Esercitazione 3
Alberi binari

Corso di Fondamenti di Informatica II

BIAR2 (Ing. Informatica e Automatica) e BSIR2 (Ing. dei Sistemi)

A.A. 2013/2014

25 Ottobre 2013

Sommario

Scopo della esercitazione è quello di realizzare una struttura dati per gestire gli alberi binari ed implementare semplici algoritmi di visita.

1 Rappresentazione di un albero binario tramite array

Un *albero binario* è una struttura dati che permette di rappresentare un albero in cui ad ogni nodo è associato un valore e che ha al più 2 nodi figli. Oltre ai metodi del TDA albero, un albero binario supporta i metodi che, per un sottoalbero T, restituiscono il sottoalbero sinistro di T (o `null` se T non ha sottoalbero sinistro) e il sottoalbero destro di T (o `null` se T non ha sottoalbero destro).

In questo esercizio si vuole implementare il TDA albero binario utilizzando una rappresentazione basata su array (Figura 1). La radice è memorizzata nella posizione di indice 1 dell'array. Se un nodo è memorizzato nella posizione di indice i dell'array, il suo figlio sinistro è nella posizione di indice $2i$, mentre il suo figlio destro è nella posizione di indice $2i + 1$. I nodi figli non presenti possono essere rappresentati attraverso riferimenti `null`. Per allocare l'array si può assumere che l'albero non debba contenere più di un certo numero costante (ad esempio, 100) di nodi. Ci si riferisca per i dettagli al libro di testo [1].

Materiale di Supporto. Viene fornita la classe `BinTreeUtil`, contenente un metodo di stampa generico per il tipo `BinTree<E>`.

Programma Java. Si implementi la classe Java generica `BinTree<E>`, che rappresenti il tipo di dati albero binario a valori di tipo `E`. La classe `BinTree<E>` dovrÃ implementare la seguente interfaccia.

```
public BinTree(E val)
```

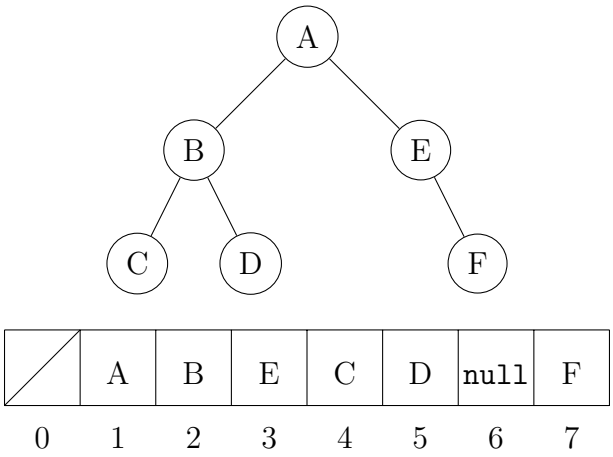

Figura 1: Albero binario rappresentato tramite array.

Costruisce un albero binario con un singolo nodo di valore `val`.

```
public E getRoot()
```

Restituisce il valore dell'elemento alla radice dell'albero.

```
public void setRoot(E val)
```

Modifica il valore dell'elemento alla radice dell'albero.

```
public BinTree<E> addLeftChild(E val)
```

Associa un nuovo figlio sinistro di valore `val` alla radice e restituisce il corrispondente sottoalbero.

```
public BinTree<E> addRightChild(E val)
```

Associa un nuovo figlio destro di valore `val` alla radice e restituisce il corrispondente sottoalbero.

```
public BinTree<E> getLeftSubtree()
```

Determina il figlio sinistro della radice e restituisce il corrispondente sottoalbero (o `null` se la radice non ha figlio sinistro).

```
public BinTree<E> getRightSubtree()
```

Determina il figlio destro della radice e restituisce il corrispondente sottoalbero (o `null` se la radice non ha figlio destro).

Suggerimenti. Si consiglia di usare i seguenti campi privati nella classe `BinTree<E>`:

```
private E   root; // valore dell'elemento
private int rank; // indice dell'elemento nell'array
private BinTree<E> subtrees[]; // array condiviso,
        // contenente i riferimenti
        // ai sottoalberi dell'albero principale
```

Per realizzare i metodi `addLeftChild()` e `addRightChild()` si consiglia inoltre di utilizzare un costruttore privato ausiliario:

```
private BinTree(BinTree<E> subtrees[], int r, E val)
```

che permetta di inserire un nuovo nodo di valore `val` e il corrispondente riferimento a sottoalbero all'indice `r` nell'array condiviso `subtrees`.

Debugging. Si testi la classe `BinTree` attraverso la classe di supporto `Test`, contenente un metodo `main` in cui (i) viene creata una istanza T di `BinTree<String>`; (ii) T viene popolata con alcuni valori; (iii) vengono fatte alcune operazioni su T. L'istanza può essere stampata con il metodo `print` della classe `BinTreeUtil`.

2 Visita di alberi binari

Programma Java. Si aggiungano i seguenti metodi alla classe `BinTreeUtil`.

```
public static <E> void printInorder(BinTree<E> t)
```

Stampa i nodi dell'albero `t` secondo una visita inorder (visita simmetrica).

```
public static <E> void printLivelli(BinTree<E> t)
```

Stampa i nodi dell'albero `t` per livelli.

3 Albero speculare (opzionale)

Dato un albero binario T, l'albero speculare a T è definito come l'albero ottenuto da T sostituendo la relazione di figlio sinistro con quella di figlio destro e viceversa; ad esempio l'albero speculare a quello in Figura 1 è mostrato in Figura 2.

Programma Java. Si aggiunga alla classe `BinTreeUtil` un metodo

```
public static <E> BinTree<E> mirror(BinTree<E> T)
```

che dato un albero binario T restituisca un nuovo albero speculare a T.

Riferimenti bibliografici

[1] M. T. Goodrich and R. Tamassia. *Strutture dati e algoritmi in Java.* Zanichelli, 2007.

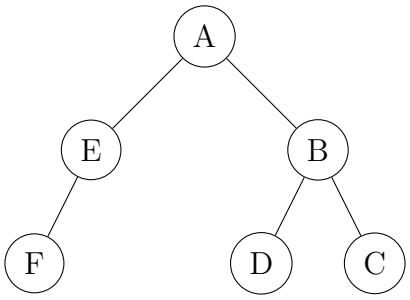

Figura 2: Albero speculare all'albero di Figura 1