

:: Forside

Motivasjon

Analyse

Θ-notasjon

O og Ω

Relasjoner

Klasser

Fallgruver

Spørsmål

Kompleksitetsanalyse

Åsmund Eldhuset

[asmunde *at* stud.ntnu.no](mailto:asmunde*at*stud.ntnu.no)

www.stud.ntnu.no/~asmunde/kompleksitetsanalyse.ppt

Motivasjon

- Vi ønsker å måle hvor rask en algoritme er
- Lite mening i å måle kjøretid i sekunder
 - Forskjeller i programmeringsspråk og maskinvare
 - Kjøretiden kan variere drastisk med dataene algoritmen blir brukt på
- Datamaskiner blir kraftigere, og algoritmer vil bli brukt på større data
- Hvordan ser kjøretiden ut når inputstørrelsen nærmer seg uendelig?

Analyse

- Vi trenger et matematisk uttrykk for hva kjøretiden blir, gitt inputstørrelsen
- Vi analyserer trinnene i algoritmen
- ```
for (i = 0; i < n; i ++)
 doSomething();
for (i = 0; i < n; i ++)
 for (j = i; j < n; j ++)
 doSomethingElse();
```
- Kjøretid:  $f(n) = c_1 \cdot n^2 / 2 + c_2 \cdot n$

# Største ledd

- $c_1$  og  $c_2$  kommer an på maskinvaren
- La oss anta at  $c_1 = 1$  og  $c_2 = 100$
- Når bare  $n$  blir stor nok ( $n > 200$ ), blir  $n^2 / 2$  større enn  $100n$ , og når  $n$  vokser mer og mer, blir  $n^2 / 2$  MYE større enn  $100n$
- Kun det største leddet ( $n^2$ ) er interessant, fordi dette leddet vil dominere de andre når  $n$  blir stor nok

# Θ-notasjon

- Hvordan uttrykke dette?
- Vi sier at  $n^2 / 2 + 100n = \Theta(n^2)$
- Viktig:  $\Theta$  er en *mengde*!  $\Theta(n^2)$  består av alle funksjoner som vokser "like raskt" som  $n^2$  når  $n$  går mot uendelig
- Likhetstegnet misbrukes – det skulle egentlig ha vært  $f(n) \in \Theta(g(n))$
- $f(n) = \Theta(g(n))$  betyr altså: "f(n) er en av de funksjonene som vokser like raskt som  $g(n)$ "

# Definisjon

- $\Theta(g(n)) = \{f(n) : \text{det finnes positive konstanter } c_1, c_2 \text{ og } n_0 \text{ slik at } 0 \leq c_1g(n) \leq f(n) \leq c_2g(n) \text{ for alle } n \geq n_0\}$
- Altså:  $\Theta(g(n))$  består av alle funksjoner som  $g(n)$  kan brukes som både øvre og nedre grense for
- Hvis vi skal vise at  $f(n) = \Theta(g(n))$ , må vi bevise ulikhetene og finne  $c_1, c_2$  og  $n_0$

# Eksempel – polynomer

- Gitt  $f(n) = n^3 - 100n^2 + n$
- Vi trikser med leddene:
  - $f(n) = n^3 - 100n^2 + n \leq n^3 + n \leq n^3 + n^3 = 2n^3$
  - $f(n) = n^3 - 100n^2 + n \geq n^3 - n^3 / 2 + n \geq n^3 / 2$
- Vi finner ut når overgangene gjelder:
  - $-100n^2 \leq 0$  : uansett
  - $n \leq n^3$  : for  $n \geq 1$
  - $-100n^2 \geq -n^3 / 2$  : for  $n \geq 200$
  - $n \geq 0$  : for  $n \geq 0$
- Altså kan vi sette  $n_0 = 200$

# Eksempel – polynomer

- Nå vet vi at  $0 \leq n^3 / 2 \leq f(n) \leq 2n^3$   
når  $n > n_0 = 200$
- Vi kan nå sette  $c_1 = 1/2$ ,  $c_2 = 2$  og  $g(n) = n^3$
- Kravene for at  $f(n) = \Theta(g(n))$  er oppfylt
- Altså er  $n^3 - 100n^2 + n = \Theta(n^3)$

Forside

Motivasjon

Analyse

:: Θ-notasjon

O og Ω

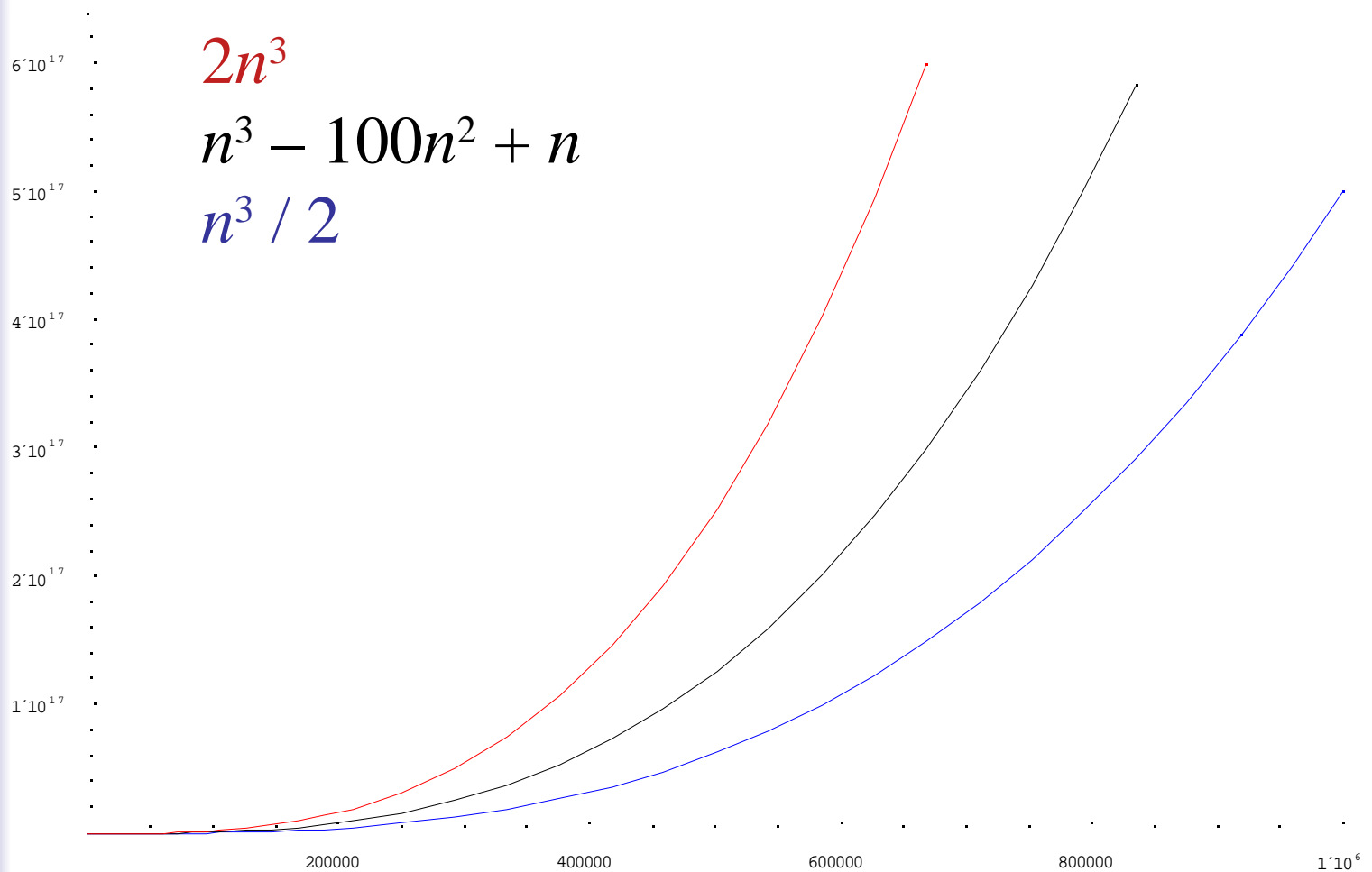
Relasjoner

Klasser

Fallgruver

Spørsmål

# Grafisk fremstilling



# O- og Ω-notasjon

- $f(n) = \Theta(g(n))$  sier at  $g$  er både en øvre og en nedre grense for  $f$
- Hvis vi bare vil si at  $g$  er en øvre grense for  $f$ , bruker vi O-notasjon:  $f(n) = O(g(n))$
- Hvis vi bare vil si at  $g$  er en nedre grense for  $f$ , bruker vi Ω-notasjon:  $f(n) = \Omega(g(n))$
- Disse grensene er ikke tette: f.eks er  $n = O(2^n)$ , og  $n^{100} = \Omega(n)$
- Se Cormen kap. 3.1 for definisjoner. Regning med O og Ω er helt tilsvarende det vi gjorde med Θ

# Relasjoner mellom $O$ , $\Omega$ og $\Theta$

- $f(n) = O(g(n)) \Leftrightarrow g(n) = \Omega(f(n))$
- $f(n) = \Theta(g(n)) \Leftrightarrow f(n) = O(g(n)) \wedge f(n) = \Omega(g(n))$
- $f(n) = \Theta(g(n)) \Leftrightarrow g(n) = \Theta(f(n))$   
 $\Leftrightarrow \Theta(f(n)) = \Theta(g(n))$
- $\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$
- Se side 49 i Cormen for nyttige relasjoner
- Prøv å bevise disse selv! Bruk definisjonene av  $O$  (side 44),  $\Omega$  (side 45) og  $\Theta$  (side 42)

# Kompleksitetsklasser

- Grovinndeling i stigende rekkefølge:
  - Konstant: 1
  - Polylogaritmisk:  $(\log_b n)^k$
  - Polynomisk:  $n^k$
  - Eksponentiell:  $2^n$
  - Faktoriell:  $n!$
  - Alt som er enda verre, f.eks.  $n^n$

# Kompleksitetsklasser

- Eksempler – alle på samme linje er "like raske" (de er  $\Theta$  av hverandre)
  - $1, 2, e, 1000000$
  - $\lg n, \ln n, \log_{10} n, \lg n^{100}$
  - $(\lg n)^2, (\ln n)^2, (\log_{10} n)^2, (\lg n^{100})^2$
  - $n, n + 1000 \lg n$
  - $n \lg n, n \log_{42} n$
  - $n^2$
  - $2^n$
  - $3^n$
  - $n!$
  - $n^n$

# Fallgruver

- Grunntallet til en logaritme har *ikke* noe å si for kompleksiteten:  $\lg_a n = \Theta(\lg_b n)$
- Grunntallet til en potens har *mye* å si for kompleksiteten:  $a^n \neq \Theta(b^n)$
- Er  $2^{n+c} = \Theta(2^n)$ ? Er  $2^{cn} = \Theta(2^n)$ ?
- Vær forsiktig med transitivitet:
  - Følgende er korrekt:  
$$f(n) = O(g(n)) \wedge g(n) = O(h(n)) \Rightarrow f(n) = O(h(n))$$
  - Men kan vi slutte noe av det følgende?  
$$f(n) = O(g(n)) \wedge g(n) = \Omega(h(n))$$

# Fallgruver

- Ikke alle funksjoner kan sammenlignes med  $O$ ,  $\Omega$  og  $\Theta$ 
  - Eksempel:  $n$  og  $n^{1 + \sin n}$
  - $1 + \sin n$  oscillerer mellom 0 og 2, så  $n^{1 + \sin n}$  veksler hele tiden mellom å være større enn og mindre enn  $n$
- $O$  angir en mengde funksjoner
  - Det er ikke i seg selv et mål på worst case-kjøretid, selv om det ofte brukes til å angi dette
  - Å si "kjøretiden er minst  $O(n)$ " er meningsløst
  - Det samme gjelder for  $\Omega$  og best case-kjøretid

# Pseudopolynomialitet

- Gitt en algoritme som tar tallet  $n$  som input og har kjøretid  $O(n)$  – hvilken kompleksitetsklasse er dette?
  - $n$  betegner her ikke størrelsen på input, men er selv en del av inputen
  - Størrelsen til  $n =$  antall bits som kreves  $= \lg n$
  - $n = 2^{\lg n} = 2^w$
  - Oi sann – eksponentiell kjøretid!
  - Dukker ofte opp som lureoppgave på eksamen!

Forside

Motivasjon

Analyse

Θ-notasjon

O og  $\Omega$

Relasjoner

Klasser

Fallgruver

:: Spørsmål

# Spørsmål?

---