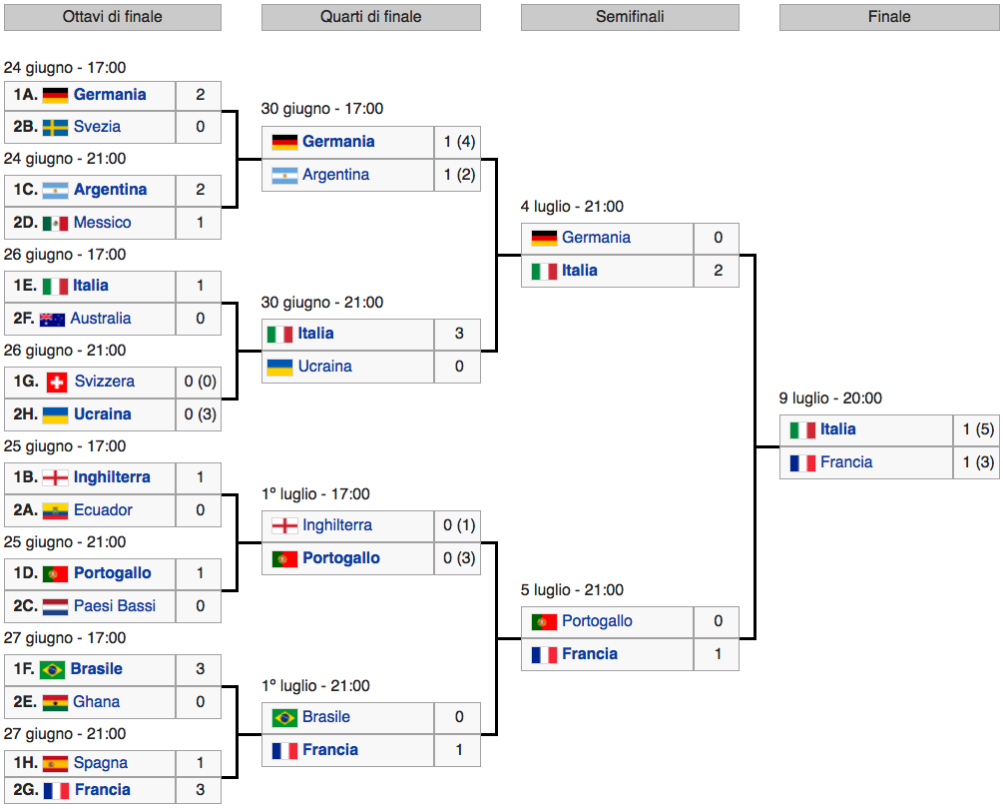
Lezione 8 – Alberi binari

Informatica

3 Maggio 2016

I mondiali di calcio



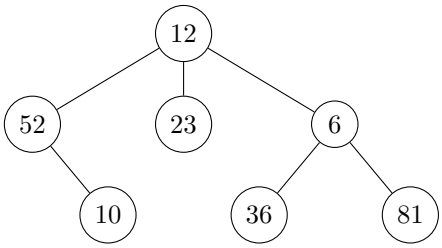
Alberi: definizione

Un albero (in inglese *tree*) ha le seguenti proprietà:

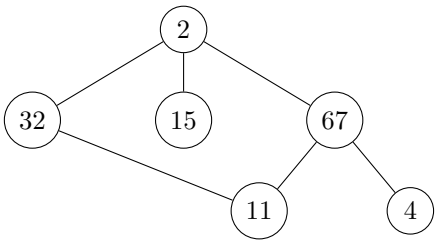
- è una collezione di *nodi* e *archi* che collegano i nodi
- ogni nodo è dotato di *etichetta* e 0 o più *successori*
- i nodi senza successori sono detti *foglie*
- esiste un unico nodo *radice* senza predecessori
- tutti i nodi, eccetto la radice, hanno un solo predecessore
- esiste un unico cammino dalla radice ad ogni altro nodo
- il predecessore di un nodo è chiamato *padre*
- i successori sono detti anche *figli*
- l'*altezza* di un albero è la lunghezza del cammino più lungo dalla radice ad una foglia

Esempi di alberi

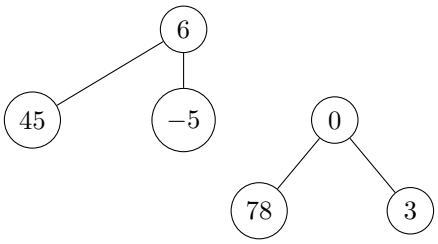
Un albero di interi:



Una struttura che *non è* un albero:



Una struttura che *non è* un albero:

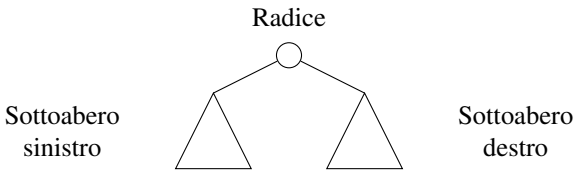


Alberi binari

Un *albero binario* è un albero in cui ogni nodo ha al più 2 successori

In alternativa, possiamo definire gli alberi binari *ricorsivamente*. Un albero binario è:

- un albero vuoto
- oppure è costituito da un nodo, detto radice e da due sottoalberi binari disgiunti, chiamati *sottoalbero sinistro* e *sottoalbero destro*



Alberi binari: interfaccia

Creazione di nuove istanze

`tree()` Crea un albero vuoto

`leaf(a)` Crea un albero composto da un unico nodo etichettato con `a`

`bin(a, l, r)` Data un etichetta `a` e due alberi `l` e `r`, crea l'albero binario con radice etichettata `a`, sottoalbero sinistro `l` e sottoalbero destro `r`

Controllo/Interrogazione

`is_empty(t)` Ritorna `True` se l'albero `t` è vuoto

`label(t)` Ritorna l'etichetta della radice dell'albero `t`

`first_child(t)` Ritorna il sottoalbero sinistro di `t`

`second_child(t)` Ritorna il sottoalbero destro di `t`

Alberi binari: rappresentazione

La definizione ricorsiva di albero binario ci aiuta a definire la sua rappresentazione:

- *albero vuoto:* la lista vuota `[ ]`
- *albero non vuoto:* una lista con tre elementi
  - il primo elemento è l'etichetta della radice
  - il secondo elemento è il sottoalbero sinistro
  - il terzo elemento è il sottoalbero destro

Esempi:

```
[ 'a', [ ], [ ] ]
[ 'a', [ 'b', [ ], [ ] ], [ 'c', [ ], [ ] ] ]
[ 'a', [ 'b', [ ], [ ] ], [ ] ]
[ 'a', [ ], [ 'c', [ ], [ ] ] ]
```

Stabilita la rappresentazione degli alberi possiamo realizzare l'implementazione dell'interfaccia:

```
def tree():
    return []

def leaf(a):
    return [a, [], []]

def bin(a, l, r):
    return [a, l, r]

def is_empty(t):
    return t == []

def label(t):
    if len(t) > 0:
        return t[0]
    else:
        return None

def first_child(t):
    if len(t) > 1:
        return t[1]
    else:
        return None

def second_child(t):
    if len(t) > 2:
        return t[2]
```

```
    else:
        return None

print(" --- ALBERI --- ")
t = bin('a', leaf('c'), leaf('d'))
print(is_empty(t))
print(t)
print(label(t))
r = first_child(t)
l = second_child(t)
print(r)
print(l)
print(is_empty(tree()))
```

Ritorniamo ai mondiali di calcio

Usiamo un albero per rappresentare i risultati del mondiale di calcio

- Ogni nodo è etichettato con una tupla `(squadra1, goal1, squadra2, goal2)`
- La radice dell'albero rappresenta la finale
- Le foglie rappresentano gli ottavi di finale
- L'albero è *completo*: tutti i nodi eccetto le foglie hanno esattamente due figli

Scriviamo una funzione che crea l'albero in modo interattivo:

- la funzione ha come argomenti due alberi con i risultati parziali
- determina quali sono le due squadre che si devono incontrare
- chiede all'utente il risultato della partita
- ritorna l'albero con la nuova partita e gli incontri precedenti

```
def gioca(t1, t2):
    p1 = label(t1)
    if p1[1] > p1[3]:
        s1 = p1[0]
    else:
        s1 = p1[2]
    p2 = label(t2)
    if p2[1] > p2[3]:
        s2 = p2[0]
    else:
        s2 = p2[2]
    print("La prossima partita è: ", s1, s2)
    (g1, g2) = eval(input("Risultato: "))
    ris = bin((s1, g1, s2, g2), t1, t2)
    return ris

print(" --- TORNEO --- ")
semi1 = leaf(('GER', 0, 'ITA', 2))
semi2  = leaf(('POR', 0, 'FRA', 1))
finale = gioca(semi1, semi2)
print(finale)
```