# Fondamenti di Informatica (Elettronici)

Alberi e alberi binari – Lezione 37

15 gennaio 2021

# Alberi in Julia


1. Terminologia
2. Alberi
3. Alberi come strutture dati
4. Alberi binari
5. Alberi ordinati
6. Il package DataStructures

## Section 1

# Terminologia

# Grafi

Nella Teoria dei grafi un grafo è una coppia $G = (N, A)$, dove $N$ è un insieme di nodi e $A$ è un insieme di archi.

$N = \{n_i, i = 1, \ldots, N\}$ (nodi) è un insieme qualsiasi finito

$A = \{(n_i, n_j) \in N \times N\} \subseteq N^2$.

Ai nodi è sempre associata un'informazione, semplice o complessa, perlomeno il numero d'ordine di ciascuno.

I grafi possono essere rappresentati graficamente disegnando un simbolo per ogni nodo (spesso un circoletto) e una freccia orientata fal primo al secondo nodo per ogni arco.

# Terminologia

- grafi orientati (Nodi e Archi: (N, A))

# Terminologia

- grafi orientati (Nodi e Archi: (N, A))
- grafi non-orientati (Vertici, spigoli: (V, E)).

# Terminologia

- grafi orientati (Nodi e Archi: (N, A))
- grafi non-orientati (Vertici, spigoli: (V, E)).
- Grafi completi

# Terminologia

- grafi orientati (Nodi e Archi: (N, A))
- grafi non-orientati (Vertici, spigoli: (V, E)).
- Grafi completi
- Grafi connessi

# Terminologia

- grafi orientati (Nodi e Archi: (N, A))
- grafi non-orientati (Vertici, spigoli: (V, E)).
- Grafi completi
- Grafi connessi
- Cammino orientato (su un grafo orientato)

# Terminologia

- grafi orientati (Nodi e Archi: (N, A))
- grafi non-orientati (Vertici, spigoli: (V, E)).
- Grafi completi
- Grafi connessi
- Cammino orientato (su un grafo orientato)
- Cammino (su un grafo non orientato)

# Terminologia

- grafi orientati (Nodi e Archi: (N, A))
- grafi non-orientati (Vertici, spigoli: (V, E)).
- Grafi completi
- Grafi connessi
- Cammino orientato (su un grafo orientato)
- Cammino (su un grafo non orientato)
- Grado (uscente) di un vertice

# Terminologia

- grafi orientati (Nodi e Archi: (N, A))
- grafi non-orientati (Vertici, spigoli: (V, E)).
- Grafi completi
- Grafi connessi
- Cammino orientato (su un grafo orientato)
- Cammino (su un grafo non orientato)
- Grado (uscente) di un vertice
- Grado (entrante) di un vertice

Section 2

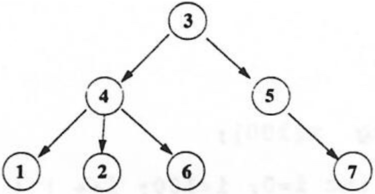
# Alberi

# Albero -definizioni

ALBERO

- Un albero (finito) e' un grafo orientato aciclico dove esiste un nodo con grado di ingresso 0 (detto radice ), mentre tutti gli altri hanno grado di ingresso 1

I nodi con grado di uscita O si dicono foglie



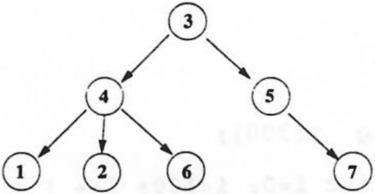

In un albero simmetrico ogni nodo puo' essere radice

# Albero -definizioni

ALBERO

- Un albero (finito) e' un grafo orientato aciclico dove esiste un nodo con grado di ingresso 0 (detto radice ), mentre tutti gli altri hanno grado di ingresso 1

I nodi con grado di uscita O si dicono foglie




- Tali alberi sono detti alberi sorgente

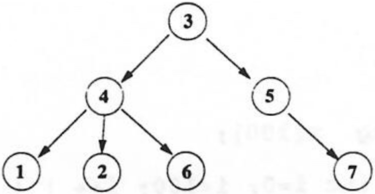
In un albero simmetrico ogni nodo puo' essere radice

# Albero -definizioni

ALBERO

- Un albero (finito) e' un grafo orientato aciclico dove esiste un nodo con grado di ingresso 0 (detto radice ), mentre tutti gli altri hanno grado di ingresso 1

I nodi con grado di uscita O si dicono foglie




- Tali alberi sono detti alberi sorgente
- Si parla anche di alberi pozzo (gli archi hanno versi opposti) e di alberi simmetrici.

In un albero simmetrico ogni nodo puo' essere radice

# Proprietà degli alberi

Gli alberi godono di molte interessanti PROPRIETA'

- Un albero con n nodi ha esattamente n—1 archi

# Proprietà degli alberi

Gli alberi godono di molte interessanti PROPRIETA'

- Un albero con n nodi ha esattamente n—1 archi
- La proprietà discende dal fatto che tutti i nodi meno uno (la radice) hanno esattamente 1 arco entrante

# Proprietà degli alberi

Gli alberi godono di molte interessanti PROPRIETA'

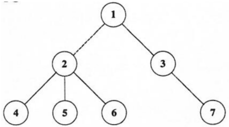
- Un albero con n nodi ha esattamente n—1 archi
- La proprietà discende dal fatto che tutti i nodi meno uno (la radice) hanno esattamente 1 arco entrante

# Proprietà degli alberi

Gli alberi godono di molte interessanti PROPRIETA'

- Un albero con n nodi ha esattamente n—1 archi
- La proprietà discende dal fatto che tutti i nodi meno uno (la radice) hanno esattamente 1 arco entrante

ESEMPIO




Nella figura ci sono 7 nodi e 6 archi. Verificare ia proprietà su altri esempi

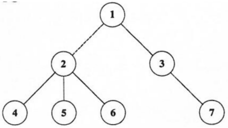
# Proprietà degli alberi

PROPRIETA'

Un albero e' un grafo (debolmente) connesso

- Un grafo connesso si dice debolmente connesso quando eliminando un arco si perde la connessione

ESEMPIO




Se si elimina un arco in un albero con \$n nodi e $n-1$ archi, si sconnette l'albero in due componenti connesse

# Albero — definizioni

Un albero (finito) e' un grafo orientato aciclico dove esiste un nodo con grado di ingresso $O$ (detto radice), mentre tutti gli altri hanno grado di ingresso 1

- I nodi con grado di uscita $O$ si dicono foglie

# Albero — definizioni

Un albero (finito) e' un grafo orientato aciclico dove esiste un nodo con grado di ingresso $O$ (detto radice), mentre tutti gli altri hanno grado di ingresso 1

- I nodi con grado di uscita $O$ si dicono foglie
- Tali alberi sono detti alberi sorgente

# Albero — definizioni

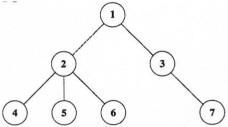
Un albero (finito) e' un grafo orientato aciclico dove esiste un nodo con grado di ingresso $O$ (detto radice), mentre tutti gli altri hanno grado di ingresso 1

- I nodi con grado di uscita $O$ si dicono foglie
- Tali alberi sono detti alberi sorgente

# Albero — definizioni

Un albero (finito) e' un grafo orientato aciclico dove esiste un nodo con grado di ingresso $O$ (detto radice), mentre tutti gli altri hanno grado di ingresso 1

- I nodi con grado di uscita $O$ si dicono foglie
- Tali alberi sono detti alberi sorgente



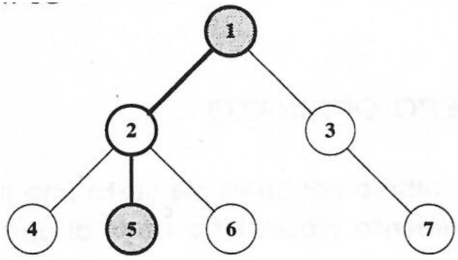

- Si parla anche di alberi pozzo (gli archi hanno versi opposti) e di alberi simmetrici. In un albero simmetrico ogni nodo puo' essere radice

# Proprietà degli alberi

In un albero esiste esattamente un cammino tra la radice e qualsiasi altro nodo

- Il grafo è connesso, e quindi ogni nodo deve essere raggiungibile dalla radice. Il cammino è unico perchè il grado di ingresso dei nodi non radice è sempre 1

ESEMPIO



# Proprietà degli alberi

Un albero rappresenta un ordinamento parziale sull'insieme dei nodi (dove sono tralasciati gli archi indotti dalla riflessivita' e dalla transitivita')

- La chiusura transitiva di ogni grafo orientato aciclico rappresenta una relazione d'ordine

ESEMPIO

La relazione d'ordine rappresentata dall'albero si può esprimere come:

# Proprietà degli alberi

Un albero rappresenta un ordinamento parziale sull'insieme dei nodi (dove sono tralasciati gli archi indotti dalla riflessivita' e dalla transitivita')

- La chiusura transitiva di ogni grafo orientato aciclico rappresenta una relazione d'ordine

ESEMPIO

La relazione d'ordine rappresentata dall'albero si può esprimere come:

- "a precede b se esiste un cammino (orientato) da a a b"

# Definizione induttiva

Un albero T e' un grafo in cui

# Definizione induttiva

Un albero T e' un grafo in cui

- (a) Esiste un nodo r con n $>$ O nodi successori

# Definizione induttiva

Un albero T e' un grafo in cui

- (a) Esiste un nodo r con n $>$ 0 nodi successori
- Il nodo r e' detto radice dell'albero.

# Definizione induttiva

Un albero T e' un grafo in cui

- (a) Esiste un nodo r con n $>$ 0 nodi successori
- Il nodo r e' detto radice dell'albero.

# Definizione induttiva

Un albero T e' un grafo in cui

- (a) Esiste un nodo r con n $>$ O nodi successori
- Il nodo r e' detto radice dell'albero.
- (b) Tutti i nodi di T tranne r possono essere ripartiti in n insiemi disgiunti.

# Definizione induttiva

Un albero T e' un grafo in cui

- (a) Esiste un nodo r con n $>$ O nodi successori
- Il nodo r e' detto radice dell'albero.
- (b) Tutti i nodi di T tranne r possono essere ripartiti in n insiemi disgiunti.
- Ciascuno di tali nodi individua in T un albero T1,. . . ,Tn con radice al,—,an, rispettivamente.

# Definizione induttiva

Un albero T e' un grafo in cui

- (a) Esiste un nodo r con n $>$ O nodi successori
- Il nodo r e' detto radice dell'albero.
- (b) Tutti i nodi di T tranne r possono essere ripartiti in n insiemi disgiunti.
- Ciascuno di tali nodi individua in T un albero T1,. . . ,Tn con radice al,—,an, rispettivamente.
- Grado di un nodo e' il numero dei suoi sottoalberi immediati

# Definizione induttiva

Un albero T e' un grafo in cui

- (a) Esiste un nodo r con $n > 0$ nodi successori
- Il nodo r e' detto radice dell'albero.
- (b) Tutti i nodi di T tranne r possono essere ripartiti in n insiemi disgiunti.
- Ciascuno di tali nodi individua in T un albero T1,. . . ,Tn con radice al,—,an, rispettivamente.
- Grado di un nodo e' il numero dei suoi sottoalberi immediati
- Le foglie hanno grado O

# Analogia genealogica

Per analogia con gli alberi genealogici, si parla di

*nodi figli di un nodo x

- nodo padre di x

# Analogia genealogica

Per analogia con gli alberi genealogici, si parla di

*nodi figli di un nodo x

- nodo padre di x
- La radice non ha padre

# Analogia genealogica

Per analogia con gli alberi genealogici, si parla di

*nodi figli di un nodo x

- nodo padre di x
- La radice non ha padre
- Le foglie non hanno figli

# Analogia genealogica

Per analogia con gli alberi genealogici, si parla di

*nodi figli di un nodo x

- nodo padre di x
- La radice non ha padre
- Le foglie non hanno figli
- Nodi figli dello stesso padre si dicono fratelli

# Analogia genealogica

Per analogia con gli alberi genealogici, si parla di

*nodi figli di un nodo x

- nodo padre di x
- La radice non ha padre
- Le foglie non hanno figli
- Nodi figli dello stesso padre si dicono fratelli
- Antenati (ancestors) di un nodo x sono tutti i nodi da cui x può essere raggiunto con un cammino (orientato)

# Definizioni varie

LIVELLO

di un nodo e' definito induttivamente:

- la radice ha livello 0

PROFONDITA'

di un albero e' il massimo livello dei suoi nodi

ALBERO ORDINATO

E' un albero nel quale sia stato imposto un ordinamento (totale) tra i figli di ogni nodo

# Definizioni varie

LIVELLO

di un nodo e' definito induttivamente:

- la radice ha livello 0
- se un nodo ha livello k allora tutti i suoi figli hanno livello k + 1

PROFONDITA'

di un albero e' il massimo livello dei suoi nodi

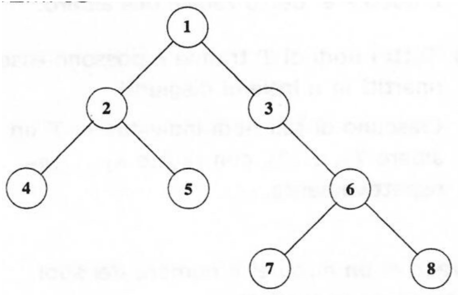
ALBERO ORDINATO

E' un albero nel quale sia stato imposto un ordinamento (totale) tra i figli di ogni nodo

# Alberi binari

Albero binario e' un albero ordinato dove ogni nodo ha al massimo due figli, detti figlio sinistro e figlio destro, rispettivamente

ESEMPIO




Albero binario di profondità 3 (massimo livello dei nodi) ## Alberi binari

ALBERI COMPLETI E BILANCIATI

Un albero binario e' detto completo se —ogni nodo non foglia ha esattamente due figli

grafo con grado entrante di tutii i nodi pari a uno

- ciclo e grafo aciclico

grafo con grado entrante di tutii i nodi pari a uno

- ciclo e grafo aciclico
- radice di un albero

grafo con grado entrante di tutii i nodi pari a uno

- ciclo e grafo aciclico
- radice di un albero
- padre di un nodo

grafo con grado entrante di tutii i nodi pari a uno

- ciclo e grafo aciclico
- radice di un albero
- padre di un nodo
- figli di un nodo

grafo con grado entrante di tutii i nodi pari a uno

- ciclo e grafo aciclico
- radice di un albero
- padre di un nodo
- figli di un nodo
- nodi cugini

grafo con grado entrante di tutii i nodi pari a uno

- ciclo e grafo aciclico
- radice di un albero
- padre di un nodo
- figli di un nodo
- nodi cugini
- profondità di un nodo

# Section 3

## Alberi come strutture dati

# Struttura dati

Sia dato un insieme finito ed ordinato di elementi detti nodi. Se tale insieme è non vuoto, allora un particolare nodo è designato come radice, ed i rimanenti nodi, se esistono, sono a loro volta partizionati in insiemi ordinati disgiunti. Una struttura dati cosi costruita è detta albero ordinato o più semplicemente albero.

Un albero è una struttura fondamentale che si presta a rappresentare svariate situazioni:

- partizioni successive di un insieme in sottoinsiemi disgiunti;

# Struttura dati

Sia dato un insieme finito ed ordinato di elementi detti nodi. Se tale insieme è non vuoto, allora un particolare nodo è designato come radice, ed i rimanenti nodi, se esistono, sono a loro volta partizionati in insiemi ordinati disgiunti. Una struttura dati cosi costruita è detta albero ordinato o più semplicemente albero.

Un albero è una struttura fondamentale che si presta a rappresentare svariate situazioni:

- partizioni successive di un insieme in sottoinsiemi disgiunti;
- organizzazione gerarchiche di dati;

# Struttura dati

Sia dato un insieme finito ed ordinato di elementi detti nodi. Se tale insieme è non vuoto, allora un particolare nodo è designato come radice, ed i rimanenti nodi, se esistono, sono a loro volta partizionati in insiemi ordinati disgiunti. Una struttura dati cosi costruita è detta albero ordinato o più semplicemente albero.

Un albero è una struttura fondamentale che si presta a rappresentare svariate situazioni:

- partizioni successive di un insieme in sottoinsiemi disgiunti;
- organizzazione gerarchiche di dati;
- procedimenti enumerativi o decisionali.

# Struttura dati

Sia dato un insieme finito ed ordinato di elementi detti nodi. Se tale insieme è non vuoto, allora un particolare nodo è designato come radice, ed i rimanenti nodi, se esistono, sono a loro volta partizionati in insiemi ordinati disgiunti. Una struttura dati cosi costruita è detta albero ordinato o più semplicemente albero.

Un albero è una struttura fondamentale che si presta a rappresentare svariate situazioni:

- partizioni successive di un insieme in sottoinsiemi disgiunti;
- organizzazione gerarchiche di dati;
- procedimenti enumerativi o decisionali.
- Un esempio di albero ordinato può essere un albero genealogico di una dinastia, che divide gerarchicamente i suoi membri in successive generazioni, a partire dal progenitore:

# aaaaaa

Altro esempio di albero può essere un albero decisionale in cui ci sono dei nodi di scelta (quelli con ?). Il ramo a sinistra corrisponde a si (test positivo), il ramo a destra corrisponde a no (test negativo).

# aaaaaa

# aaaaaa

## aaaaaa

Section 4

# Alberi binari

# AbstractTree.jl

```
julia> if !isdefined(@__MODULE__, :BinaryNode)
           include("binarytree_core.jl")
       end
rightchild (generic function with 1 method)
```

# AbstractTree.jl

```
julia> ## Things we need to define
       function AbstractTrees.children(node::BinaryNode)
           if isdefined(node, :left)
               if isdefined(node, :right)
                   return (node.left, node.right)
               end
               return (node.left,)
           end
           isdefined(node, :right) && return (node.right,)
           return ()
       end
```

# AbstractTree.jl

```
julia> ## Things that make printing prettier
       AbstractTrees.printnode(io::IO, node::BinaryNode) = pri

julia> ## Optional enhancements
       # These next two definitions allow inference of the ite
       # (They are not sufficient to solve all internal infere
       Base.eltype(::Type{<:TreeIterator{BinaryNode{T}}}) wher

julia> Base.IteratorEltype(::Type{<:TreeIterator{BinaryNode{T}
```

# AbstractTree.jl

```
julia> ## Let's test it. First build a tree.
       root = BinaryNode(0)
BinaryNode{Int64}(0, #undef, #undef, #undef)

julia> l = leftchild(1, root)
BinaryNode{Int64}(1, BinaryNode{Int64}(0, #undef, BinaryNode{I

julia> r = rightchild(2, root)
BinaryNode{Int64}(2, BinaryNode{Int64}(0, #undef, BinaryNode{I

julia> lr = rightchild(3, l)
BinaryNode{Int64}(3, BinaryNode{Int64}(1, BinaryNode{Int64}(0,

julia> print_tree(root)
```

# AbstractTree.jl

```
julia> collect(PostOrderDFS(root))
4-element Array{BinaryNode{Int64},1}:
 BinaryNode{Int64}(3, BinaryNode{Int64}(1, BinaryNode{Int64}(0
 BinaryNode{Int64}(1, BinaryNode{Int64}(0, #undef, BinaryNode{
 BinaryNode{Int64}(2, BinaryNode{Int64}(0, #undef, BinaryNode{
 BinaryNode{Int64}(0, #undef, BinaryNode{Int64}(1, BinaryNode{
```

# AbstractTree.jl

```
julia> @static if isdefined(@__MODULE__, :Test)
           @testset "binarytree_easy.jl" begin
               @test [node.data for node in PostOrderDFS(root)
               @test [node.data for node in PreOrderDFS(root)]
               @test [node.data for node in Leaves(root)] == [
           end
       end

julia> [node.data for node in PostOrderDFS(root)] == [3, 1, 2,
true
```

# Section 5

## Alberi ordinati

# aaaaaa

Section 6

# Il package DataStructures

# aaaaaa

# aaaaaa

# aaaaaa

# aaaaaa