Esame di Programmazione II

Appello di giorno 8 Ottobre 2014
Università degli Studi di Catania - Corso di Laurea in Informartica

Testo della Prova

Definizione Iniaziale.
Un *Albero con Molteplicità* (MultiBST) è un albero binario di ricerca (BST) in cui ogni nodo x ha un campo speciale che indica la molteplicità della chiave $key(x)$. Tale campo è indicato con il simbolo $mul(x)$. Esso rappresenta il numero di volte che il valore $key(x)$ è presente nell'insieme degli elementi inseriti nell'albero.
Se $mul(x) = k$ allora sono presenti nell'albero k copie dell'elemento $key(x)$.

Specifiche.
La corretta implementazione di ciascuno dei seguenti esercizi permette l'acquisizione di 9 punti. La corretta implementazione della classe come template è facoltativa e permette l'acquisizione di ulteriori 3 punti:

1. Si fornisca una classe `C++`, denominata `MyMultiBST<H>`, che implementi la seguente interfaccia `MultiBST<H>`, che rappresenta un albero con molteplicit e contenente i seguenti metodi virtuali.
   (a) `MultiBST<H>* ins(H x)` aggiunge un nuovo elemento alla struttura dati e restituisce un puntatore ad un oggetto di tipo `MultiBST<H>`.
   (b) `int multiplicity(H x)` restituisce la molteplicità dell'elemento x, se questo è presente nell'albero, 0 altrimenti;
   (c) `void inorder()` è una procedura che stampa in output gli elementi dell'albero secondo una visita inorder;

   Si crei quindi un'istanza di `MyMultiBST<int>` e si inseriscano al suo interno i seguenti valori:

   ```
   10, 7, 7, 23, 30, 4, 1, 5, 9, 5, 1, 7, 1, 9
   ```

   Si esegua in seguito la stampa dei valori inseriti nell'albero attraverso la procedura inorder. L'output del programma sarà quindi:

   ```
   1, 1, 1, 4, 5, 5, 7, 7, 7, 9, 9, 10, 23, 30
   ```

```
template <class H> class MultiBST {
  public:
     virtual MultiBST<H>* ins(H x) = 0;
     virtual int search(H x) = 0;
     virtual void inorder() = 0;
 }
```

•••

2. Si inserisca all'interno della classe `MyMultiBST<H>` l'implementazione della seguente procedura
   - `MultiBST<H>* del(H` x`)`

   che cancella un'occorrenza dell'elemento x dalla struttura dati, se presente, e restituisce un puntatore ad un oggetto di tipo `MultiBST<H>`.
   Si esegua in seguito la cancellazione, dall'istanza dell'albero creata al passo precedente, degli elementi (nell'ordine) `7, 4, 23, 7` e `7`.
   Si stampi in seguito la lista dei valori dell'albero. L'output del programma sarà quindi:

   ```
   1, 1, 1, 5, 5, 9, 9, 10, 30
   ```

•••

3. Si aggiunga alla classe `MyMultiBST<H>` un nuovo metodo denominato `rank`, la cui definizione è quella data di seguito.

   `int rank(H` x`)`

   Tale metodo prende in input un elemento di tipo `H` e restituisce in output il suo rango, ossia la sua posizione all'interno dell'insieme ordinato contenuto all'interno della struttura dati. Tale valore corrisponde al numero di elementi più piccoli di x, aumentato di uno. Il metodo restituisce il valore 0 se il valore non presente nella struttura dati. Si esegua tre volte la procedura `rank` con input `5, 9` e `30`, rispettivamente. L'output del programma sarà quindi:

   ```
   4 6 9
   ```

•••