Laboratorio di Python

Esercizi su alberi

Università di Bologna

12 maggio 2015

Sommario

Alcune funzioni da ricordare

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)

Esercizio 1

Utilizzando solo le funzioni definite precedentemente, (empty, is_empty, fist_child, second_child, label, bin) In un albero, l'altezza di un nodo è la lunghezza dell'unico cammino che congiunge u alla radice. Per esempio, bin(2,bin(3,empty(),empty()),bin(4,empty(),empty())) ha zero nodi di altezza 2, due nodi di altezza 1 e un solo nodo di altezza 0 (la radice). Manipolando alberi solo con le funzioni assegnate, si scriva una funzione Python maxalt(a) che dato un albero a restituisca la coppia (v,h), dove *v* è il valore con cui è etichettato uno dei nodi a massima altezza e *h* è tale altezza. Se l'albero è vuoto, maxalt restituisce $(-1, -1)$.

Soluzione

```
def maxalt(a):
        if is_empty(a):
                return (-1,-1)
        mh_s=maxalt(first_child(a))
        mh_d=maxalt(second_child(a))
        if mh_s[1]==mh_d[1]==-1: #entrambi figli vuoti
                return (label(a),0)
        if mh_s[1]>mh_d[1]:
                return (mh_s[0],mh_s[1]+1)
        else:
                return (mh_d[0],mh_d[1]+1)
```

Testiamo la funzione

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

Esercizio 2

In un albero binario un nodo interno è pieno se entrambi i suoi figli sono non vuoti. Manipolando alberi solo con le funzioni assegnate, si scriva una funzione Python sommap(a) che dato un albero a etichettato con numeri naturali, restituisca la somma del prodotto dei valori sui nodi pieni con il prodotto dei valori sui nodi non pieni. Il prodotto di un insieme vuoto di valori è 1.

Soluzione 2

```
def pieno(a): # true ss a è pieno
        return not is_empty(first_child(a)) and
                                    not is_empty(second_child(a))
def aux(a): # ritorna una coppia (p,np):
# p prodotto pieni, np prodotto non pieni
        if is_empty(a):
                return (1,1)
        p_f, np_f = aux(first_child(a))
        p_s, np_s = aux(second_child(a))
        if pieno(a):
                return (label(a)*p_f*p_s,np_f*np_s)
        else:
                return (p_f*p_s,label(a)*np_f*np_s)
def sommap(a):
        p_pieni , p_nonpieni = aux(a)
                return p_pieni + p_nonpieni
```

Testiamo la funzione

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

Esercizio 3

Cosa stampa il seguente frammento di codice Python:

```
A = [1,2,3,4,5]
def f(C):
        D = C.append(6)
        C = [9,A]
        return D
A.append(7)
B=f(A)
A[0] = 10
print(B,A)
```

Esercizio 4

Si consideri la seguente funzione foo: si descriva nel modo piùsintetico possibile la sua azione su una lista di numeri interi.

```
def foo(s):
        for i in range(1, len(s)):
                val = s[i]
                j=i-1
                while (j >= 0) and (s[j] > val):
                        s[j+1] = s[j]
                        j=j-1
                s[j+1] = val
```

Cosa abbiamo fatto?