DV1: Datavetenskapens Byggstenar

Analysmoment av OU5

Theodor Jonsson

2022-02-24

ens18trn thjo0102

Inledning

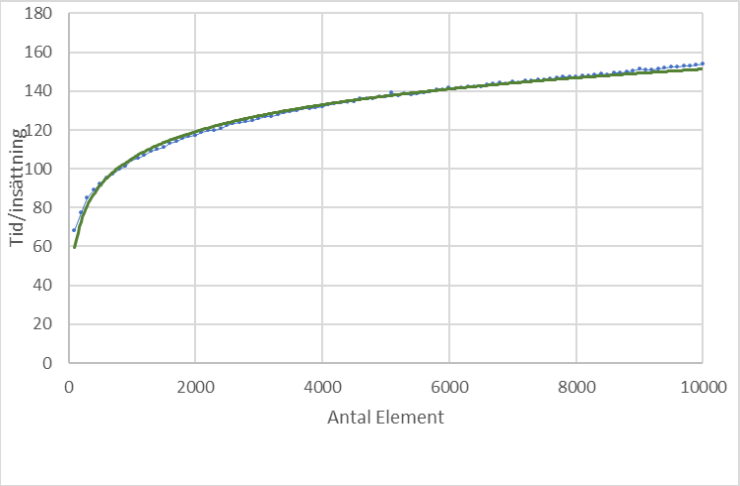
I ett komplett träd är både medeltidskomplexiteten och värstafallstidskomplexiteten för sökning av ett värde O(log n). Det är därför intressant att studera hur tidskomplexiteten ser ut för sökning av ett värde i ett binärt sökträd där data blir insatt in i trädet i slumpmässig ordning.

Metod

Vi skapar ett binärt sökträd och sätter in 100 element med slumpmässiga värden från 0-100. Sedan mäter vi tiden som algoritmen tar för att söka igenom och hitta alla värden i trädet. Sedan divideras denna tid med antal element för att få den genomsnittliga tiden för sökning av ett värde. Detta sedan upprepas 100 gånger för att få den genomsnittliga tiden för den mängden element. Till sist upprepas allt detta och ökar max antal element och värde med 100 tills maxvärdet har nått 10000.

Resultat

Resultaten presenteras i Figur 1. Tiden för insättning av element anges i nanosekunder på den lodräta axeln samt antalet element anges på den vågräta axeln. Den teoretiska värstafallstidskomplexiteten för sökning i ett icke-komplett träd blir O(n). Detta kan hända om de insatta värdena hamnar i storleksordning och alla element hamnar efter en gren. Sedan att de elementet som söks ligger längst nere i grenen vilket gör att hela trädet behöver bli kollat igenom innan man hittar de rätta elementet.



Figur 1. Blåa datapunkter visar tid för sökning i ökande mängd element tiden mäts i nanosekunder.

Diskussion

De experimentella resultatet förhåller sig mycket mer mot en tidskomplexitet som liknar ett komplett träd än de värsta fall för ett icke-komplett. Detta anser jag att de är en mycket större chans att ett slumpmässigt insatt träd liknar mer ett komplett träd med vissa avvikelser men ej stora nog för att påverka resultatet markant. Värdena i detta experiment är valda för att försöka minimera påverkan av slumpmässighet. Startvärdet valdes för att få ett markant mindre värde än maxvärdet för att enklare kunna se om resultatet skulle likna en logaritmisk funktion. Upprepningen av varje mängd valdes för att minimera slumpmässigheten. Ökning av antal element per iteration valdes för att få många datapunkter.