Komplexitetsanalys

Niklas Baerveldt, Lucas Ruud

January 2018

1

1.1 a)

Om man uppskattar komplexiteten på de tre algoritmerna med handviftning så får man:
I det första fallet så ser vi att det finns tre for-loopar som går ungefär till n varje gång, vilket leder till komplexiteten $O(n^3)$. I det andra fallet så har vi två for-loopar som går ungefär till n vilket leder till komplexiteten $O(n^2)$. Och i det tredje fallet så har vi bara en for-loop som går till n vilket ger komplexiteten $O(n)$.
Om vi istället räknar ut komplexiteten mer precist för de tre algoritmerna och antar att det som "kostar" något är aritmetiska operationer (+,-,* etc.) och att tilldelningar är gratis. Vi har alltså antagit att a.length betecknas av n, alltså om något går till < a.length så går det till n-1. Indexbeteckningar har vi valt att behålla som de är, alltså i=0, j=i, k=i, dessa återkommer i summorna. Varje aritmetisk operation har vi bestämt att de kostar "1". If satser kostar inte heller någonting, men eventuella jämförelser i if-satserna kostar om det är aritmetiska operationer.
En for-loop som går från i=0 till i < a.length, a.length kallar vi för n. Här utförs två aritmetiska operationer som vi ser i for-loopen, nämligen jämförelsen för att se om i är $\leq$ n-1 och ökningen av i (i++). Sedan har vi en inre summa som går från j=i till j < a.length. Även här ser vi att vi utför två aritmetiska operationer i for-loopen, jämförelse och ökning. Men efteråt så sker en till jämförelse i en if-sats vilket lägger till en operation. Till sist så kommer den en tredje for-loop som går från k=i till k $\leq$ j. Här utförs tre aritmetiska operationer, de två vanliga i "huvudet" på for-loopen, jämförelse och ökning, men även en addition inne i for-loopen.
Detta leder till att vi får följande matematiska uttryck för komplexiteten:

Algoritm 1:
I den första summan utförs två aritmetiska operationer i "huvudet" på for-loopen dessa ger en 2:a inuti denna summa, sedan så kommer en till for-loop vilket betecknas av en ny summa inne i den första summan. Den nya summan utför även den två aritmetiska operationer i "huvudet" av for-loopen, men den utför även en extra aritmetisk operation, nämligen en jämförelse vilket lägger till ett till antalet utförda operationer som kostar. Detta ger oss en trea inuti den andra summan. Efter detta så kommer ytterligare en for-loop vilket betecknas av ytterligare en summa inne i den andra summan. Den tredje summan utför, som de tidigare summorna, två aritmetiska operationer i "huvudet", men även en ytterligare aritmetisk operation i "kroppen". Detta ger en trea inuti den tredje summan.
$\sum_{i=0}^{n-1}(2+\sum_{j=i}^{n-1}(3+\sum_{k=i}^{j}3)) = \{$Distribuera in den andra summan$\} =$
$\sum_{i=0}^{n-1}(2+\sum_{j=i}^{n-1}3+\sum_{j=i}^{n-1}\sum_{k=i}^{j}3) = \{$Flytta ut treorna från de innersta summorna$\}$
$= \sum_{i=0}^{n-1}(2+3\sum_{j=i}^{n-1}1+3\sum_{j=i}^{n-1}\sum_{k=i}^{j}1) = \{$Utveckla den sista summan (övre-nedre+1)$\}$
$= \sum_{i=0}^{n-1}(2+3\sum_{j=i}^{n-1}1+3\sum_{j=i}^{n-1}(j-i+1)) = \{$Förenkla den sista summan$\}$
$= \sum_{i=0}^{n-1}(2+3\sum_{j=i}^{n-1}1+3(\sum_{j=i}^{n-1}j-\sum_{j=i}^{n-1}i+\sum_{j=i}^{n-1}1)) = \{$Multiplicera in trean i den sista summan$\}$
$= \sum_{i=0}^{n-1}(2+3\sum_{j=i}^{n-1}1+3\sum_{j=i}^{n-1}j-3\sum_{j=i}^{n-1}i+3\sum_{j=i}^{n-1}1) = \{$Förenkla alla inre summor med (övre-nedre+1) samt summationsregler$\}$
$= \sum_{i=0}^{n-1}(2+3(n-1-i+1)+3\frac{(n-1+i)(n-1-i+1)}{2}-3i(n-1-i+1)+3(n-1-i+1))$
$= \{$Förenkla alla inre uttryck som går$\}$
$= \sum_{i=0}^{n-1}(2+3(n-i)+3\frac{(n-1+i)(n-i)}{2}-3i(n-i)+3(n-i)) = \{$Förenkla alla inre uttryck som går$\}$
$= \sum_{i=0}^{n-1}(2+3n-3i+3\frac{n^2-n+i-i^2}{2}-3ni+3i^2+3n-3i) = \{$Förenkla alla inre uttryck som går$\}$
$= \sum_{i=0}^{n-1}(2+6n-6i+\frac{3n^2}{2}-\frac{3n}{2}+\frac{3i}{2}-\frac{3i^2}{2}-3ni+3i^2) = \{$Förenkla alla inre uttryck som går$\}$
$= \sum_{i=0}^{n-1}(2+\frac{9n}{2}-\frac{9i}{2}+\frac{3n^2}{2}+\frac{3i^2}{2}-3ni) = \{$Distribuera in den första summan$\}$
$= \sum_{i=0}^{n-1}2+\sum_{i=0}^{n-1}\frac{9n}{2}-\sum_{i=0}^{n-1}\frac{9i}{2}+\sum_{i=0}^{n-1}\frac{3n^2}{2}+\sum_{i=0}^{n-1}\frac{3i^2}{2}-\sum_{i=0}^{n-1}3ni = \{$Förenkla summorna$\}$
$= 2\sum_{i=0}^{n-1}1+\frac{9n}{2}\sum_{i=0}^{n-1}1-\frac{9}{2}\sum_{i=0}^{n-1}i+\frac{3n^2}{2}\sum_{i=0}^{n-1}1+\frac{3}{2}\sum_{i=0}^{n-1}i^2-3n\sum_{i=0}^{n-1}i =$ $\{$örenkla summorna med (övre-nedre+1) samt summationsregler$\}$
$= 2(n-1-0+1)+\frac{9n}{2}(n-1-0+1)-\frac{9}{2}\frac{n(n+1)}{2}+\frac{3n^2}{2}(n-1-0+1)+\frac{3}{2}\frac{n(n+1)(2n+1)}{6}-3n\frac{n(n+1)}{2} = \{$Förenkla$\}$
$= 2n+\frac{9n^2}{2}-\frac{9n^2+9n}{4}+\frac{3n^3}{2}+\frac{6n^3+9n^2+3n}{12}-\frac{3n^3+3n^2}{2} \in O(n^3)$

Algoritm 2:

I den första summan så utförs två aritmetiska operationer i "huvudet", detta leder till att vi får en tvåa i den första summan. Efter det så kommer det en till for-loop, vilket ger en ny summa inne i den första summan. I den andra for-loopen så utförs det två "vanliga" aritmetiska operationer i "huvudet", samt en addition och en jämförelse i kroppen. Detta ger att vi får en 4:a i den andra summan.

$\sum_{i=0}^{n-1}(2+\sum_{j=i}^{n-1}4) = \{\text{Förenkla den innersta summan}\}$
$= \sum_{i=0}^{n-1}(2+4\sum_{j=i}^{n-1}1) = \{\text{Utveckla med (övre-nedre+1)}\}$
$= \sum_{i=0}^{n-1}(2+4(n-1-i+1)) = \{\text{Förenkla den innersta summan}\}$
$= \sum_{i=0}^{n-1}(2+4(n-i)) = \{\text{Förenkla den innersta summan}\}$
$= \sum_{i=0}^{n-1}(2+4n-4i) = \{\text{Distribuera in den yttre summan}\}$
$= \sum_{i=0}^{n-1}2+\sum_{i=0}^{n-1}4n-\sum_{i=0}^{n-1}4i = \{\text{Förenkla med summationsregler}\}$
$= 2\sum_{i=0}^{n-1}1+4n\sum_{i=0}^{n-1}1-4\sum_{i=0}^{n-1}i = \{\text{Förenkla med summationsregler och (övre-nedre+1)}\}$
$= 2(n-1-0+1)+4n(n-1+0)-4\frac{n(n+1)}{2} = \{\text{Förenkla}\}$
$= 2n+4n^2-2n^2-2n = \{\text{Förenkla}\}$
$= 2n^2 \in O(n^2)$

Algoritm 3:

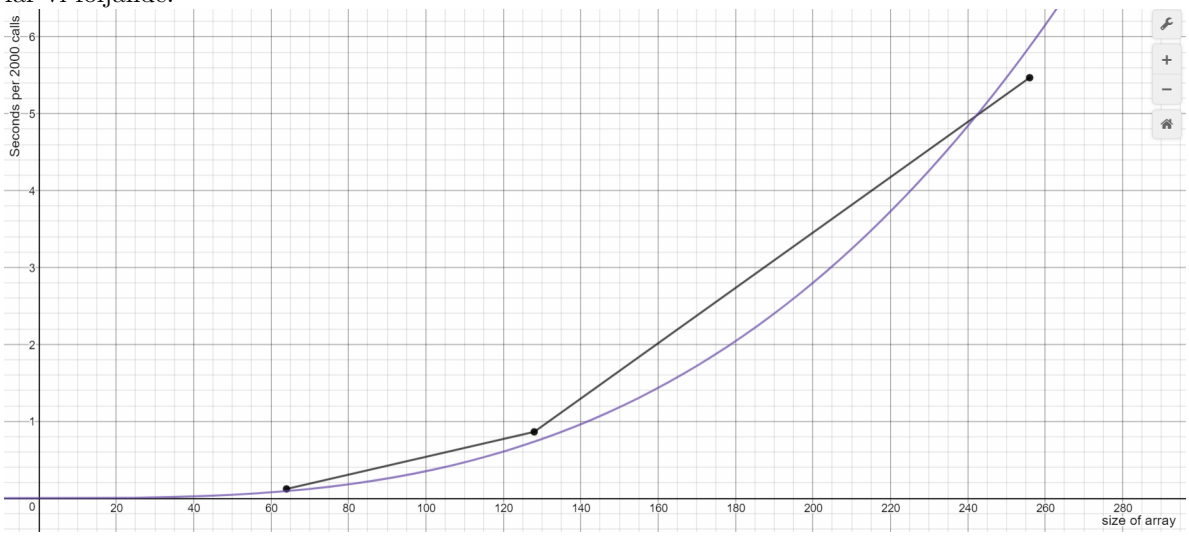
Denna algoritm bestor endast av en for-loop vilket ger oss endast en summa. Det utförs två aritmetiska operationer i "huvudet" samt en addition i kroppen. Men sedan så kommer ett if-else-if-block. Vi har antagit att det värsta fallet kommer att ske vilket skulle innebära att det utfördes en jämförelse samt ytterligare en addition. Detta ger oss, i kombination med de tidigare aritmetiska operationerna (3+2=5) att vi får en 5:a inne i summan.

$\sum_{i=0}^{n-1}5 = \{\text{Förenkla med summationsregler}\}$
$= 5\sum_{i=0}^{n-1}1 = \{\text{Förenkla med (övre-nedre+1)}\}$
$=5(n-1-0+1) = 5n \in O(n)$

1.2 b)

Algoritm 1:

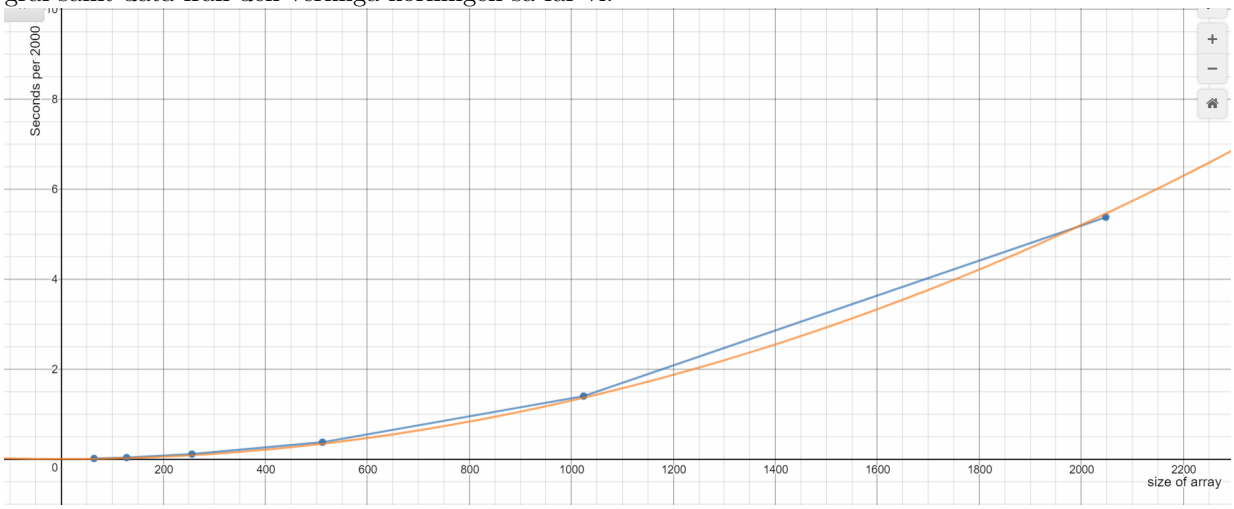
Från komplexitetsanalysen så har vi fått fram att algoritmen tillhör $O(n^3)$ om vi ritar ut en n^3-kurva samt sätter in data från när vi faktiskt körde algoritmen och fick fram värden på tidsåtgången och sedan ritar ut dessa i samma graf så får vi följande:



De svarta punkterna representerar de empiriskt uppmätta värdena, och den lila linjen visar en n^3-kura som är skalad i y-led för att passa så bra in på de uppmätta värdena som möjligt. Om vi tittar på linjerna så ser vi att de överlappar ganska bra, det är inte helt perfekt men det är inte särskilt förvånande. Eftersom tidsåtgången för en algoritm kan ändra sig beroende på t.ex. vilken hårdvara och vilken kompilator som man använder sig av. Men som sagt så verkar datan passa ganska bra in på n^3-kurvan, så vi kan anta att det är en tillräckligt bra representation av tidsåtgången för denna algoritm.
Vi kan också komma fram till att en n^3-kurva är en bra representation av tidsåtgången eftersom algoritmen innehåller tre stycken for-loopar. Detta borde leda till att algoritmen tillhör $O(n^3)$ eftersom varje enskild for-loop tillhör $O(n)$.

Algoritm 2:

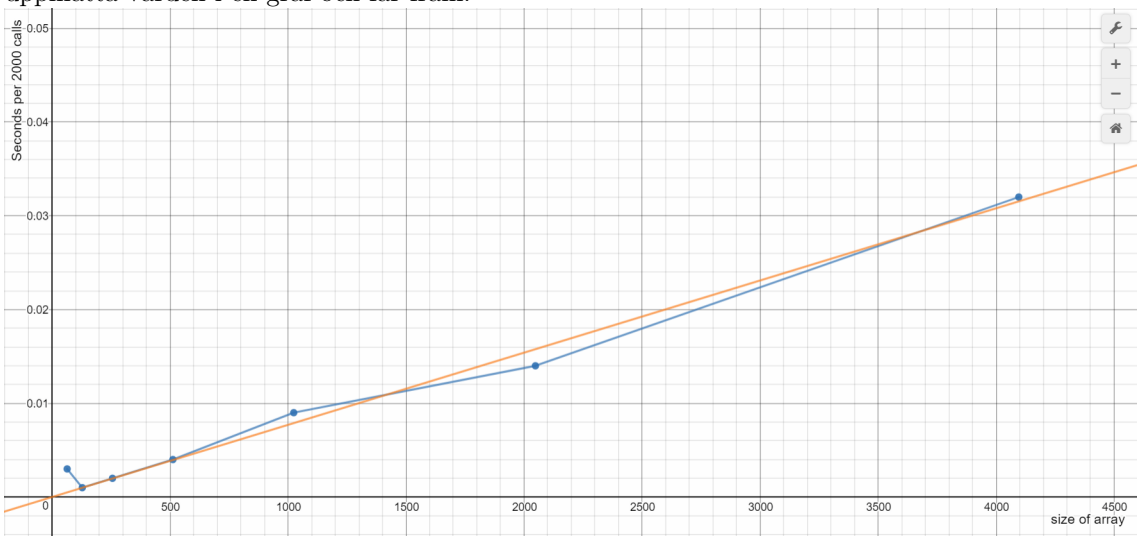
Algoritm 2 fick vi fram att den tillhör $O(n^2)$, om vi ritar ut denna kurva i en graf samt data från den verkliga körningen så får vi:



De blåa punkterna representerar de empiriskt uppmätta datan och den orangea linjen visar en n^2-kurva som är skalad i y-led för att passa in så bra som möjligt på de uppmätta värdena. Vi ser att de två linjerna överlappar ganska så bra, särskilt om man jämför med den föregående grafen. Så vi kan även här anta att n^2-kurvan modellerar tidsåtgången för algoritmen ganska bra.

Enligt samma resonemang som tidigare kan vi anta att en n^2-kurva representerar tidsåtgången bra då denna algoritm innehåller två stycken for-loopar. Detta skulle då leda till att algoritmen tillhör $O(n^2)$.

Algoritm 3:
Till sist så fick vi fram att algoritm 3 tillhör $O(n)$, vi ritar ut denna kurva samt uppmätta värden i en graf och får fram:



De blåa punkterna representerar de empiriskt uppmätta värdena och den orangea linjen visar en n-kurva (linje) som är skalad i y-led för att passa in på de uppmätta värdena så bra som möjligt. Här ser vi att punkterna följer linjen hyfsat, inte perfekt men tillräckligt för att anta att en n-kurva modellerar algoritmens tidsåtgång för olika indata.
Enligt tidigare så får vi fram att en n-kurva passar ganska bra för att modellera tidsåtgången, för denna algoritm har endast en for-loop, vilket borde göra så att algoritmen tillhör $O(n)$.
En intressant sak som vi ser från denna graf är att det tar längre tid för algoritm 3 att summera den första arrayen än den andra. Detta verkar väldigt orimligt då det intuitivt borde ta längre tid att summera en längre array än en mindre. Detta kan vara ett mindre fel beroende på andra faktorer än själva koden. Som tidigare nämnt så kan tidsåtgången för en algoritm ändras beroende på vilken hårdvara samt vilken kompilator m.m. som man använder. Det är dock fortfarande lite underligt att vi skulle få fram detta resultatet.