Introduktion till Gnuplot

Martin Pola
<mpola@kth.se>

8 mars 2019

Innehåll

Introduktion

Gnuplot är ett verktyg som kan användas för att skapa en uppsjö av olika sorters diagram. Med hjälp av olika kommandon kan man visualisera olika sorters data, företrädesvis mätvärden och andra resultat. Programmet finns förinstallerat på skolans Ubuntu-datorer, men om man vill ha det på sin egen dator kan man hämta hem det gratis från http://www.gnuplot.info/.

För att kontrollera om Gnuplot finns på sin dator kan man anropa det med `-version` som argument i en terminal.

```
[mpola@sport-04 ~]$ gnuplot --version
gnuplot 5.2 patchlevel 6
```

1 Lite data

Ett sätt att lära sig Gnuplot är att utgå från exempel. För att kunna göra det är det bra att ha lite data, så vi skriver en naiv algoritm för att räkna ut det n:te Fibonaccitalet. Algoritmen är naiv i avseendet att den hela tiden anropar funktionen `fibonacci` rekursivt, något som snabbt gör att det tal lång tid att beräkna ett svar. Här är vi dock inte intresserade av att skriva en effektiv algoritm, tvärtom! Har vi större variation i vår data får vi bättre underlag för att upptäckta de olika funktioner som Gnuplot erbjuder.

```
import timeit

def fibonacci(n):
   if n == 0: return 0
   if n == 1: return 1

   return fibonacci(n - 1) + fibonacci(n - 2)


# testkör fibonacci-funktionen
print('n', 'Fibonaccital', 'tidsåtgång')

for i in range(40):
   answer = fibonacci(i)
   time_spent = timeit.timeit(stmt = lambda: fibonacci(i), number = 1)

   print(i, answer, time_spent)
```

Vi ser till att inte bara skriva ut själva Fibonaccitalen, utan även hur lång tid det tog att räkna fram respektive tal. När man kör ovanstående program blir utskriften ungefär som följande.

```
[mpola@sport-04 ~]$ python3 fibonacci.py
#n Ficonaccital tidsåtgång
0 0 2.052000127150677e-06
1 1 1.7500001376902219e-06
2 1 2.069999936793465e-06
3 2 2.4390001271967776e-06
9 34 1.4976000329625094e-05
[...]
36 14930352 4.983006977000059
37 24157817 7.098211364000235
38 39088169 11.40919385899997
39 63245986 18.00078893900036
```

För att kunna visualisera vår data med Gnuplot är det lämpligt att spara resultatet i en textfil, i stället för att skriva ut den i terminalen. Det görs med `>`-teckenet och ett filnamn.

```
[mpola@sport-04 ~]$ python3 fibonacci.py > fibonacci.dat
```

Filändelsen, som i vårt fall är `.dat`, spelar ingen roll för Gnuplot. Det viktiga är att filens innehåll, som ju endast beror på vad Python-skriptet skriver ut, är formaterat på ett sätt som gör att Gnuplot kan läsa det. I det här fallet har vi en rad för varje värde på n, och varje rad innehåller n, Fibonaccitalet och tiden det tog att räkna ut det. De tre värdena är separerade med mellanslag.

2 Ett första diagram

Dags att rita upp vår data! För att göra det, skapa en ny fil `kommandofil.txt` med följande innehåll.

```
set terminal png
plot 'fibonacci.dat' using 1:2
```

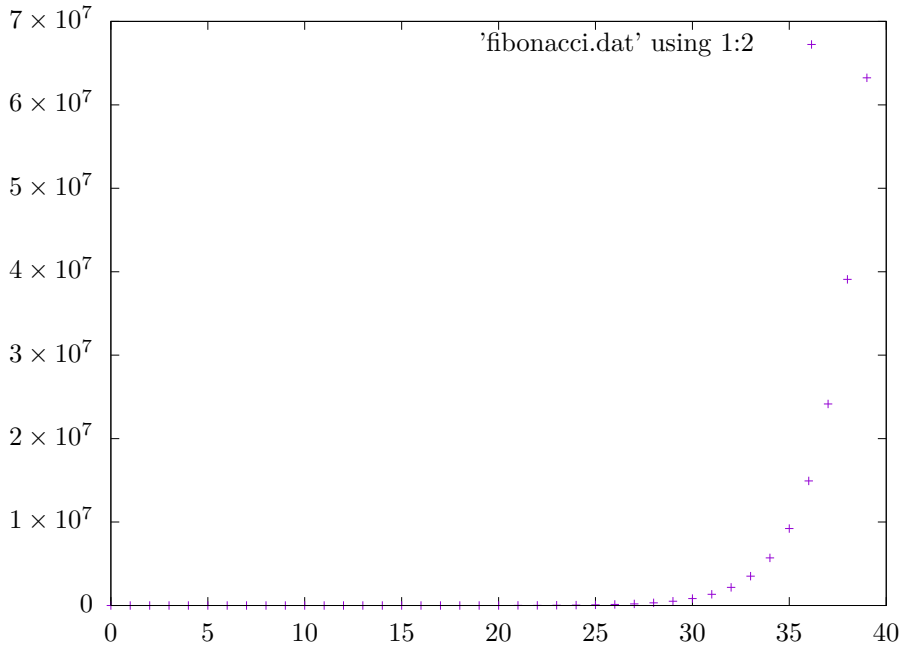
Spara filen och kör den sedan genom Gnuplot genom att skicka namnet på kommandofilen som argument:

```
[mpola@sport-04 ~]$ gnuplot kommandofil.txt
```

Troligtvis får du en helt oläslig utskrift tillbaka, eventuellt med vissa normala tecken på sina håll. Det du fick tillbaka var nämligen den PNG-bild som Gnuplot precis skapade. Gnuplot kan rita upp data i många olika format, men i det här exemplet – med den här kommandofilen – har vi sagt att vi vill ha en PNG-bild. För att spara bilden på datorn och därmed faktiskt kunna öppna den, använd åter igen `>` och ett filnamn.

```
[mpola@sport-04 ~]$ gnuplot kommandofil.txt > fibonacci.png
```

Gör du det ska du få fram en bild, `fibonacci.png`, som ser ut ungefär som följande.



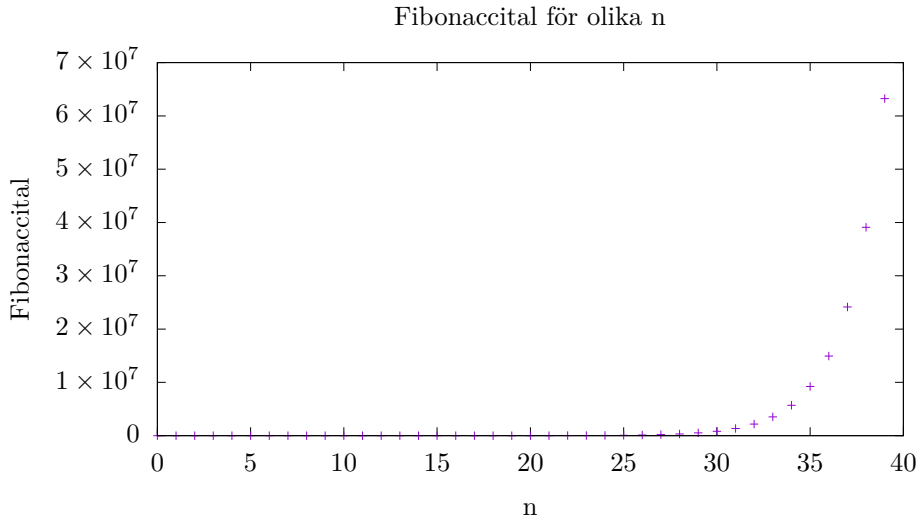
Om vi går tillbaka till kommandofilen så ska vi se vad vi faktiskt gjorde. Den första raden (`set terminal png`) angav att vi ville ha vårt diagram som en PNG-bild och den andra raden (`plot`) angav vad det var vi skulle rita ut. Vi valde att vi ville rita ut kolumn 1 mot kolumn 2. Gnuplot väntar sig att varje rad är en datapunkt och att varje rad har en kolumn 1 och en kolumn 2. Så är fallet hos oss, eftersom vi har n i första kolumnen och Fibonaccitalet i andra kolumnen. Dessutom har vi en tredje kolumn med tidsåtgången, som Gnuplot dock inte gjorde något med.

För att få bort den automatiska titeln på dataserien lägger vi till `notitle` i slutet av `plot`-kommandot. Dessutom noterar vi att diagrammet är lite för kvadratiskt; det hade kanske varit finare om det var lite plattare. Det kan åtgärdas med den nya raden `set size ratio 0.5`.

3 Titlar och axlar

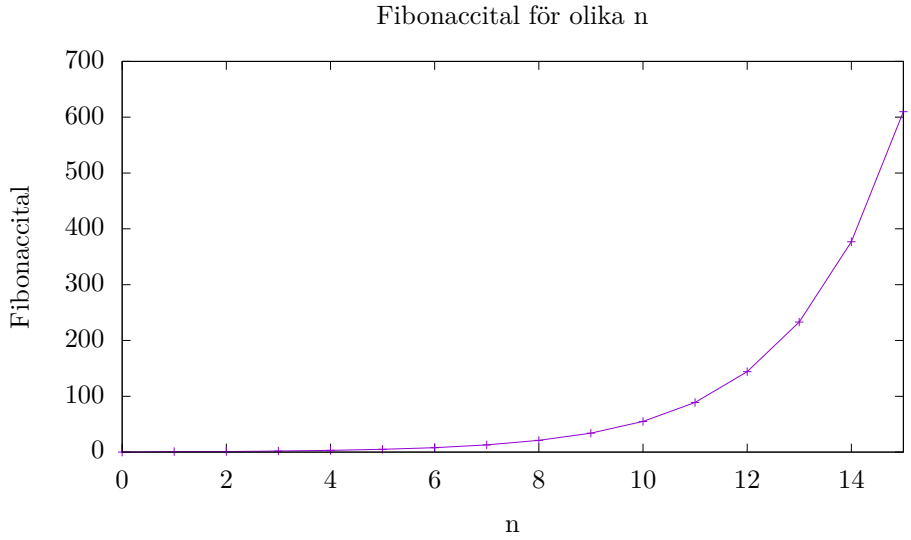
Ponera att vi kanske även vill ha en titel för själva diagrammet och en beskrivning av vad vi har på axlarna. Det kan vi också lägga in, med `set title` respektive `set xlabel` och `set ylabel`.

```
set size ratio 0.5
set title 'Fibonaccital för olika n'
set xlabel 'n'
set ylabel 'Fibonaccital'
plot 'fibonacci.dat' using 1:2 notitle
```



Eftersom Fibonaccitalen snabbt springer iväg i storlek blir det svårt att avläsa vilka värden talen för $n < 30$ har – de ligger alla inte långt över x-axeln. För att endast rita ut punkter för de mindre värdena på n kan man så klart ta bort de sista raderna från datafilen, men det är oftast inte optimalt. I en del fall vill man behålla alla resultat för att enklare kunna påvisa en trend, men samtidigt *zooma in* på intressanta delmängder av sin data. Vi använder oss av `set xrange` för att åstadkomma detta.

```
set size ratio 0.5
set title 'Fibonaccital för olika n'
set xlabel 'n'
set ylabel 'Fibonaccital'
set xrange [0:15]
set yrange [0:*]
plot 'fibonacci.dat' using 1:2 notitle with linespoints
```

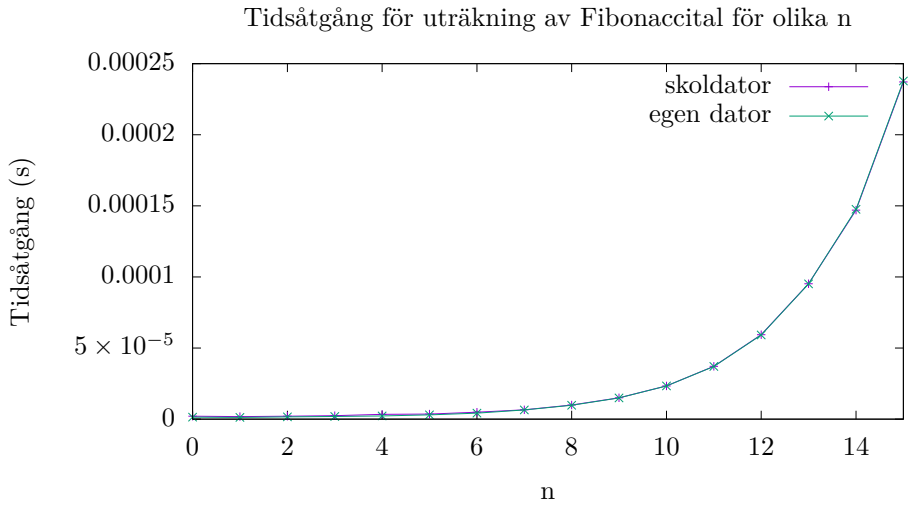


Utöver det sa vi åt Gnuplot att låta Y-axeln börja på 0, och sedan gå så långt som det behövdes (`set yrange`). Det hade vi inte behövt säga explicit, men om man exempelvis vill visa data endast på $y > 100$ kan man använda sig av det kommandot. Ytterligare en förändring är att vi sa att vi ville rita ut vår data med linjer och punkter (`with linespoints`), och inte bara med punkter. Det gör ofta att man betydligt enklare kan urskilja en trend, och om man inte vill ha punkterna alls kan man skriva `lines` i stället för `linespoints`.

4 Flera dataserier i samma diagram

I datafilen har kolumn 3 ännu inte använts till något. Det går att byta ut `using 1:2` mot `using 1:3` för att visualisera tidsåtgången i stället, men från ett Gnuplot-perspektiv bidrar det inte med något nytt. Däremot råkar det vara så att jag har kört samma Python-skript på min egen dator, och sparat resultatet i filen `fibonacci-egen-dator.dat`. De två första kolumnerna är identiska, men däremot var det inte samma tidsåtgång – kolumn 3 skiljer sig åt. För att jämföra tidsåtgången mellan körningarna av skriptet kan vi utöka `plot` med den data från den nya filen.

```
set size ratio 0.5
set title 'Tidsåtgång för uträkning av Fibonaccital för olika n'
set xlabel 'n'
set ylabel 'Tidsåtgång (s)'
set xrange [0:*]
set yrange [0:*]
plot 'fibonacci.dat' using 1:3 title 'skoldator' with linespoints, \
     'fibonacci-egen-dator.dat' using 1:3 title 'egen dator' with linespoints
```

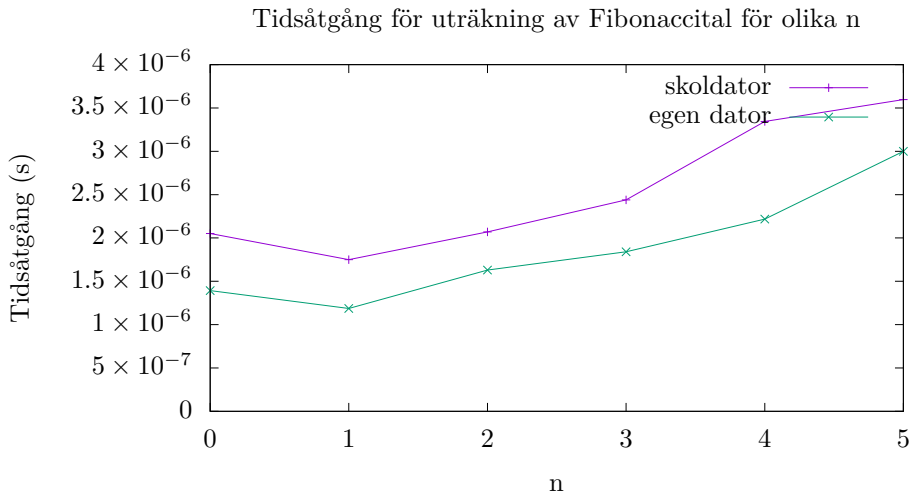


De två dataserierna separeras med ett kommatecken följt av ett bakvänt snedstreck. Det dock egentligen bara kommatecknet som separerar dataserier från varandra; det bakvända snedstrecket används för att berätta för Gnuplot att det aktuella kommandot, `plot`, fortsätter på nästa rad.

Som diagrammet avslöjar är min egen dator väldigt likvärdig skoldatorn, åtminstone när det kommer till tiden det tog att räkna ut Fibonaccitalen. Det finns dock en del skillnader, om än små, som man kan skönja genom att endast titta på $n = 0 \leq n \leq 5$. Det gör vi i exemplet nedan.

Utöver det kan man notera att argumenten till `plot`-kommandot gjorde att raden blev väldigt lång. Många av de vanligt förekommande orden i Gnuplot kan förkortas. I exemplet nedan åstadkommer `plot`-kommandot samma sak som ovan, men det är avsevärt kortare skrivet. Det kan vara bra att känna till om man sedan ger sig ut på webben för att hitta hur man gör specifika saker i Gnuplot; en del väldigt korta ord kan vara förkortningar för längre ord.

```
set size ratio 0.5
set title 'Tidsåtgång för uträkning av Fibonaccital för olika n'
set xlabel 'n'
set ylabel 'Tidsåtgång (s)'
set xrange [0:5]
set yrange [0:*]
plot 'fibonacci.dat' u 1:3 t 'skoldator' w lp, \
     'fibonacci-egen-dator.dat' u 1:3 t 'egen dator' w lp
```



5 Flera y-axlar

I våra datafiler har vi både Ficonaccital och tidsåtgången för att generera dem. Det är två värden som troligtvis har en korrelation, men eftersom de är i väsentligt olika magnituder – tidsåtgången är i många fall under en sekund medan Fibonaccitalen snabbt växer till många tusen – kan de inte direkt ritas upp bredvid varandra. I Gnuplot kan man lösa det genom att introducera ytterligare en y-axel, en `y2`. Med kommandona `set ytics` och `set y2label` förbereder vi den nya axeln, så att kolumner 1:3 kan ritas ut på den med tillägget `axes x1y2`. I diagrammet nedan blir det tydligt att de två dataserierna är korrelerade, trots att de faktiska värdena är långt ifrån varandra. Tidsåtgången, axel `y2`, lade sig till höger i bilden.

Slutligen drar vi nytta av att den första raden i vår datafil innehåller strängar som förklarar vad kolumnerna avser, och använder de strängarna direkt i vårt diagram (`columnheader`) för att särskilja dataserierna åt.

```
set size ratio 0.5
set title 'Tidsåtgång för uträkning av Fibonaccital för olika n'
set xlabel 'n'
set ylabel 'Fibonaccital'
set xrange [0:20]
set yrange [0:*]
set y2tics
set y2label 'Tidsåtgång (s)' rotate by 270
plot 'fibonacci.dat' u 1:2 t columnheader(2) w lp, \
     'fibonacci.dat' u 1:3 t columnheader(3) w lp axes x1y2
```