Laboration 3

Sorterat binärt träd

Uppgiften syftar främst till att öva och examinera lärandemålen 1 t.o.m. 5. Laborationen är uppbyggd som en stegvis påbyggnad av ett sorterat binärt träd.

---

a) Skapa ett sorterat binärt träd.

I denna uppgift ska ni skapa ett program där ni kan hantera ett sorterat binärt träd, vars informationsdel är ett heltal. Ni behöver börja med att göra en design, där ni börjar med en datatyp som hanterar själva noderna och de eventuella funktioner som behövs för dessa. Nodtypen skall alltså bestå av ett värde, en pekare till ett vänsterbarn och en pekare till ett högerbarn. Ni kan också implementera en pekare till en förälder om ni vill.

Ni ska definiera en ADT för ett sorterat binärt träd som är implementerat med hjälp av dessa trädnoder. Börja med att göra operationer som lägger till data och som skriver ut innehållet i trädet i de olika utskriftsordningarna (in-, pre- och postorder).

Designa också en huvudmeny (ej del av ADTn) som låter användaren välja mellan att lägga in tal, skriva ut trädet och avsluta programmet. Vid utskrift ska det finnas en undermeny så att man vid utskrift får välja mellan pre-, in- eller postorder. Det behöver inte vara avancerade menyer, utan det räcker med att alternativen skrivs ut och att du som användare får välja vad som ska hända härnäst i programmet.

När designen är klar är det dags att implementera det hela, så att ni kan testa om det fungerar.

b) Leta och ta ut från trädet.

Ni ska nu öka funktionaliteten för ert träd genom att lägga till möjlighet att söka efter ett visst värde, respektive ta bort ett visst värde. Börja med att göra en design som beskriver hur det ska gå till, samt tänk ut en strategi. Implementera sedan era funktioner.

Tips: Börja med att ta bort noder med inget eller max ett barn. Utöka sedan funktionaliteten så att alla noder kan tas bort, om ni får problem med detta så fråga labbassen om råd.

Det ska också finnas möjlighet att skriva ut antalet noder (n), nivådjupet samt $\log_2(n+1)-1$ som är det teoretiska minvärdet för antalet nivåer (= djupet i trädet). Observera att detta teoretiska minvärde måste vara ett heltal och att när en nivå blir full medför en extra nod att man måste ner till nästa nivå, dvs antalet nivåer ökar med 1. Genom att jämföra det teoretiska minvärdet för nivådjupet med det faktiska nivådjupet på trädet kan man få en uppfattning om hur balanserat trädet är.

Glöm inte att uppdatera menyn i huvudprogrammet så att användaren kan testa den nya funktionaliteten som nu finns på trädet.

c) Balansera trädet

Bygg vidare på ditt träd genom att lägga till en rekursiv metod som balanserar trädet efter följande algoritm:

1) skriv ut alla element sorterat till en array.
2) töm trädet.
3) bygg trädet rekursivt enligt punkt 4.
4) ta det mittersta värdet från fältet och lägg i roten sedan:
    I) vänster del av fältet till vänster delträd
    II) höger del av fältet till höger delträd till alla värden i fältet är inlagda.

Tips: Om fältet innehåller ett jämnt antal värden, så får ni ta ett av de mittersta värdena som er rot.

(Om ni hellre vill lösa balanseringen genom att rotera trädet så går det bra).

Glöm inte att uppdatera menyn i huvudprogrammet så att användaren också kan välja att balansera trädet.

Redovisning: Visa och diskutera trädet med labbassen. Visa olika testfall och var beredd på att labbassen vill köra egna testfall på trädet. Redovisningen ska göras när hela trädet är färdigt men diskutera gärna de olika delarna med labbassen under labbens gång.
Redovisas förslagsvis vid laborationstillfälle 5.