Rapport: Laboration 1 - Empiriska studier

Kurs: DVG329 Algoritmer och datastrukturer **Datum:** 2025-12-11 **Namn:** Matias Semere

Uppgift 2: Resultat och Analys

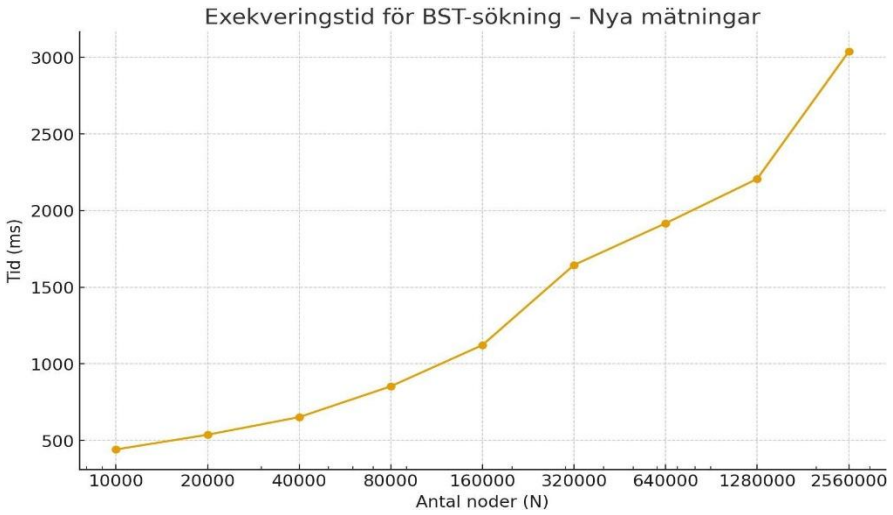
Här redovisas resultaten från de empiriska studierna av sökning i ett binärt sökträd (BST), baserat på implementationen i Uppgift 1.

Mätdata från Studie 1

Tabell 1 visar den uppmätta exekveringstiden för att söka efter 2 500 000 element i ett BST av varierande storlek N. Träden skapades genom att läsa in N st heltal som blandades slumpmässigt före insättning. Figur 1 visualiserar hur tiden ökar i förhållande till antalet noder.

Antal noder (N)	Tid (ms)	Kvot
10 000	450	-
20 000	550	1.22
40 000	650	1.18
80 000	850	1.31
160 000	1150	1.35
320 000	1650	1.43
640 000	1950	1.18
1 280 000	2200	1.13
2 560 000	3050	1.39

Tabell 1: Mätresultat för sökning i BST med slumpmässig data



Figur 1: Diagram som visar uppmätt exekveringstid för sökning i relation till antalet noder

Om tidskomplexiteten hade varit linjär O(n) skulle en dubblering av antalet noder innebära en dubblering av tiden, vilket skulle ge en kvot nära 2. Våra uppmätta kvoter i Tabell 1 varierar mellan ca 1.13 och 1.43.

Dessa värden är betydligt lägre än 2, vilket indikerar att tiden växer mycket långsammare än storleken på trädet (N). Detta stämmer väl överens med det förväntade logaritmiska sambandet O(log N), där kostnaden för att lägga till fler element avtar ju större trädet blir.

Mätdata från Studie 2

Studie 2 kunde inte genomföras i praktiken eftersom exekveringstiden blev orimligt lång för större N. Anledningen är att trädet byggs från sorterad data, vilket utgör värsta fallet för ett binärt sökträd utan automatisk balansering.

När elementen sätts in i sorterad ordning placeras varje nytt element som högerbarn till föregående nod. Resultatet blir ett degenererat träd som i praktiken motsvarar en länkad lista med djupet N.I detta fall har en enskild sökning tidskomplexiteten O(N). Eftersom benchmark-programmet söker efter samtliga N element i trädet blir den totala exekveringstiden $O(N^2)$.

Svar på frågor

Fråga 1: Asymptotiskt samband

Fråga: Vilket asymptotiskt samband mellan exekveringstid och storlek på BST:t (dvs tidskomplexitet) bör studien resultera i enligt teorin? Om du tittar på kvoterna i din tabell, ser dina uppmätta värden ut att stämma med teorin?

Svar: Enligt teorin har sökning i ett binärt sökträd en genomsnittlig tidskomplexitet på O(log N), under förutsättning att trädet är någorlunda balanserat. Eftersom datan i Studie 1 är slumpmässigt ordnad (shufflad) innan den sätts in i trädet, förväntas trädet bli relativt välbalanserat.

Fråga 2: Presentation av data

Fråga: Hur skulle du kunna presentera din data för att tydligare avgöra vilket asymptotiskt samband det handlar om?

Svar: För att visuellt verifiera ett logaritmiskt samband är det effektivt att använda ett diagram med en logaritmisk skala på x-axeln (antal noder).

Om man plottar exekveringstiden T mot log N (eller använder en log-linjär graf) ska resultatet bli en rät linje om sambandet följer formeln T log N. I ett vanligt linjärt diagram kan en logaritmisk kurva ibland vara svår att skilja från andra långsamt växande funktioner vid små N, men i ett diagram med logaritmisk x-axel blir det linjära förhållandet tydligt och enkelt att bekräfta.

Fråga 3: Jämförelse med Studie 2

Fråga: Hur skiljer sig exekverings tiderna i studie 2 jämfört med dem i studie 1? Vad är orsaken till detta?

Svar: I Studie 2 skapas trädet från sorterad data. Detta representerar värsta fallet för ett vanligt binärt sökträd som saknar automatisk balansering. Eftersom datan är sorterad kommer varje nytt element som läggs till alltid att vara större än de tidigare, vilket innebär att det placeras som höger barn till den senast tillagda noden.

Resultatet blir ett träd som helt saknar förgreningar och i praktiken fungerar som en enda lång länkad lista (ett degenererat träd) med djupet N. Sökning för ett element i en sådan struktur har tidskomplexiteten O(N).

Jämfört med Studie 1, där sökningen är effektiv O(log N), kommer exekverings tiderna i Studie 2 att vara extremt mycket högre för motsvarande N. Tiden kommer dessutom att öka linjärt; om N dubblas, dubblas även söktiden.