Laboration 3

Algoritmer och datastrukturer HT2023

Selma Abbassi

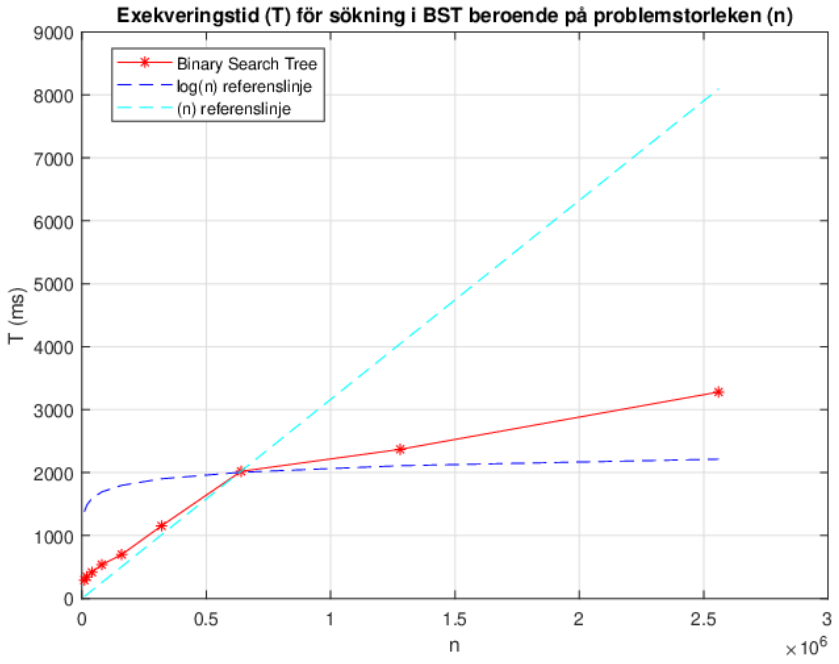
Studie 1: Binärt sökträd (BST) med osorterade data

TABELL 1 visar resultatet av den empiriska studien som gjordes för det binära sökträdet med osorterade data. Tre sökningar exekverades med ett konstant antal element att söka efter (2 500 000) för varje mängd (n) i det binära sökträdet. Mellan varje exekvering skapades ett nytt binärt sökträd och listan blandades innan insättning. Tid (ms) avg visar medelvärdet för varje tre sökningar per (n) antal element. Dessa användes för att rita GRAF 1 och GRAF 2.

Osorterat				
Antal element (n)	**Tid (ms) #1**	**Tid (ms) #2**	**Tid (ms) #3**	**Tid (ms) avg**
10 000	309	278	286	291
20 000	347	350	340	346
40 000	407	412	428	416
80 000	545	528	537	537
160 000	786	648	661	698
320 000	1136	1152	1175	1154
640 000	2047	1997	2015	2020
1 280 000	2193	2272	2646	2370
2 560 000	2856	3407	3578	3280

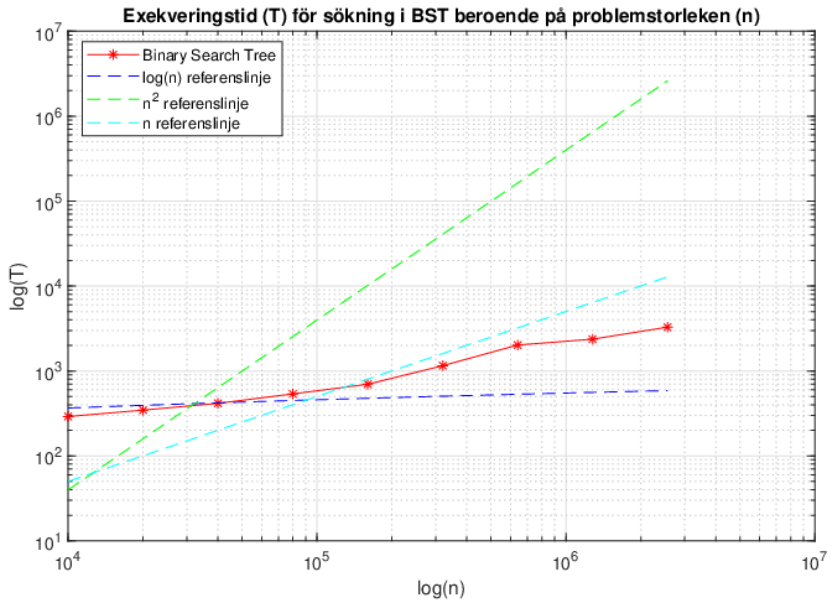
TABELL 1: SÖKNING PÅ 2 500 000 ELEMENT I BINÄRT SÖKTRÄD MED OSORTERADE ELEMENT (N)

GRAF 1 visar mätvärdena från TABELL 1 i linjär skala tillsammans med referenslinjerna (log(n)) och (n).



GRAF 1: EXEKVERINGSTIDER FÖR SÖKNING I ETT BINÄRT SÖKTRÄD I LINJÄR SKALA

GRAF 2 visar mätvärdena från TABELL 1 i logaritmisk skala tillsammans med referenslinjerna (log(n)), (n^2) och (n).



GRAF 2: EXEKVERINGSTIDER FÖR SÖKNING I ETT BINÄRT SÖKTRÄD I LOGARITMISK SKALA

Studie 2: Binärt sökträd (BST) med sorterade data

Tabell 2 visar resultatet av den empiriska studien som gjordes på det binära sökträdet med sorterade data. Samma metod som för studie 1 användes.

Sorterat				
Antal element (n)	**Tid (ms) #1**	**Tid (ms) #2**	**Tid (ms) #3**	**Tid (ms) avg**
10 000	67 660	68 060	67 647	67 789
20 000	143 285	144 606	144 630	144 173

TABELL 2: SÖKNING PÅ 2 500 000 ELEMENT I BINÄRT SÖKTRÄD MED SORTERADE ELEMENT (N)

Svar på frågor

Fråga 1

För stude 1: Hur beror exekveringstiderna för sökning (av ett konstant antal element) i ditt BST på storleken på BST:t? Vilket asymptotiskt samband (dvs tidskomplexitet) resulterar din studie i? Stämmer detta överens med teorin?

Den förväntade tidskomplexiteten för ett binärt sökträd i bästa fallet är O(1) då elementet som söks hittas direkt. I genomsnittsfallet har sökningen tidskomplexiteten O(log(n)) då en sök-operation utförs för varje steg som tas ner i det binära sökträdet. I värsta fallet har algoritmen tidskomplexiteten O(n) då trädet är obalanserat och måste söka igenom varje element i trädet.

Exekveringstiderna i studie 1 ser ut att få en logaritmisk ökning, dvs log(n). Detta syns särskilt tydligt i GRAF 1 där mätvärdena följer referenslinjen för O(log(n)). Tittar man på GRAF 2 så ser det ut som att vissa mätvärden följer referenslinjen för O(n) vid några mätpunkter. Detta skulle kunna förklaras av att värdena som sattes in vid dessa exekveringar, fick ett högt eller lågt värde som första värde i listan, vilket skapade en något obalanserad trädstruktur. Grafen planar dock sedan ut i linje med referenslinjen O(log(n)).

Jag skulle säga att den empiriska studien visar på att implementationen av BST är korrekt då den följer den teoretiska tidskomplexiteten; i graferna ser mätvärdena ut att följa referenslinjen för log(n) i de flesta fallen.

Fråga 2

Hur skiljer sig exekveringstiderna i studie 2 jämfört med dem i studie 1? Vad är orsaken till detta?

I studie 2 visar sig exekveringstiderna bli betydligt långsammare. Detta skulle kunna förklaras av att elementen som insätts i BST redan är sorterade och resulterar i ett väldigt obalanserat träd. Det första värdet sätts till roten av trädet, och eftersom detta värde är det minsta eller högsta så kommer noden och resterande noder att endast få en barnnod till höger eller vänster. I slutändan får vi i en länkad lista vars tidskomplexitet att söka i blir det värsta fallet för BST, dvs O(n).

Om vi har en sorterad lista [1,2,3,4,5] så kommer trädet att innehålla noder som endast har en barnnod till höger, se BILD 1. Om listan är omvänt sorterad [5,4,3,2,1], så kommer noderna endast ha barnnoder till vänster, se BILD 2.

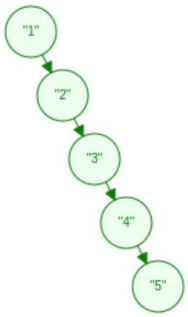


BILD 1

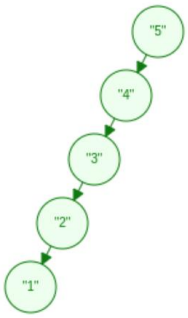


BILD 2