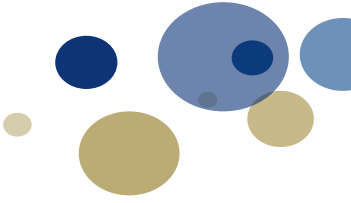
NTNU

Kunnskap for en bedre verden



Algoritmer og datastrukturer

Kompleksitetsanalyse og tidsmålinger

Helge Hafting

Institutt for datateknologi og informatikk

21. august 2018

Dataressurser

— Hvor mye tid, eller plass, går med for å kjøre programmet?
— Viktig når vi har store datamengder
— Avhenger av:
  - Problemet
  - Algoritmen vi bruker
  - Dataene
  - Datamaskinen

Kompleksitetsanalyse

— Finne en formel for mye tid (evt. plass) som går med for å kjøre en gitt algoritme
  - Uavhengig av maskintype!
  - Avhengig av antall data, n
  - Kan avhenge av hvordan dataene «ser ut»
  - Vi kan beregne verste, beste og/eller gjennomsnittlig kjøretid

Uavhengig av maskin?

— En rask maskin blir jo fortere ferdig enn en treg!
— Poenget er hvordan tidsforbruket øker med datamengde:
  - Vil 10× data ta 10× tid?
  - Vil 10×data ta 100× tid eller 1000× tid?
  - Vil det ikke være noen forskjell i hastighet?
— Når vi vet dette, kan vi regne ut tidsbruk på en gitt maskin ved å ta tiden på *ett* datasett.

Tidskompleksitet

— Omtrentlig antall «enkle instruksjoner» som utføres når algoritmen kjøres
  - Operasjoner som:
    — $+$, $-$, $\cdot$, $\div$
    — $<$, $>$, ==, !=, =
    — `if`-test, tabelloppslag
  - Ved metodekall må vi telle med operasjonene i metoden som kalles
  - Vi antar at alle slike operasjoner tar omtrent like lang tid
  - Antall operasjoner gir et tilnærmet uttrykk for hvor lang tid programmet tar
— Eksempler i læreboka, s. 9

Eksempel, tidskompleksitet

Programkode	Tid	
public static int sum(int []t, int n) {	0	deklarasjon
int sum = 0;		
for (int i = 0; i < n; ++i) {		
sum += t[i];		
}		
return sum;		
}		

Eksempel, tidskompleksitet

Programkode	Tid	
`public static int sum(int []t, int n) {`	0	deklarasjon
`  int sum = 0;`	1	tilordning
`  for (int i = 0; i < n; ++i) {`		
`  sum += t[i];`		
`  }`		
`  return sum;`		
`}`		

Eksempel, tidskompleksitet

Programkode	Tid	
`public static int sum(int []t, int n) {`		
`  int sum = 0;`	0	deklarasjon
`  for (int i = 0; i < n; ++i) {`	1	tilordning
`  sum += t[i];`	1+2n	1×tilordning, n×sammenl, n×inkr
`  }`		
`  return sum;`		
`}`		

Eksempel, tidskompleksitet

Programkode	Tid	
`public static int sum(int []t, int n) {`		
`  int sum = 0;`	0	deklarasjon
`  for (int i = 0; i < n; ++i) {`	1	tilordning
`  sum += t[i];`	1+2n	1×tilordning, n×sammenl, n×inkr
`  }`	2n	n addisjoner, n tabelloppslag
`  return sum;`		
`}`		

Eksempel, tidskompleksitet

Programkode	Tid	
`public static int sum(int []t, int n) {`	0	deklarasjon
`  int sum = 0;`	1	tilordning
`  for (int i = 0; i < n; ++i) {`	1+2n	1×tilordning, n×sammenl, n×inkr
`  sum += t[i];`	2n	n addisjoner, n tabelloppslag
`  }`	0	
`  return sum;`		
`}`		

Eksempel, tidskompleksitet

Programkode	Tid	
`public static int sum(int []t, int n) {`		
`  int sum = 0;`	0	deklarasjon
`  for (int i = 0; i < n; ++i) {`	1	tilordning
`  sum += t[i];`	1+2n	1×tilordning, n×sammenl, n×inkr
`  }`	2n	n addisjoner, n tabelloppslag
`  return sum;`	0	
`}`	1	1 return

Eksempel, tidskompleksitet

Programkode	Tid	
public static int sum(int []t, int n) {	0	deklarasjon
int sum = 0;	1	tilordning
for (int i = 0; i < n; ++i) {	1+2n	1×tilordning, n×sammenl, n×inkr
sum += t[i];	2n	n addisjoner, n tabelloppslag
}	0	
return sum;	1	1 return
}		

- Kjøretid 4n+3
- lineær avhengighet av n

Asymptotisk analyse

— Se på hva som skjer når datamengden går mot uendelig ($n \to \infty$).

— Asymptotisk analyse gjør analysearbeidet *enklere*

- vi teller bare løkker og metodekall
- heller enn hver eneste enkle operasjon

— Vi kan forenkle eller «runde av» kompliserte uttrykk som $8n^2 + 24n + 17$ til noe enklere, som n^2.

Eksempel, asymptotisk tidskompleksitet

Programkode	Tid

public static int sum(int []t, int n) {
  int sum = 0;
  for (int i = 0; i < n; ++i) {
  sum += t[i];
  }
  return sum;
}
	Skjer 1 gang

Eksempel, asymptotisk tidskompleksitet

Programkode	Tid
`public static int sum(int []t, int n) {`	
`  int sum = 0;`	Skjer 1 gang
`  for (int i = 0; i < n; ++i) {`	Skjer n ganger
`    sum += t[i];`	
`  }`	
`  return sum;`	
`}`	

Eksempel, asymptotisk tidskompleksitet

Programkode	Tid
`public static int sum(int []t, int n) {`	
`  int sum = 0;`	Skjer 1 gang
`  for (int i = 0; i < n; ++i) {`	Skjer n ganger
`    sum += t[i];`	
`  }`	
`  return sum;`	Mer av det som skjer 1 gang
`}`	

Eksempel, asymptotisk tidskompleksitet

Programkode	Tid
`public static int sum(int []t, int n) {`	
`  int sum = 0;`	Skjer 1 gang
`  for (int i = 0; i < n; ++i) {`	Skjer n ganger
`    sum += t[i];`	
`  }`	Mer av det som skjer 1 gang
`  return sum;`	
`}`	

— Kjøretid *n*

— lineær avhengighet av n

Praktisk betydning

— Program med lineær kjøretid, n
  - dobbel datamengde gir dobbel kjøretid
  - tidobbel datamengde gir tidobbel kjøretid
— Kvadratisk kjøretid, n^2
  - dobbel datamengde: $2^2 = 4$ ganger kjøretid
  - tidobbel datamengde: $10^2 = 100$ ganger kjøretid
— Kubisk kjøretid, n^3
  - dobbel datamengde: $2^3 = 8$ ganger kjøretid
  - tidobbel datamengde: $10^3 = 1000$ ganger kjøretid

Asymptotisk analyse, matematikken bak

— O, den øvre grensen, «verre blir det ikke»

- $f(n) \in O(g(n)) \Rightarrow g(n)$ er en øvre grense for $f(n)$
- Eks: $f(n) = n^2 - 2n + 4 \in O(n^2)$. Her er n^2 øvre grense
- Se side 11 i læreboka

— Ω, den nedre grensen, «ikke bedre enn dette»

- $f(n) \in \Omega(g(n)) \Rightarrow g(n)$ er en nedre grense for $f(n)$
- Se side 12

— $\Theta(n)$, felles øvre og nedre grense.

- Brukes når O og Ω har samme uttrykk.
- Se side 13

Asymptotisk analyse

— Vanlige kompleksiteter:
  - $O(1)$, $O(n)$, $O(n^2)$, $O(n^3)$, $O(n^4)$, $O(n^5)$, ...
  - $O(\log n)$, $O(n \log n)$, $O(2^n)$
— Korte greie uttrykk, som er lette å sammenligne.
— Vi kan finne ut hvilke programmer som er raskest på store datamengder

Måleteknikk

— Bruk minst 3, men helst 4–5 ulike verdier for n.
— Varier n systematisk, f.eks. $n = 10\,000$, $n = 33\,333$, $n = 100\,000$
— Kjør målinger flere ganger, unngå avvik
— Sett opp tabell som viser tidsforbruk ved ulike n
— Ikke ha utskrift/grafikk/filbehandling inni målingen!
— For små n-verdier kan gi avvik. Vi undersøker hva som skjer når $n \to \infty$.
— For stor n kan gi avvik i form av swapping
— Annen programvare kan forstyrre...
— Maskinklokka har ikke høy nok oppløsning for svært korte kjøringer

Eksempel 1

n	Tid
100	0,00525 ms
1000	0,0249 ms
10 000	0,197 ms
100 000	1,94 ms

Eksempel 2

n	Tid
100	0,037 ms
1000	0,45 ms
10 000	44 ms
100 000	4,3 s

Håndtere korte kjøringer

```
Date start = new Date();
int runder = 0;
double tid;
Date slutt;
do {
  r = testdata.algoritme();
  slutt = new Date();
  ++runder;
} while (slutt.getTime()-start.getTime() < 1000);
tid = (double)
      (slutt.getTime()-start.getTime()) / runder;
System.out.println("Millisekund pr. runde:" + tid);
```

Tips om objektorientering

— Ikke gi *klassen* navn etter algoritmen
— Algoritmer implementeres som *metoder*
  - metoden får navn etter algoritmen den implementerer
— Klasser får navn etter hva de inneholder av *data*, ikke etter hva vi kan gjøre med dem.