Laboratorio di Python

Alberi, Esercizi su alberi

Università di Bologna

7 maggio 2015

Sommario

Definizione ricorsiva alberi binari

Un albero binario è un insieme finito di nodi. Un albero binario è:

- un insieme vuoto di nodi
- un nodo radice con due alberi binari disgiunti detti sotto-albero sinistro e sotto-albero destro

Definizioni

radice è il nodo dell'albero che non ha nessun arco entrante

padre è il nodo che ha almeno un arco uscente

figlio è un nodo che ha un arco entrante

foglia è il nodo senza archi uscenti

altezza di un nodo : è la distanza del nodo dalla radice dell'albero (la radice ha altezza 0)

Rappresentiamo un albero

Per rappresentare un albero:

- usiamo una struttura dati ricorsiva (sequenze mutabili)
- [1, [], []]: ad esempio usiamo una lista annidata, dove ogni sotto-lista rappresenta un sotto-albero.

Albero binario

Un albero si dice binario se ha al più due figli per ogni nodo.
Nella nostra rappresentazione avrà la seguente forma:
$[x, [y, [], []], [z, [k, [], []], []]]$

Alcune funzioni

Data la nostra rappresentazione di albero binario, implementiamo le seguenti funzioni, per creare la nostra struttura di dato astratto.

- Una funzione che restituisca l'albero vuoto (empty)
- Una funzione che dato un albero restituisca vero se e soltanto se l'albero vuoto (is_empty)
- Una funzione che dato un albero che restituisca il figlio sinistro. (fist_child)
- Una funzione che dato un albero che restituisca il figlio destro. (second_child)
- Una funzione che dato un albero restituisca l'etichetta della radice (label)
- Una funzione che dati tre parametri a, f, s restituisca l'albero che abbia a come l'etichetta della radice, f come figlio sinistro e s come figlio destro (bin)

Funzioni

```
def empty():
    a=[]
    return(a)

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

def first_child(a):
    if not (is_empty(a)):
        f=a[1]
        return(f)
    else:
        return None
```

Funzioni

```
def second_child(a):
    if not is_empty(a):
        s=a[2]
        return(s)
    else:
        return None

def label(a):
    if not is_empty(a):
        return a[0]
    else:
        return None

def bin(a,f,s):
    x=[]
    x.append(a)
    x.append(f)
    x.append(s)
    return (x)
```

Esercizio 1

Utilizzando solo le funzioni definite precedentemente, (empty, is_empty, fist_child, second_child, label, bin) si scriva una funzione Python sost(alb,et,c) che restituisce l'albero che si ottiene da alb sostituendo ogni foglia etichettata *et* l'albero *c*. Si ricordi che una foglia è un nodo con entrambi i figli vuoti; si può assumere che il primo e il terzo argomento siano alberi ben formati.

Soluzione

```
def isleaf(a):
    if not is_empty(a):
        return(is_empty (first_child(a))
                    and is_empty (second_child(a)))
    else:
        return is_empty(a)

def sost(a, et, c):
    if is_empty(a):
        return (a)
    if isleaf(a) and label(a)==et:
        return(c)
    return (bin(label (a), sost(first_child(a),et, c),
                sost(second_child(a),et, c)))
```

Testiamo la funzione

```
x=[0,[3,[6,[],[]],[6,[],[6,[],[]]]],[5,[],[]]]
c=[99,[1,[],[]],[78,[],[]]]
r=sost(x,6,c)
print(r)
```

Esercizio 2

Utilizzando solo le funzioni definite precedentemente, (empty, is_empty, fist_child, second_child, label, bin) si scriva una funzione Python sost2(alb,et,et1) che restituisce l'albero che si ottiene da alb sostituendo ogni foglia etichettata *et* l'etichetta *et*1. Si ricordi che una foglia è un nodo con entrambi i figli vuoti; si può assumere che il primo sia un albero ben formato.

Soluzione

```
def isleaf(a):
    if not is_empty(a):
        return(is_empty (first_child(a))
                    and is_empty (second_child(a)))
    else:
        return is_empty(a)

def sost2(a, et, et1):
    if is_empty(a):
        return (a)
    if isleaf(a) and label(a)==et:
        a=bin(et1,empty(),empty())
        return(a)
    return (bin(label (a), sost(first_child(a),et, et1),
                   sost(second_child(a),et, et1)))
```

Testiamo la funzione

```
x=[0,[3,[6,[],[]],[6,[],[6,[],[]]]],[5,[],[]]]
r=sost(x,6,4)
print(r)
```

Cosa abbiamo fatto?

1. Alberi