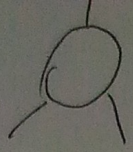


Esame di Programmazione II

Appello di giorno 27 Giugno 2016
Università degli Studi di Catania - Corso di Laurea in Informatica
- Prova B -

Testo della Prova

Definizione Iniaziale.
Un *Brik-Tree* è una struttura dati costituita da un albero binario di ricerca i cui elementi sono delle pile (stack) di valori numerici. Sia s una pila di valori numerici. Indichiamo con $\alpha(s)$ la somma degli elementi in essa contenuti, con $\beta(s)$ il numero di elementi in essa contenuti e con $\gamma(s)$ il valore che si trova in testa alla pila. Le pile contenute all'interno del Brik-Tree sono ordinate in base ai seguenti criteri:

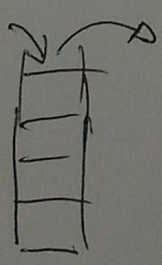
- se $\alpha(s_1) > \alpha(s_2)$ allora $s_1 > s_2$
- se $\alpha(s_1) = \alpha(s_2)$ e $\beta(s_1) > \beta(s_2)$ allora $s_1 > s_2$
- se $\alpha(s_1) = \alpha(s_2)$, $\beta(s_1) = \beta(s_2)$ e $\gamma(s_1) > \gamma(s_2)$ allora $c_1 > c_2$

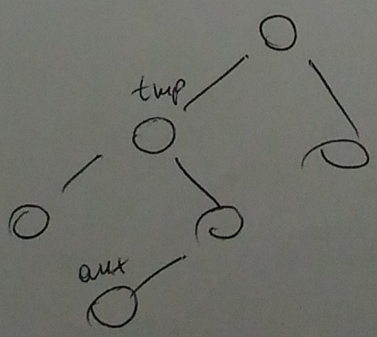
Un nuovo elemento può essere inserito all'interno dell'albero in due modi, in una nuova pila inizialmente vuota, oppure nella pila più piccola del Brik-Tree. Nella struttura dati non sono presenti pile vuote. Quando da una pila viene estratto l'ultimo elemento, questa viene eliminata dall'albero. L'unico elemento che può essere eliminato dalla pila è l'elemento che si trova in testa alla pila più piccola contenuta all'interno del BrikTree.

Si fornisca una classe `C++`, denominata `MyBrikTree<H>`, che implementi la seguente interfaccia `BrikTree<H>` contenente i seguenti metodi virtuali.

1. `BrikTree<H>* ins(H x)` aggiunge una nuova pila al BrikTree, contenente il solo elmento x. Il numero di pile dell'albero aumenta di uno. La funzione restituisce un puntatore ad un oggetto di tipo BrikTree<H>;
2. `BrikTree<H>* push(H x)` aggiunge un nuovo elemento x alla pila più piccola contenuta nel BrikTree. Il numero di pile dell'albero non aumenta a meno che l'albero non sia vuoto. In questo caso una nuova pila verrà creata contenente l'elemento x. La funzione restituisce un puntatore ad un oggetto di tipo BrikTree<H>;
3. `H* pop()` estrae l'elemento di testa dalla pila più piccola contenuta nel BrikTree. Se l'elemento estratto è l'unico presente nella pila allora quest'ultima viene eliminata dall'albero. La funzione restituisce un puntatore ad un oggetto H;
4. `int search(H x)` restituisce 1 se x è presente nella struttura dati, 0 altrimenti;
5. `void print()` è una procedura che stampa in output gli elementi della struttura. La stampa procede dalla prima all'ultima pila dell'albero. Per ogni pila gli elementi vengono stampati a partire dall'elemento in testa.

Nota Bene: La coda dovrà essere implementata utilizzando una lista come struttura dati di base.





Si crei quindi un'istanza di `MyBrikTree<int>` e si inseriscano 10 nuove pile contenenti rispettivamente i valori

```
15 12 6 9 10 4 2 30 23 11
```

Si esegua, in seguito, per quattro volte consecutive: l'estrazione di un elemento attraverso la procedura `pop()` e il suo re-inserimento nella struttura dati, attraverso la procedura `push()`.
L'output del programma sarà quindi:

```
6 2 4
15
9 10
23
11 12
30
```

•••

```
template <class H> class BrikTree {
  public:
     virtual BrikTree<H>* ins(H x) = 0;
     virtual BrikTree<H>* push(H x) = 0;
     virtual H* pop() = 0;
     virtual int search(H x) = 0;
     virtual void print() = 0;
  }
```

•••

Valutazione.
La corretta implementazione dell'albero permette di acquisire 9 punti. La corretta implementazione della pila permette di acquisire 9 punti. La corretta implementazione della classe `MyBrikTree` permette di acquisire ulteriori 9 punti. La corretta implementazione delle classi come template è facoltativa e permette l'acquisizione di ulteriori 3 punti:

•••