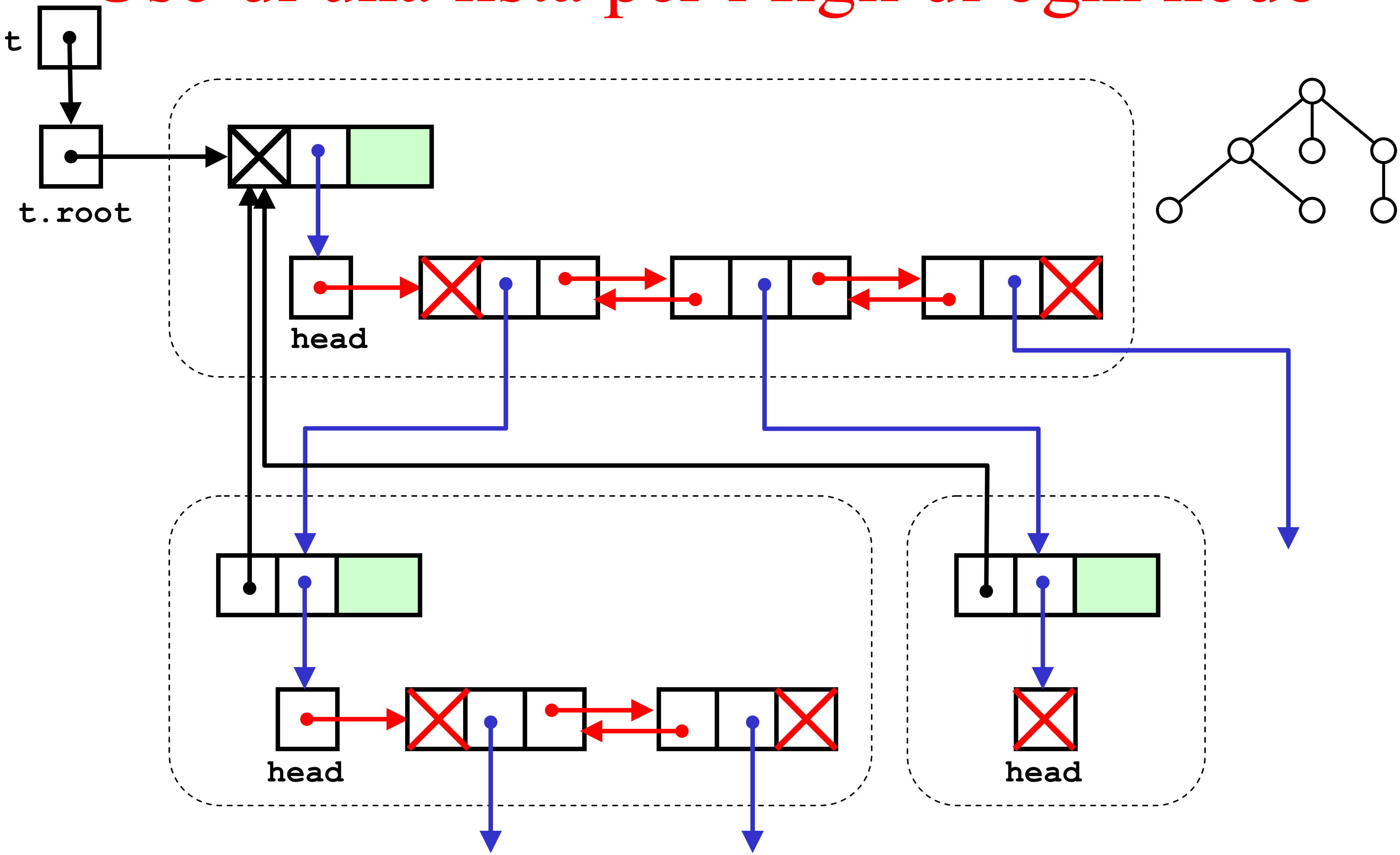
 

# Rappresentazione di alberi di grado arbitrario

- Per rappresentare alberi di grado arbitrario si possono utilizzare diverse strategie
  - uso di una lista per i figli di ogni nodo
    - poco usato perché molto prolisso
  - uso di una struttura detta “figlio-sinistro-fratello-destro”
    - più sintetico

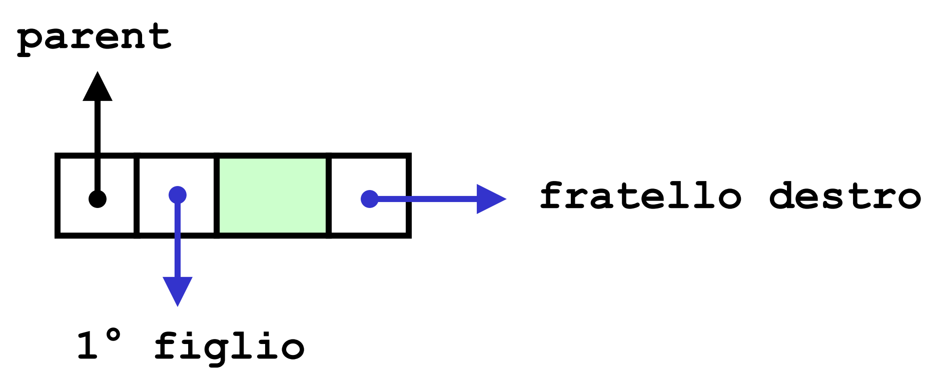
# Uso di una lista per i figli di ogni nodo



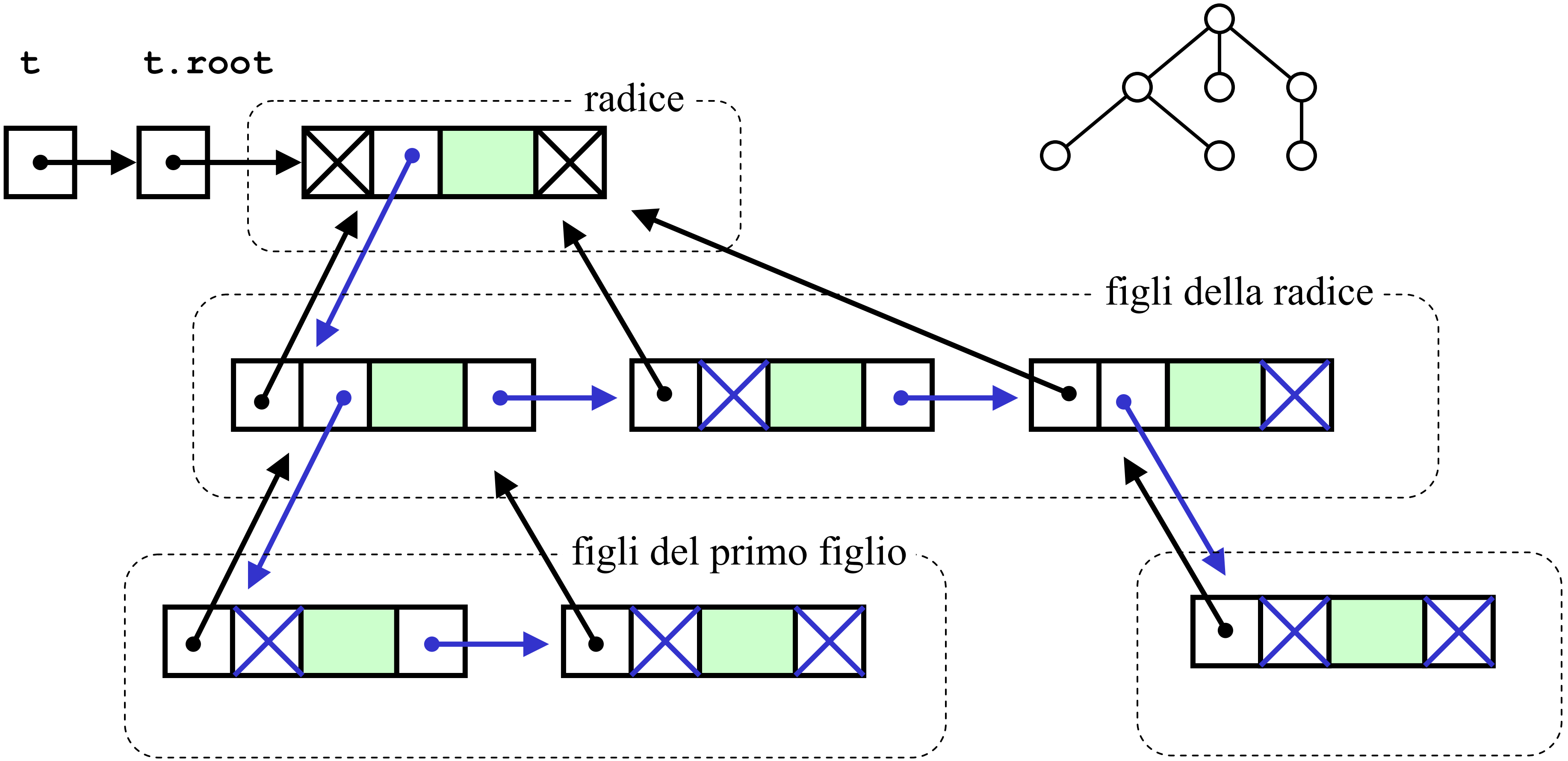
# Struttura “figlio-sinistro-fratello-destro”

- I nodi hanno gli usuali campi `parent`, `left`, `right` e `info`
  - i campi `parent` e `info` hanno il significato usuale
  - il campo `left` è un riferimento al figlio di sinistra (cioè al primo figlio)
  - il campo `right`, invece di essere un riferimento al figlio destro, è un riferimento al prossimo fratello



# Figlio-sinistro-fratello-destro



# Operazioni sugli alberi qualsiasi

- `NEW_TREE()`
  - restituisce una struttura rappresentante l'albero vuoto
- `IS_EMPTY(t)`
  - restituisce `TRUE` se l'albero è vuoto
- `ROOT(t)`
  - restituisce il riferimento alla radice dell'albero (`NULL` se `t` è vuoto)
- `FIRST_CHILD(t,n)`
  - restituisce il riferimento (può essere `NULL`) al figlio sinistro del nodo `n`
- `NEXT_SIBLING(t,n)`
  - restituisce il riferimento (può essere `NULL`) al fratello destro del nodo `n`
- `INFO(t,n)`
  - restituisce l'intero memorizzato nel nodo `n`
- ...

# Esercizi sugli alberi qualsiasi

6. Scrivi lo pseudocodice della procedura `ADD_ROOT(t,z)` che aggiunga un nodo radice con valore `z` all'albero `t`
   – supponi che l'albero `t` sia vuoto
7. Scrivi lo pseudocodice della procedura `ADD_SIBLING(t,n,z)` che aggiunge al nodo `n` un figlio che contiene il valore `z`