



# Studiepraktik 2016

## Algoritmer og problemløsning

Arinbjörn Brandsson  
Benjamin Rotendahl  
Mathias Fleig Mortensen  
Christopher Mulvad Groot



- ❶ Dagens program
- ❷ Introduktion til algoritmer
- ❸ Algoritme vs Algoritme
- ❹ Algoritme design



# Program for idag

Indtil kl 11.45

- Algoritme design og metoder.



# Program for idag

Indtil kl 11.45

- Algoritme design og metoder.
- Hvordan man kan sammenligne forskellige løsninger.



# Program for idag

## Indtil kl 11.45

- Algoritme design og metoder.
- Hvordan man kan sammenligne forskellige løsninger.
- Hvad er grænserne for algoritmer?



# Program for idag

Indtil kl 11.45

- Algoritme design og metoder.
- Hvordan man kan sammenligne forskellige løsninger.
- Hvad er grænserne for algoritmer? **Historie tid!**



# Program for idag

## Indtil kl 11.45

- Algoritme design og metoder.
- Hvordan man kan sammenligne forskellige løsninger.
- Hvad er grænserne for algoritmer? **Historie tid!**

## Fra 13 til 14.30

- Øvelser i algoritmer



# Program for idag

## Indtil kl 11.45

- Algoritme design og metoder.
- Hvordan man kan sammenligne forskellige løsninger.
- Hvad er grænserne for algoritmer? **Historie tid!**

## Fra 13 til 14.30

- Øvelser i algoritmer
- Sjove gåder





# Program for idag

## Indtil kl 11.45

- Algoritme design og metoder.
- Hvordan man kan sammenligne forskellige løsninger.
- Hvad er grænserne for algoritmer? **Historie tid!**

## Fra 13 til 14.30

- Øvelser i algoritmer
- Sjove gåder
- Opsamling og spørgsmål



# Hvad er en Algoritme?

*An algorithm is a self-contained step-by-step set of operations to be performed that can be expressed within a finite amount of space and time and in a well-defined formal language.*



# Hvad er en Algoritme?

*An algorithm is a self-contained step-by-step set of operations to be performed that can be expressed within a finite amount of space and time and in a well-defined formal language.*

## På dansk

En algoritme er en **opskrift** på hvordan et bestemt problem kan løses.



# Havregryns algoritme

## Eksempel

### Algorithm 1

Indgangsbetingelser: En skål, mælk, havregryn

Udgangsbetingelser: Morgenmad

**while** Skålen ikke er fyldt **do**

    Hæld Gryn i Skålen

**end while**

**if** Jeg er tyk **then**

    mælk = Minimælk

**else**

    mælk = Letmælk

**end if**

**while** Skålen ikke er fyldt **do**

    Hæld mælk i Skålen

**end while**

# Krav til en algoritme?

## Krav til en algoritme

**Veldefineret** Ingen tvetydigheder og vendinger som: “ så tager du det bedste resultat ... ”



# Krav til en algoritme?

## Krav til en algoritme

**Veldefineret** Ingen tvetydigheder og vendinger som: “ så tager du det bedste resultat ... ”

**Terminerer** Den må ikke køre for evigt, du skal garantere at den rent faktisk finder sit svar.



# Krav til en algoritme?

## Krav til en algoritme

**Veldefineret** Ingen tvetydigheder og vendinger som: “ så tager du det bedste resultat ... ”

**Terminerer** Den må ikke køre for evigt, du skal garantere at den rent faktisk finder sit svar.

**input og output** Jeg skal vide at hvis jeg giver den  $A$  så returnerer den  $B$ .



# Krav til en algoritme?

## Krav til en algoritme

**Veldefineret** Ingen tvetydigheder og vendinger som: “ så tager du det bedste resultat ... ”

**Terminerer** Den må ikke køre for evigt, du skal garantere at den rent faktisk finder sit svar.

**input og output** Jeg skal vide at hvis jeg giver den  $A$  så returnerer den  $B$ .

**Kan bevises** Det er muligt både at bevise korrekthed og køretid for algoritmen.





# Eksempler på algoritmer

Algoritmer bruges inden for alle former for problemløsnings.



# Eksempler på algoritmer

Algoritmer bruges inden for alle former for problemløsnings.

## Bioinformatik

*Longest common subsequence:* Sammenligning af DNA strenge for at se hvor beslægtet to strenge er.



# Eksempler på algoritmer

Algoritmer bruges inden for alle former for problemløsnings.

## Bioinformatik

*Longest common subsequence:* Sammenligning af DNA strenge for at se hvor beslægtet to strenge er.

## Primtals faktorisering

Bruges i *kryptering*: Basis for at vi kan have sikker kommunikation.



# Eksempler på algoritmer

Algoritmer bruges inden for alle former for problemløsning.

## Bioinformatik

*Longest common subsequence:* Sammenligning af DNA strenge for at se hvor beslægtet to strenge er.

## Primtals faktorisering

Bruges i *kryptering*: Basis for at vi kan have sikker kommunikation.

## Machine Learning

En samling af algoritmer der selv kan lære og finde egenskaber i store data sæt. Gør det muligt at løse problemer der før var uden for menneskers kunnen.



# Valget af algoritme

## Soterings algoritmer

Givet en liste af  $n$  tal ønsker vi at returnere en sorteret liste af længde  $n$ .



# Valget af algoritme

## Soterings algoritmer

Givet en liste af  $n$  tal ønsker vi at returnere en sorteret liste af længde  $n$ .

### Hold A

- Har en computer

### Hold B



# Valget af algoritme

## Soterings algoritmer

Givet en liste af  $n$  tal ønsker vi at returnere en sorteret liste af længde  $n$ .

### Hold A

- Har en computer
- Bruger algoritmen  
*Merge Sort*

### Hold B



# Valget af algoritme

## Soterings algoritmer

Givet en liste af  $n$  tal ønsker vi at returnere en sorteret liste af længde  $n$ .

### Hold A

- Har en computer
- Bruger algoritmen *Merge Sort*
- De kan sortere 10 millioner tal på under 20 minutter

### Hold B





# Valget af algoritme

## Soterings algoritmer

Givet en liste af  $n$  tal ønsker vi at returnere en sorteret liste af længde  $n$ .

### Hold A

- Har en computer
- Bruger algoritmen *Merge Sort*
- De kan sortere 10 millioner tal på under 20 minutter
- De kan sortere 100 millioner tal på 4 timer.

### Hold B



# Valget af algoritme

## Soterings algoritmer

Givet en liste af  $n$  tal ønsker vi at returnere en sorteret liste af længde  $n$ .

### Hold A

- Har en computer
- Bruger algoritmen *Merge Sort*
- De kan sortere 10 millioner tal på under 20 minutter
- De kan sortere 100 millioner tal på 4 timer.

### Hold B

- Har en computer der er 1000 gange hurtigere end hold A



# Valget af algoritme

## Soterings algoritmer

Givet en liste af  $n$  tal ønsker vi at returnere en sorteret liste af længde  $n$ .

### Hold A

- Har en computer
- Bruger algoritmen *Merge Sort*
- De kan sortere 10 millioner tal på under 20 minutter
- De kan sortere 100 millioner tal på 4 timer.

### Hold B

- Har en computer der er 1000 gange hurtigere end hold A
- Bruger algoritmen *Insertion Sort*



# Valget af algoritme

## Soterings algoritmer

Givet en liste af  $n$  tal ønsker vi at returnere en sorteret liste af længde  $n$ .

### Hold A

- Har en computer
- Bruger algoritmen *Merge Sort*
- De kan sortere 10 millioner tal på under 20 minutter
- De kan sortere 100 millioner tal på 4 timer.

### Hold B

- Har en computer der er 1000 gange hurtigere end hold A
- Bruger algoritmen *Insertion Sort*
- De kan sortere 10 millioner tal på 5 timer.



# Valget af algoritme

## Soterings algoritmer

Givet en liste af  $n$  tal ønsker vi at returnere en sorteret liste af længde  $n$ .

### Hold A

- Har en computer
- Bruger algoritmen *Merge Sort*
- De kan sortere 10 millioner tal på under 20 minutter
- De kan sortere 100 millioner tal på 4 timer.

### Hold B

- Har en computer der er 1000 gange hurtigere end hold A
- Bruger algoritmen *Insertion Sort*
- De kan sortere 10 millioner tal på 5 timer.
- De kan sortere 100 millioner tal på **23 dage!**



# Sammenligning af Algoritmer

*Vi bruger begrebet Køretid for at beskrive hvordan tiden en algoritme bruger stiger med input.*



# Sammenligning af Algoritmer

*Vi bruger begrebet Køretid for at beskrive hvordan tiden en algoritme bruger stiger med input.*

## Definition på køretid

En øvregrænse for den tid der bliver brugt på at løse et problem af størrelse  $n$ . Skrives som

$$O(n), O(n^2), O(n \lg n), O(n!), O\left(\frac{a}{b}\right)$$



# Sammenligning af Algoritmer

*Vi bruger begrebet Køretid for at beskrive hvordan tiden en algoritme bruger stiger med input.*

## Definition på køretid

En øvregrænse for den tid der bliver brugt på at løse et problem af størrelse  $n$ . Skrives som

$$O(n), O(n^2), O(n \lg n), O(n!), O\left(\frac{a}{b}\right)$$

## Algoritme for minimums funktionen

Givet en liste  $X = [x_1, x_2, \dots, x_n]$  ønsker vi at returnere det mindste tal i listen. Hvad er algoritmen og hvad er køretiden?





# Minimums algoritme

## Eksempel

### Algorithm 2

Input: En liste  $X = [x_1, x_2, \dots, x_n]$

Ouput: Det mindste tal i listen.

---

$\text{min} = x_1$

**for**  $x_i$  in  $X$  **do**

**if**  $x_i < \text{min}$  **then**

$\text{min} = x_i$

**end if**

**end for**

---



# Minimums algoritme

## Eksempel

### Algorithm 3

Input: En liste  $X = [x_1, x_2, \dots, x_n]$

Output: Det mindste tal i listen.

---

$\text{min} = x_1$

**for**  $x_i$  in  $X$  **do**

**if**  $x_i < \text{min}$  **then**

$\text{min} = x_i$

**end if**

**end for**

---

## Analyse af algoritmen

Køretid?



# Minimums algoritme

## Eksempel

### Algorithm 4

Input: En liste  $X = [x_1, x_2, \dots, x_n]$

Output: Det mindste tal i listen.

---

$\text{min} = x_1$

**for**  $x_i$  in  $X$  **do**

**if**  $x_i < \text{min}$  **then**

$\text{min} = x_i$

**end if**

**end for**

---

## Analyse af algoritmen

Køretid?  $O(n)$



# Minimums algoritme

## Eksempel

### Algorithm 5

Input: En liste  $X = [x_1, x_2, \dots, x_n]$

Output: Det mindste tal i listen.

---

$\text{min} = x_1$

**for**  $x_i$  in  $X$  **do**

**if**  $x_i < \text{min}$  **then**

$\text{min} = x_i$

**end if**

**end for**

---

## Analyse af algoritmen

Køretid?  $O(n)$

Er den optimal?



# Minimums algoritme

## Eksempel

### Algorithm 6

Input: En liste  $X = [x_1, x_2, \dots, x_n]$

Output: Det mindste tal i listen.

---

$\text{min} = x_1$

**for**  $x_i$  in  $X$  **do**

**if**  $x_i < \text{min}$  **then**

$\text{min} = x_i$

**end if**

**end for**

---

## Analyse af algoritmen

Køretid?  $O(n)$

Er den optimal? **Jeps!**



# Eksempler på køretid

## Bogo Sort

- Køretid på  $O(n!)$

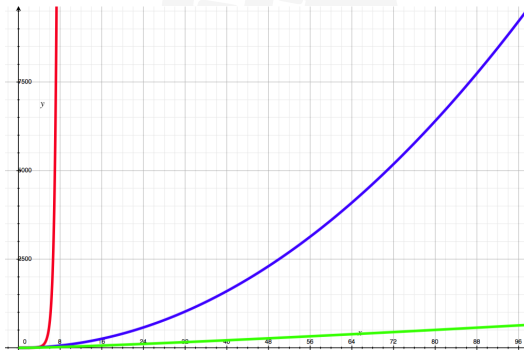
## Insertion Sort

- Køretid?  $O(n^2)$

## Merge sort

- Køretid på  $O(n \lg n)$

Figure: Graf over køretider



# Hvordan finder man på en algoritme

## Algoritme for algoritmer

- 1 Beskriv problemet med egne ord.



# Hvordan finder man på en algoritme

## Algoritme for algoritmer

- 1 Beskriv problemet med egne ord.
- 2 Del problemet op i mindre dele.





# Hvordan finder man på en algoritme

## Algoritme for algoritmer

- 1 Beskriv problemet med egne ord.
- 2 Del problemet op i mindre dele.
- 3 Definer output



# Hvordan finder man på en algoritme

## Algoritme for algoritmer

- 1 Beskriv problemet med egne ord.
- 2 Del problemet op i mindre dele.
- 3 Definer output
- 4 Definer input



# Hvordan finder man på en algoritme

## Algoritme for algoritmer

- 1 Beskriv problemet med egne ord.
- 2 Del problemet op i mindre dele.
- 3 Definer output
- 4 Definer input
- 5 Beskriv trin for at gå fra input til output



# Så er det historie tid

