Inlämningsuppgift 2 - AVL-träd
DV1490

Christian Nordahl, Andreas Jonasson, & Joakim Ståhle Nilsson

April 18, 2019

1 Uppgift

Ni skall implementera den abstrakta datatypen (ADT) AVL-träd som implementerar följande API:

- `void insert(T element);` - Lägger till ett element i AVL-trädet. Skall hantera så att inga dupliceringar tillåts läggas till.
- `void remove(T element);` - Tar bort det efterfrågade elementet i AVL-trädet om det finns.
- `boolean find(T element);` - Letar upp det element som efterfrågas. Returnerar true om det finns, false om det inte finns.
- `ArrayList<T> preOrderWalk();` - Traverserar AVL-trädet enligt principen pre order och lägger till varje nods värde i ArrayListen som returneras.
- `ArrayList<T> inOrderWalk();` - Traverserar AVL-trädet enligt principen in order och lägger till varje nods värde i ArrayListen som returneras.
- `ArrayList<T> postOrderWalk();` - Traverserar AVL-trädet enligt principen post order och lägger till varje nods värde i ArrayListen som returneras.
- `int getTreeHeight();` - Returnerar höjden på trädet. Om trädet är tomt returnerar ni -1.
- `T getMin();` - Letar upp och returnerar det minsta värdet i AVL-trädet.

- `T getMax();` - Letar upp och returnerar det största värdet i AVL-trädet.

Er klass skall ha namnet AVL, och skall ligga i paketet AVL, och den skall kunna hantera vilken datatyp som helst, d.v.s. ni skall implementera en generisk klass. Ni ska anta att `int compareTo()` metoden är implementerad för alla datatyper som används i samband med AVL-trädet.

AVL-trädet skall kunna hantera en godtycklig mängd data och ingen duplicerad data får lagras i trädet. Ni skall även se till att hantera eventuella fel som kan uppstå vid användning av ert AVL-träd genom att kasta ett exception. Tänk noga igenom vilka fel som kan uppstå vid användning.

När ni tar bort ett element från AVL-trädet och den nod ni skall ta bort har två barn, skall ni ersätta denna nod med noden som har det högsta värdet i det vänstra barnets subträd.

När er lösning är färdigimplementerad, och ni klarar de tester som är givna, skall ni se till att er lösning klarar av det testscript som ges vid sidan av denna uppgiftsspecifikation. Mer information kring detta testscript finns under Appendix A.

Tips: Implementera först ett fungerande binärt sökträd och gå sedan över till att modifiera det till ett AVL-träd.

2 Medföljande Projektfiler

Tillsammans med inlämningen medföljer två mappar för att arbeta i, antingen i *Eclipse* eller om ni hellre sitter i en texteditor och kompilerar via terminal med hjälp av en `makefile`. Det antas att de studenter som vill använda sig av en `makefile` vet hur dessa fungerar på förhand eller tar reda på detta själva, ingen hjälp erbjuds för detta. Mapparna innehåller en mapp- och paketstruktur som ni skall efterfölja, och innehåller filerna `AVL.java` och `Tests.java`.

Ni finner även ett par hjälpmetoder i AVL.java som ser ut som följer:

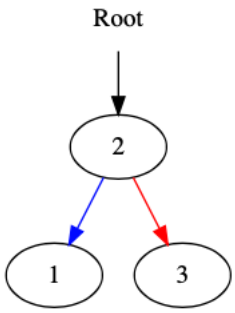
```
public String toGraphViz();
private String toGraphViz(String data, Node current);
```

Funktionen `String toGraphViz();` kan ni använda för att få ut en grafisk representation av ert nuvarande träd. T.ex., ett binärt sökträd med noderna 1, 2, och 3 där nod 2 är roten ger följande sträng:

```
digraph {
Root [shape=plaintext];
```

```
"Root"  -> 0 [color=black];
0 [label="2"];
0 -> 1 [color=blue];
1 [label="1"];
2 [label=nill,style=invis];
1 -> 2 [style=invis];
3 [label=nill,style=invis];
1 -> 3 [style=invis];
0 -> 4 [color=red];
4 [label="3"];
5 [label=nill,style=invis];
4 -> 5 [style=invis];
6 [label=nill,style=invis];
4 -> 6 [style=invis];
}
```

Denna sträng kan ni sedan använda för att få ut en grafisk verison av ert AVL-träd. Går ni sedan in på hemsidan `www.webgraphviz.com` och matar in strängen får ni följande bild, där blå pil pekar på vänsterbarn och röd på högerbarn:



OBS! Ovan nämnda metoder är baserade på den lösning vi själva skapat. Ni får modifiera variabelnamn, parametrar och annat efter behov. Dessa funktioner skall finnas i AVL klassen, och de kräver att ni har en klassvariabel vid namn nodeID av typen int för att fungera.

2.1 Tester

Den medföljande filen `Tests.java` innehåller tester, för att du iterativt skall kunna säkerhetsställa att din kod uppfyller de specificerade kraven. Dessa

tester kräver att namngivningen är konsekvent med detta dokument, men givet att klassen `AVL` är deklarerad i filen `AVL.java` som ligger i paketet `AVL` så kommer dessa tester att kompilera och informera om eventuella fel i din implementation. Det är starkt rekommenderat att du *inte* sätter dig in i denna kod, utan enbart fokuserar på att implementera den specificerade klassen och passera testerna enligt utskrifter. Du är fri att skapa dina egna tester för din implementation, men det är starkt rekommenderat att använda de medföljande stommarna. Testscriptet är baserat på den här strukturen för filer och paket.

3 Kravspecifikation och Inlämning

- All kod skall skrivas av dig själv och inlämningarna är individuella. Alla inlämningar kommer genomgå en plagieringskontroll.
- Följande, och endast följande, paket är tillåtna att användas:
  - `import java.util.NoSuchElementException;`
  - `import java.util.Iterator;`
  - `import java.util.ArrayList;`
- AVL-trädet skall uppfylla alla de krav som specificerats under Rubrik 1.
- Inga kompileringsvarningar eller kompileringsfel får finnas i er lösning som ni skickar in. Dessa skall ses över innan inlämning.
- Innan inlämning skall det medföljande testscriptet köras. Passerar inte de tester som utförs av detta script är din implementation ej redo för inlämning.
  - De studenter som lämnar in en implementation som inte klarar av detta script kommer automatiskt att bli underkända.
  - För körinstruktioner vänligen se Appendix A.
  - Notera att detta script *endast* är ämnat att fungera på skolans datormiljö. Studenter som önskar köra scriptet på egen dator erbjuds ingen hjälp med detta.
- Ni skall *endast* lämna in filen `AVL.java.txt` som innehåller den generiska klassen `AVL` som ligger i paketet AVL. **Ingen main-metod skall finnas i er fil!** Om en fil med felaktigt namn eller med en main-metod implementerad lämnas in kommer denna inlämning automatiskt att bli underkänd, då den varken kommer passera ert givna testscript eller vårt rättningsscript.

A Körinstruktioner för testscript

Det medföljande testscriptet skall köras och passeras innan inlämning. Detta script kontrollerar så att er inlämning har korrekt namn, inte använder några otillåtna paket, med mera. Detta script tillhandahålles för att göra rättnings- processen transparent. Inlämnad kod som *inte* klarar av detta script kommer inte rättas.

Notera att er implementation inte garanterat uppfyller de krav som ställs i uppgiften även om testerna passerar. Detta script är det första och mest grundläggande test som utförs. Ett passerat script är alltså inte ett garanterat godkänt på uppgiften.

För att köra testscriptet kopierar ni filen `AVL.java` till mappen *Student_Folder*. Byt sedan filändelsen från `.java` till `.java.txt`. Högerklicka på filen *Submission_Test.ps1* och välj *Kör med PowerShell*. Skrivs texten *"Submission script succesfully finished."* ut är er implementation redo för inlämning.