Facoltà di Ingegneria

Corso di laurea in Ingegneria Informatica

“Albero di ricerca binaria (Bilanciato)”

Professore del corso di POO:
Prof. Libero Nigro

Studente:
Carmelo Gugliotta

ANNO ACCADEMICO 2020-2021

Progetto: Albero di ricerca binaria (Bilanciato)

Il progetto consente di implementare la classe ABR (vista a lezione) che espone però, una nuova struttura di iterazione e la proprietà di bilanciamento.

La gerarchia di classi utilizzata per lo sviluppo del progetto è la seguente:

1. ABR_Bilanciato (**Classe concreta**)

1. ABR_Bilanciato (Classe Concreta)

La classe concreta sulla base di ciò che è stato discusso a lezione ridefinisce e implementa i seguenti metodi:

- size()
  - Il seguente metodo restituisce la size dell'albero.
    Avendo a disposizione per ogni elemento cfd e cfs, la size totale sarà pari alla somma di cfd e cfs della radice più sé stessa.
- Add (T x)
  - Riceve un parametro x, che è il nuovo oggetto da inserire all'interno della struttura dati, istanzia uno Stack di nodi e richiama il metodo Add(Nodo<T> radice, T x ,Stack<Nodo<T>> StackNodi_NB) che si occupa dell'aggiunta vera e propria dell'elemento passato. Lancia IllegalStateException() se l'elemento da aggiungere è già presente all'interno dell'albero.
    L'aggiornamento dei parametri cfs e cfd avviene risalendo(a seguito della catena di ritorni della ricorsione) il percorso del nodo raggiunto sino alla radice, aumentando di uno, cfs o cfd (dipende dal percorso intrapreso).
    Successivamente all'aggiornamento dei parametri se sussiste uno sbilanciamento del nodo considerato, su quest'ultimo viene effettuato il push all'interno dello stack di nodi.
    Il Bilanciamento, che avviene nel caso lo stack non fosse vuoto, è implementato seguendo le indicazioni del pdf.

- Remove(T x)
  - Riceve un parametro x, che è l'oggetto da rimuovere all'interno della struttura dati, istanzia uno Stack di nodi e richiama il metodo remove(Nodo<T> radice, T x ,Stack<Nodo<T>> StackNodi_NB) che si occupa della rimozione vera e propria dell'elemento passato.
    L'aggiornamento dei parametri cfs e cfd avviene risalendo(a seguito della catena di ritorni della ricorsione) il percorso del nodo raggiunto sino alla radice, diminuendo di uno, cfs o cfd (dipende dal percorso intrapreso).

    Attenzione: Nel caso in cui il nodo da rimuovere avesse tutti i due figli
    - **Nel 1 sotto caso:** La cardinalità destra sarà pari alla cardinalità del nodo promosso (o può anche essere diminuita di uno),la cardinalità sinistra rimarrà immutata.
    - **Nel 2 sotto caso:** l'elemento da rimuovere è il minimo, quindi per ogni padre si avrà un figlio che il potenziale elemento da

rimuovere, questo implica che per ogni padre trovato si deve diminuire la sua cardinalità sinistra di 1.
Successivamente alla promozione (vista la procedura iterativa) e l'aggiornamento del cfd, controlliamo a partire dalla radice promossa il bilanciamento dell'albero tramite il metodo ricorsivo, visto a lezione, isBilanciato(radice).

Successivamente all'aggiornamento dei parametri se sussiste uno sbilanciamento del nodo considerato, quest'ultimo viene pushato all'interno dello stack di nodi.
Il Bilanciamento, che avviene nel caso lo stack non fosse vuoto, è implementato seguendo le indicazioni del pdf.

All'interno della classe concreta è implementata anche una nuova struttura di iterazione, utilizzando la tecnica proposta a lezione e tenendo conto di eccezioni come:

ConcurrentModificationException();**IllegalStateException()**;

N.B: Si noti che successivamente alla remove di un elemento si effettua la ricostruzione dello stack, perché il bilanciamento può implicare anche cambiamenti di percorsi della struttura ad albero. La ricostruzione avviene pushando nello stack gli elementi ancora non visitati (maggiori del nodo rimosso).

Come richiesto è disponibile, all'interno di poo.agendina, AgendinaABR_Bilanciato.java.