Tema 3: Träd
Rapport AVL-träd

Tobias Ahnhem `toah5501`
Daniel Andersson `daan2233`
Eric Egan `ereg8941`

2 februari 2017

1 Inledning

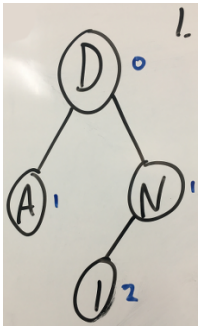
Uppgiften består i att sortera in bokstäver i ett balanserat binärt sökträd, ett s.k AVL-träd. Som data använder vi bokstäver ur gruppmedlemmarnas namn: *D, A, N, I , E, L, G, H* och *F*.

1.1 AVL-träd

Ett AVL-träd är en binärt sökträd, men som också ställer villkoret att det hela tiden skall vara balanserat. För att säkerställa detta kontrolleras höjden på varje nods subträd. Skillnaden i höjd mellan två syskonträd får högst vara 1. I det fall en nod endast har ett subträd föreställer man sig att det andra trädet finns och då har höjden 0.

2 Genomförande

2.1 Insättning av D, A, N och I

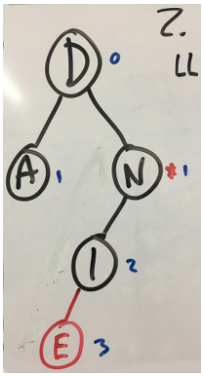


Steg 1

Vi sätter in bokstäverna D, A, N, I en efter en och får trädet till vänster. Efter varje insättning i trädet så kontrollerar vi att det inte bryter mot villkoret för AVL-träd. Detta görs för alla noder som ligger mellan den senast insatta och roten av trädet. Nod I har inga subträd så vi föreställer oss att det finns två stycken som båda har höjden 0. Eftersom höjdskillnaden är < 1 så är I okej. Nod N har ett subträd med höjden 1 och ett subträd med höjden 0. Inte heller detta bryter mot villkoret. Sista kontrollen görs på roten, D.

D har subträd som är 1 respektive 2 höga och eftersom inte heller det bryter mot villkoret kan vi dra slutsatsen att trädet är tillräckligt balanserat.

2.2 Insättning av E

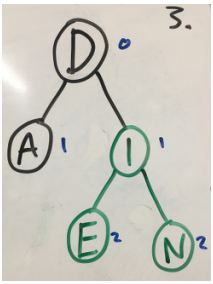


Steg 2

Vi placerar in E i trädet och gör sedan samma kontroll som i *Steg 1* d.v.s. börjar på den senast insatta noden och fortsätter hela vägen upp till roten. När vi kommer till N upptäcker vi att den bryter villkoret. Ns vänstra subträd har höjden 2 och det högra 0 vilket ger att skillnaden är $>$ 1. För att balansera trädet och återigen uppfylla villkoret måste nod E flyttas upp en nivå, från 3 till 2, samtidigt som trädet fortfarande är korrekt sorterat.

Detta löser vi genom att utföra en s.k. *enkelrotation* på den del av trädet där problemet uppstår. I detta fall från E upp till noden där problemet upptäcktes, N. För att få reda på åt vilket håll vi ska rotera tittar vi på vilket subträd som är högst med utgångspunkt i toppnoden, N, i trädet som ska roteras. I detta fallet är Ns vänstra träd högst. Sedan upprepar vi denna kontroll på toppnoden i Ns vänstra subträd, i det här fallet I. Då kan vi se att även för I så är det det vänstra subträdet som är högst. Det betyder att vi har en *vänster-vänster*-situation och trädet ska därför roteras i motsatt riktning d.v.s. höger.

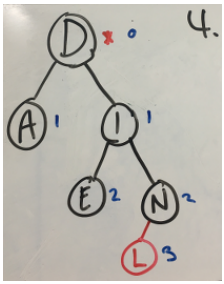
2.3 Enkelrotation åt höger



Steg 3

Steg 3 nedan visar hur trädet ser ut efter enkelrotationen. Subträdet *E* - *I* - *N* har roterat åt höger kring mittnoden *I* och resultatet blir att *I* nu är rot med *N* som höger subträd och *E* som vänster (vilket är samma som innan rotationen). Den enda lösningen för att flytta upp *E* är att också flytta upp *I*. För att hålla trädet sorterat måste då *N* flyttas ned och bli höger subträd till *I*.
En kontroll nedifrån och upp visar nu att vi inte har någon nod som bryter mot villkoret för ett AVL-träd.

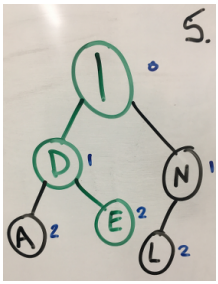
2.4 Insättning av L



Steg 4

Nästa bokstav som ska sättas in är L och den placeras till vänster om N. Vi kontrollerar om någon nod nu bryter mot villkoret och börjar med L och arbetar oss upp mot D. När vi kommer till D upptäcker vi att den har en obalans mellan höger och vänster subträd. Precis som förra gången tittar vi nu på vilket träd det är som är högst för varje nod under D och får det till *höger-höger* denna gång. För att lösa detta behöver vi därför genomföra en enkelrotation åt vänster. Situationen är liknande den i *Steg 2*, men nu har även två av de ingående noderna (I och N) varsitt barn (E resp. L). På samma sätt som i *Steg 2* roteras trädet kring mittpunkten I, men denna gång åt vänster.

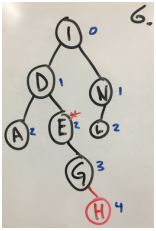
2.5 Enkelrotation åt vänster



Steg 5

Steg 5 visar hur trädet ser ut efter vänsterrotationen. På samma sätt som tidigare har mittpunkten för rotationen, *I*, flyttats upp en nivå och är nu rot i hela trädet. En konsekvens av att *I* flyttas upp blir att *D* flyttas ner och blir vänster barn till *I*. Eftersom det är ett binärt träd så fungerar det inte att *I* har kopplingar till tre noder (*D, E, N*), så *E* sorteras in som högerbarn till *D*. Det löser problemet med antal barn till *I* och trädet är nu också korrekt sorterat. Trädet kontrolleras ännu en gång för att se att inga noder bryter mot AVL-villkoret.

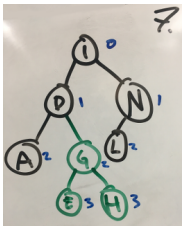
2.6 Insättning av H



Steg 6

Mellan *Steg 5* och *Steg 6* har nästa bokstav, *G*, satts in utan problem. Nästa bokstav är *H* som sätts in och efter en kontroll nerifrån och upp, hittar vi problem vid nod *E*. Den är av samma typ som i *Steg 4* d.v.s. *höger-höger* och vi löser detta med en vänsterrotation enligt samma mönster som *Steg 5*.

2.7 Enkelrotation åt vänster

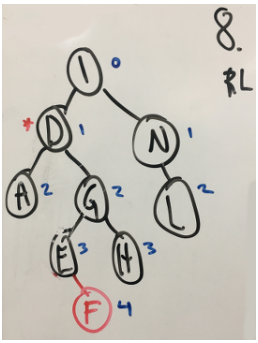


Steg 7

G flyttas upp en nivå och drar då även upp H en nivå. E måste då falla ned som vänster barn till G.
Samtliga noder i trädet kontrolleras för att säkerställa att ingen nod bryter mot villkoret.

I nästa steg sätter vi in F som inte ingår i våra namn, men som används för att framkalla en dubbelrotation.

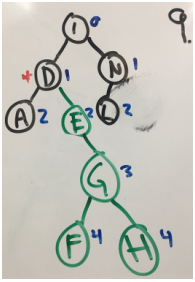
2.8 Insättning av F



Steg 8

F sätts in som höger barn till *E* och efter en kontroll av alla noder hittas ett problem hos *D*. Vänster subträd har höjd 1 och höger subträd har höjd 3. För att ta reda på vilken rotation som krävs för att lösa detta gör vi en kontroll på samma sätt som tidigare. Denna gång blir resultatet dock *höger-vänster*. Detta är en ny situation som inte kan lösas med hjälp av en enkelrotation eftersom det inte skulle flytta upp *F* tillräckligt för att balansera trädet. Istället får vi lösa det med en *dubbelrotation*.

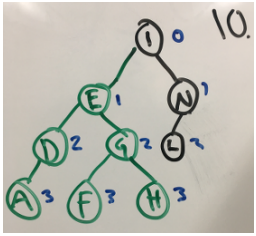
2.9 Dubbelrotation Steg 1



Steg 9

Målet med dubbelrotationen är samma som vid enkelrotation; att få upp det senast insatta elementet, som orsakade obalansen, en nivå. En dubbelrotation är sammansatt av två enkelrotationer där rotationerna är åt olika håll, så antingen först *vänster* och sedan *höger* eller tvärt om. Trädet som ska roteras hade formen *höger-vänster* (sett uppifrån och ned) och på samma sätt som med enkelrotationer så ska vi rotera åt motsatt håll. Eftersom trädet är *höger-vänster* ska vi rotera *vänster-höger*. Vi vill dock börja rotera nerifrån d.v.s. så nära det senast insatta elementet som möjligt, så den första rotationen skall vara en *höger*-rotation som sedan följs upp med en *vänster*-rotation.
Vid den första rotationen (*höger*) flyttas E upp och byter plats med G. G behåller sitt högerbarn och får F som vänsterbarn.
Steg 9 visar att det fortfarande finns ett problem hos D eftersom obalansen 1 mot 3 inte har försvunnit.

2.10 Dubbelrotation Steg 2



Steg 10

Nu är det dags för den andra delen av dubbelrotationen, en *vänster*-rotation. Vi roterar åt vänster med E som mittpunkt. När E flyttas upp så drar den även med sig hela sitt högra subträd och alla noder i det trädet flyttas upp en nivå. D faller ner och blir vänsterbarn till E. Trädet är nu sorterat och balanserat eftersom inga noder bryter mot AVL-villkoret.