

# Algoritmi e Strutture di Dati

## Alberi radicati

*m.patrignani*

110-alberi-radicali-08 copyright ©2019 maurizio.patrignani@uniroma3.it

## Nota di copyright

- queste slides sono protette dalle leggi sul copyright
- il titolo ed il copyright relativi alle slides (inclusi, ma non limitatamente, immagini, foto, animazioni, video, audio, musica e testo) sono di proprietà degli autori indicati sulla prima pagina
- le slides possono essere riprodotte ed utilizzate liberamente, non a fini di lucro, da università e scuole pubbliche e da istituti pubblici di ricerca
- ogni altro uso o riproduzione è vietata, se non esplicitamente autorizzata per iscritto, a priori, da parte degli autori
- gli autori non si assumono nessuna responsabilità per il contenuto delle slides, che sono comunque soggette a cambiamento
- questa nota di copyright non deve essere mai rimossa e deve essere riportata anche in casi di uso parziale

110-alberi-radicali-08 copyright ©2019 maurizio.patrignani@uniroma3.it

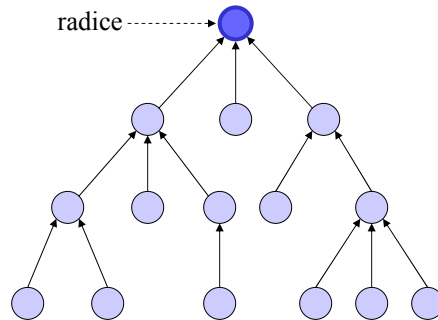
## Sommario

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

110-alberi-radicali-08 copyright ©2019 maurizio.patrignani@uniroma3.it

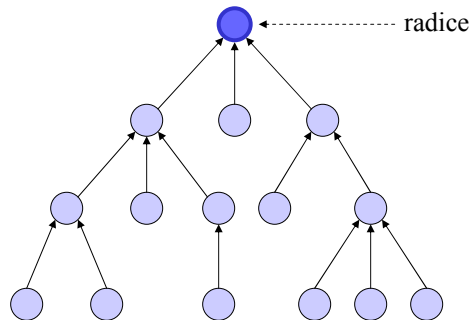
## Definizione di albero radicato (rooted tree)

- 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



110-alberi-radicali-08 copyright ©2019 maurizio.patrignani@uniroma3.it

## 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

110-alberi-radicali-08

copyright ©2019 maurizio.patrignani@uniroma3.it

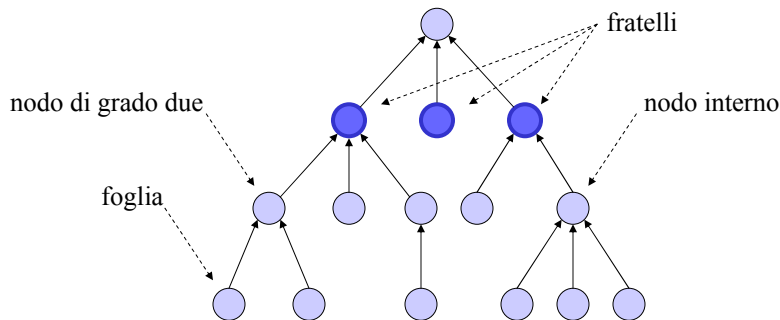
## Numerose applicazioni usano alberi

- 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
- ...

110-alberi-radicali-08

copyright ©2019 maurizio.patrignani@uniroma3.it

## 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*

110-alberi-radicali-08

copyright ©2019 maurizio.patrignani@uniroma3.it

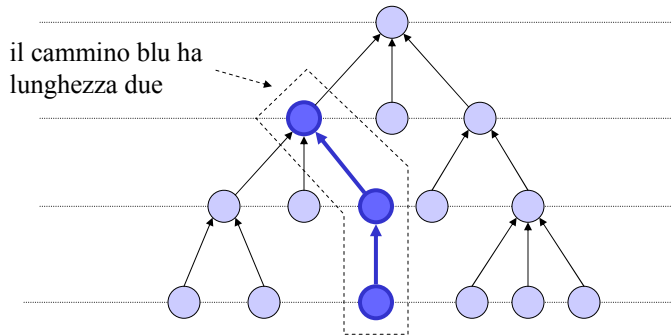
## Tipi di alberi

- 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

110-alberi-radicali-08

copyright ©2019 maurizio.patrignani@uniroma3.it

## Alberi: definizioni

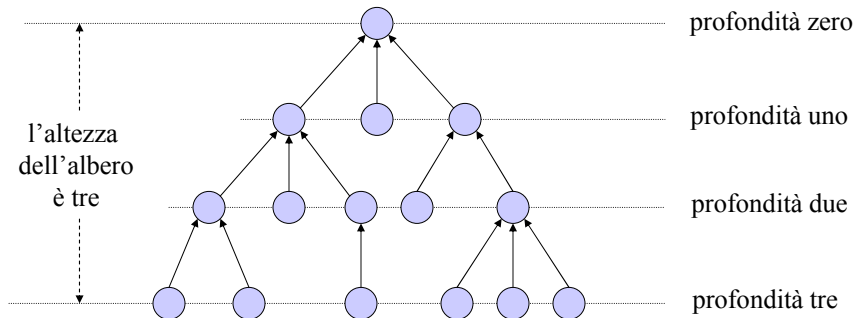


- Una sequenza di nodi tali che uno è il genitore del successivo è detta *cammino*
  - il cammino percorre gli archi alla rovescia rispetto alla figura qui sopra
- Il numero degli archi di un cammino è la sua *lunghezza*

110-alberi-radicali-08

copyright ©2019 maurizio.patignani@uniroma3.it

## Alberi: definizioni

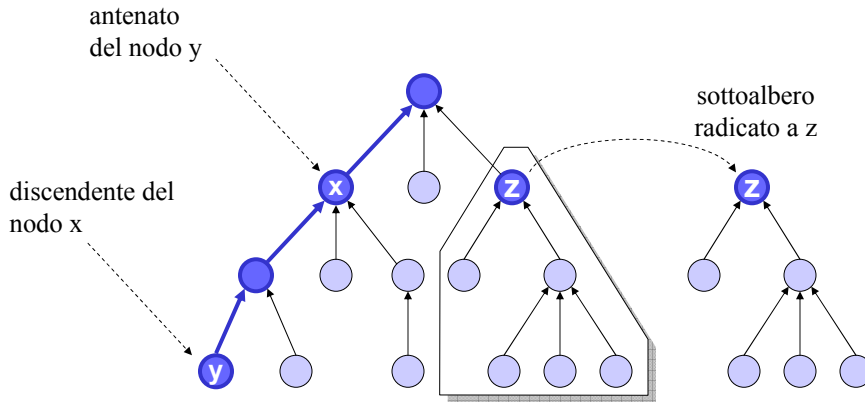


- La *profondità* di un nodo è la lunghezza del cammino dal nodo alla radice
- La profondità del nodo più profondo è *l'altezza* dell'albero

110-alberi-radicali-08

copyright ©2019 maurizio.patignani@uniroma3.it

## Alberi: definizioni

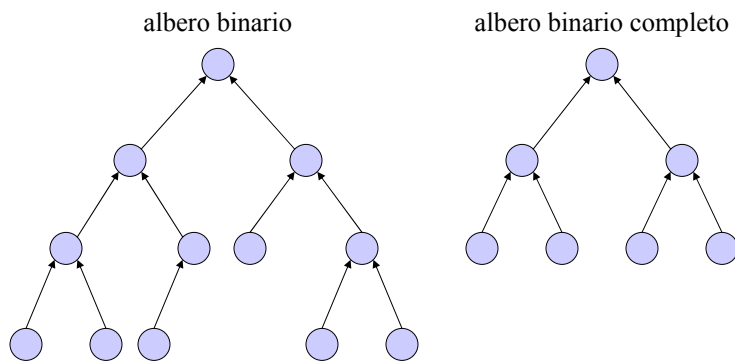


- Qualunque nodo  $x$  sul cammino (unico) dalla  $y$  alla radice è un *antenato* di  $y$ , mentre  $y$  è un *discendente* di  $x$ ;
- L'insieme costituito da un nodo  $z$  e da tutti i suoi discendenti è il *sottoalbero radicato a z*

110-alberi-radicali-08

copyright ©2019 maurizio.patrignani@uniroma3.it

## Alberi: definizioni

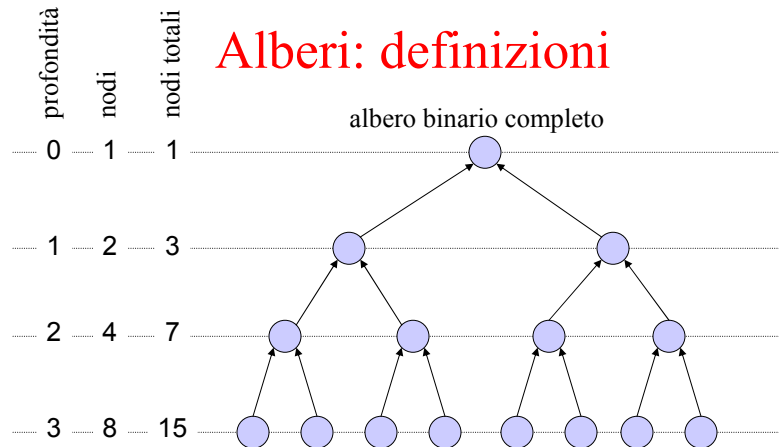


- Un albero *ordinato* è un albero per il quale l'ordine dei figli di ogni nodo è significativo (non possono essere permutati)
- Un albero *binario* è un albero ordinato in cui i nodi hanno grado al più due
- Un albero binario è *completo* se ogni livello presenta tutti i nodi possibili

110-alberi-radicali-08

copyright ©2019 maurizio.patrignani@uniroma3.it

## Alberi: definizioni



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

110-alberi-radicali-08

copyright ©2019 maurizio.patrignani@uniroma3.it

## 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  $Z = \{0, 1, -1, 2, -2, \dots\}$
    - dominio di supporto: i booleani  $B = \{\text{true}, \text{false}\}$
  - costanti
    - l'albero vuoto

110-alberi-radicali-08

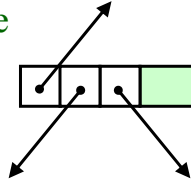
copyright ©2019 maurizio.patrignani@uniroma3.it





## Rappresentazione di alberi binari

- Un nodo dell'albero binario è un oggetto con i quattro campi *struct*
  - 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



110-alberi-radicali-08

copyright ©2019 maurizio.patignani@uniroma3.it

## 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  $n$
- ...

110-alberi-radicali-08

copyright ©2019 maurizio.patignani@uniroma3.it

## 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

110-alberi-radicali-08

copyright ©2019 maurizio.patrignani@uniroma3.it

## 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`

110-alberi-radicali-08

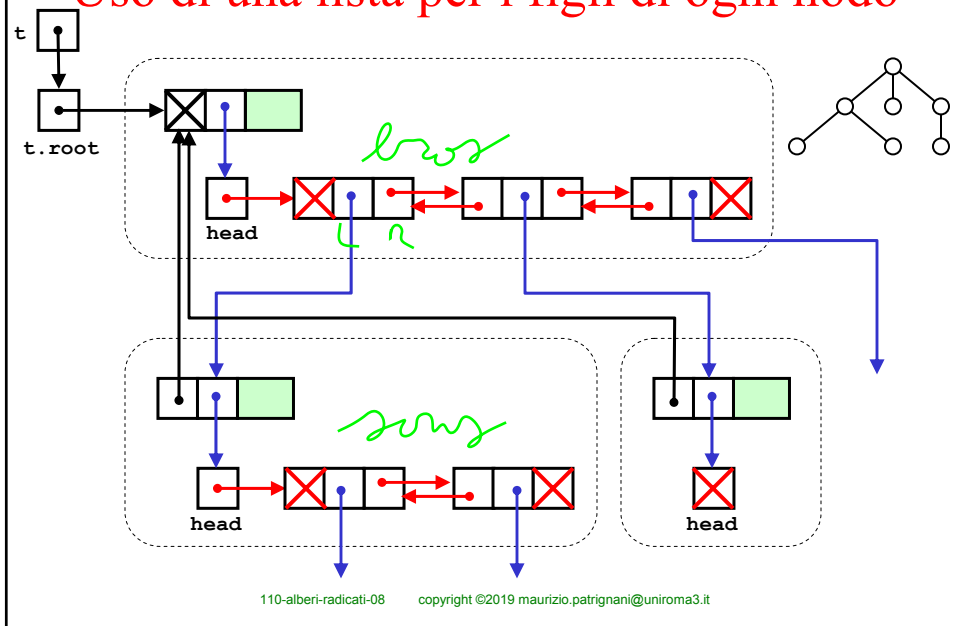
copyright ©2019 maurizio.patrignani@uniroma3.it

## 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

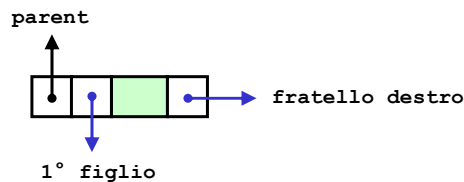
110-alberi-radicali-08 copyright ©2019 maurizio.patignani@uniroma3.it

## Uso di una lista per i figli di ogni nodo



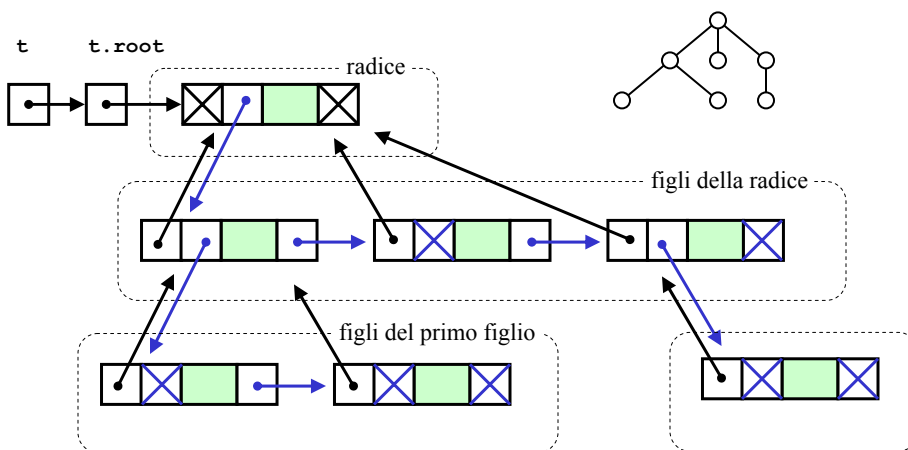
## Struttura “figlio-sinistro-fratello-destro”

- 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



110-alberi-radicali-08 copyright ©2019 maurizio.patignani@uniroma3.it

## Figlio-sinistro-fratello-destro



110-alberi-radicali-08 copyright ©2019 maurizio.patignani@uniroma3.it

## 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`
- ...

110-alberi-radicali-08 copyright ©2019 maurizio.patrignani@uniroma3.it

## 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`

110-alberi-radicali-08 copyright ©2019 maurizio.patrignani@uniroma3.it