Datastrukturer och algoritmer - laboration 3

Grupp 2 - Mats Högberg, Maxim Goretskyy

Februari 2016

1 Komplexitetsanalys

1.1 SortedLinkedListSet

1.1.1 Add

Add-metoden traverserar genom listan för att hitta rätt plats att lägga in elementet. I värsta fall så kommer vi lägga in elementet på sista platsen, vilket ger oss värstafallskomplexiteten $O(n)$. Själva insättningen, dvs länka in elementet när vi väl har hittat platsen går på konstant tid.

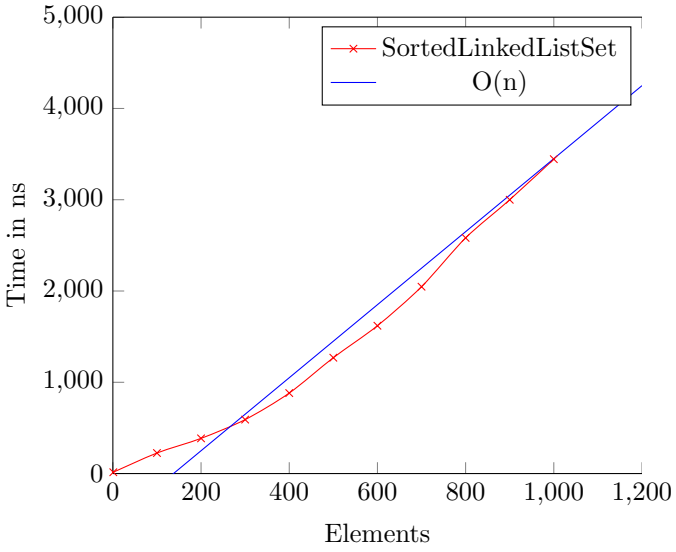
1.1.2 Remove

Remove-metoden har även den komplexiteten $O(n)$. Vi drar den slutsatsen genom att se att i värsta fall måste man gå genom hela länkade-listan med n element om det man vill ta bort är sista elementet (eller inte finns i listan alls). Att jämföra ett element med ett annat går på konstant tid och vi behöver inte bry oss om det. Även omlänkningen som motsvarar själva borttagningen av ett element går på konstant tid.

1.1.3 Contains

Contains-metoden har också komplexiteten $O(n)$ av samma anledning som för remove.

1.1.4 Plot



Vi ser att kurvan för SortedLinkedListSet inte är helt spikrak, men man ser ändå en tydlig linjär trend för de uppmätta värdena. Hade vi kört testet med större antal element så hade förmodligen kurvan rätat ut sig ännu mer, men då hade testerna behövt köra mycket längre tid för att säkerställa korrekta snittider.

1.2 SplayTreeSet

1.2.1 Add

Add metoden använder sig av binärsökning (addRecursive-metoden), dvs vi halverar antal element vi behöver jobba med för varje iteration. Vi får alltså en komplexitet som motsvarar trädets höjd. I snitt leder det till $O(log(n))$ komplexitet, men om trädets höjd är mindre optimal så kan det ta $O(n)$ tid. Splaying:en tar på samma sätt som binärsökningen också $O(log(n))$ tid i snitt, och i värsta (och unika/osannolika) fall $O(n)$.

Eftersom splaying inte är nästlad inuti binärasökningen så går de tillsammans på $O(2log(n)) = O(log(n))$. n här är antal element i trädet.

Att länka om noder i trädet när man väl har hittat dem går på konstant ($O(1)$) tid.

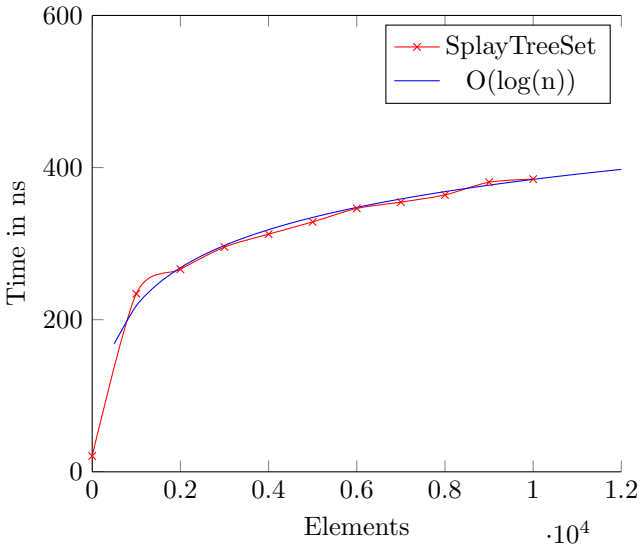
1.2.2 Remove

Eftersom remove också använder sig av binärsökning och splaying så appliceras samma resonemang här som på add. Remove tar alltså också ($log(n)$) tid i snitt.

1.2.3 Contains

Contains använder sig också av binärsökningen och splaying. Återigen appliceras samma resonemang även här som på add och remove. Contains tar alltså också $(log(n))$ tid i snitt.

1.2.4 Plot



Även här ser vi att de uppmätta värdena följer den teoretiska kurvan $O(log(n))$ hyfsat väl. Här körde vi testet med många fler element än för SortedLinkedListSet, och vi ser att kurvan stämmer bättre överens med den teoretiska kurvan, vilket var ganska förväntat.