Esercitazione 09

2 Dicembre 2024

Lo scopo di questa esercitazione è implementare due classi (`Node` e `BinarySearchTree`) che implementino un albero binario di ricerca in Python, avente la possibilità di: inserire nell'albero una coppia chiave e valore; cercare una chiave (resituendo il corrispondente valore); convertire l'albero in una stringa da usare, per esempio, con `print`.

Implementazione in Python

La prima classe da implementare è `Node`, per la quale si richiede di implementare i seguenti metodi:

```
def __init__(self, key, value)
```

Inizializza il nodo con una chiave e un valore. Inizializza anche i figli (nodo destro e sinistro) a `None`.

```
def search(self, key)
```

Ricerca la chiave nel nodo corrente e nei suoi sottoalberi. Se la chiave esiste viene ritornato il valore associato, altrimenti `None`.

```
def insert(self, key, value)
```

Inserisce la chiave nel sottoalbero avente radice il nodo corrente (`self`). Se la chiave esiste viene rimpiazzato il valore associato (i.e., la stessa chiave non può apparire in più nodi dell'albero). In ogni caso, non ritorna nulla.

```
def __str__(self)
```

Ritorna una rappresentazione sotto forma di stringa del nodo mostrando solo le chiavi e usando parentesi innestate per rappresentare i diversi sottoalberi. Per esempio una possibile rappresentazione è la seguente: `(27 (1 None (14 (8 (4 2 None) (9 None (10 None 13))) (15 None (20 None 22)))) (30 28 None))`. Non vi sono vincoli particolari se non che tutte le chiavi contenute nell'albero e la sua struttura devono essere ritornati come una stringa. Notate che il metodo `__str__` è speciale (e.g., viene invocato implicitamente quando facciamo `print(object)`).

Per la classe `BinarySearchTree` è necessario implementare gli stessi metodi (che invocheranno in qualche modo i precedenti), con la differenza che il costruttore avrà signature `__init__(self)`, in quanto l'albero dovrà essere inizialmente vuoto.

Extra: wrapping di una implementazione di BST in C

Viene fornita l'implementazione in C di un albero binario di ricerca. Si richiede di creare una classe `CTree` che implementa gli stessi metodi della classe `BinarySearchTree` ma appoggiandosi all'implementazione in C data (il Makefile compila già la libreria condivisa corretta). Si assuma che tutte le chiavi siano interi e tutti i valori siano floating point a precisione singola.

In particolare si osservi che:

- Il metodo `__init__` è già fornito, così come la classe `CTreeNode` per rappresentare le strutture del C.
- Il metodo `__str___` non ha corrispettivo nella versione C, quindi si richiede di implementarlo interamente in Python accedendo alla struttura C. Si ricorda che, ad esempio, per accedere al campo `key` di un oggetto `node` che punta ad un `CTreeNode` è necessario utilizzare `node.contents.key`.
- Esiste già un metodo `__del__` che viene invocato in automatico quando l'oggetto viene distrutto. Questo metodo chiama una funzione in C che dealloca la memoria allocata durante la costruzione dell'albero.
- Si noti come la libreria venga caricata una volta sola sfruttando un attributo della classe e non dell'istanza.

Note

I file .c, .h e il makefile sono usati solo per la parte extra.