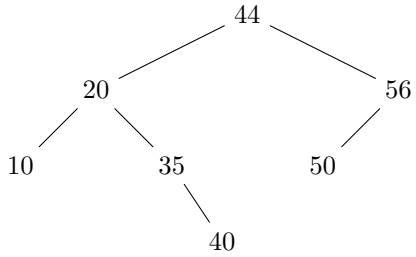
Lektion 2 – Linjära strukturer och träd

Filip Strömbäck

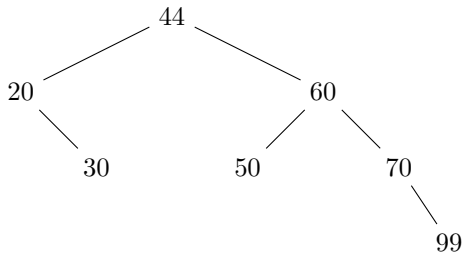
13 september 2018

Övningsuppgifter

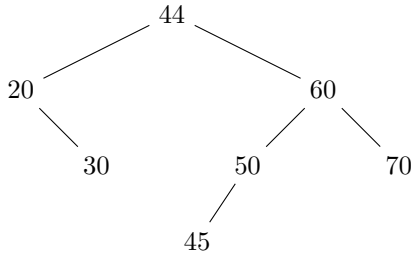
1. Ta bort 50 ur följande AVL-träd:



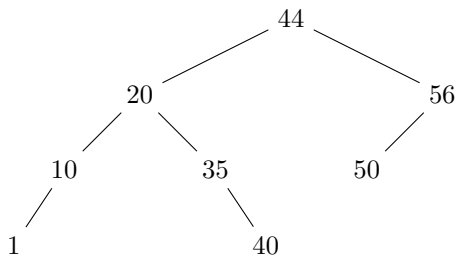
2. Ta bort 30 ur följande AVL-träd:



3. Ta bort 30 ur följande AVL-träd:



4. Ta bort 44 ur följande AVL-träd:



5. Du sitter och programmerar en väldigt primitiv dator. Datorn har en indataström, en utdataström och ett temporärt minne. Datorn kan utföra tre operationer:

   1: läs ett tecken från indataströmmen och skriv ut det direkt

   2: läs ett tecken från indataströmmen och lägg det i minnet

   3: hämta ett tecken från minnet och skriv ut det

   Minnet i datorn är också primitivt, och du kan välja mellan att använda en stack eller en kö som minne. Om sekvensen X Y U V W läses av indataströmmen, är det då möjligt att skriva ett program som producerar någon av utdatasekvenserna X U W V Y eller Y U W X V om minnet är:

   (a) en stack?

   (b) en kö?

6. Beskriv hur det är möjligt att med två stackar implementera en kö.

   (a) Beskriv idén.

   (b) Vad är körtiden i det bästa och värsta fallet för `enqueue` och `dequeue`?

   (c) Vad är den amorterade körtiden för `enqueue` och `dequeue`?

7. Beskriv med hjälp av pseudokod hur följande operationer kan implementeras i ett binärt sökträd. Ingen av operationerna får använda mer än $\mathcal{O}(h)$ tid, där h är trädets höjd.

   (a) `find-min(x)`, som returnerar den minsta noden i ett träd eller ett delträd `x`.

   (b) `next-larger(x)`, som returnerar den nod som är närmast större än `x`.

8. Du vill hålla reda på vad det kostar att äta lunch på alla lunchresaturanger i stan. Du har valt att använda ett (balanserat) binärt sökträd för ändamålet. Du har insett att du ofta är intresserad av att veta hur många restauranger som har lunch för under ett visst belopp, `count-cheaper(x)`. Det sökträd som finns i standardbiblioteket har dock inte en effektiv implementation av just denna operation (den tar $\mathcal{O}(n)$ i värsta fallet). Beskriv hur du kan snabba upp implementationen av `count-cheaper(x)` så att den förbrukar $\mathcal{O}(h) = \mathcal{O}(log(n))$ tid.

9. Ta bort 32 ur följande AVL-träd (kom ihåg att vi ersätter med närmast efterföljande nod):