Grundlæggende Algoritmer og Datastrukturer

Om kurset…

Grundlæggende Algoritmer og Datastrukturer

Undervisningsformer
Forelæsninger: 4 timer/uge (2+2). Øvelser: 3 timer/uge. Café.

Obligatorisk program
13 teoretisk[illegible]gramerings opgaver

E
2 tim[illegible]

Sprog
Dansk

Forelæs[illegible]
stoffet fra b[illegible]
arbejder

De teoretiske opgaver skal alle være godkendt for at kunne gå til eksamen. Opgaverne afleveres i grupper af max 3 personer. Programmeringsopgaverne tæller med til den endelige karakter.

Forelæs[illegible]
mat[illegible]
dans[illegible]

Eksamen består af multiple choice og små skriftlige opgaver i stil med de ugentlige opgaver

TØ timer (øvelser)

3 timer om ugen med en instruktor.

En række teoretiske opgaver til hver gang.

Format:

1. I forbereder jer hjemmefra og forsøger at forstå og løse alle opgaverne inden i kommer.
   Brug studiecaféen til at få hjælp.
2. Ved øvelserne gennemgåes opgaverne af de studerende, imens instruktoren hjælper med at rette misforståelser og fremhæve vigtige pointer.

Hvilke opgaver: Opgaverne på “Course Plan” som hører til datoer *efter* jeres sidste TØ time, og *op til* denne TØ time.

Afleveringsopgaver

Vil fremgå af Blackboard

Øvelsesholdet aftaler afleveringsfrist med instruktoren

Afleveres via. Blackboard

Teoretiske afleveringsopgaver

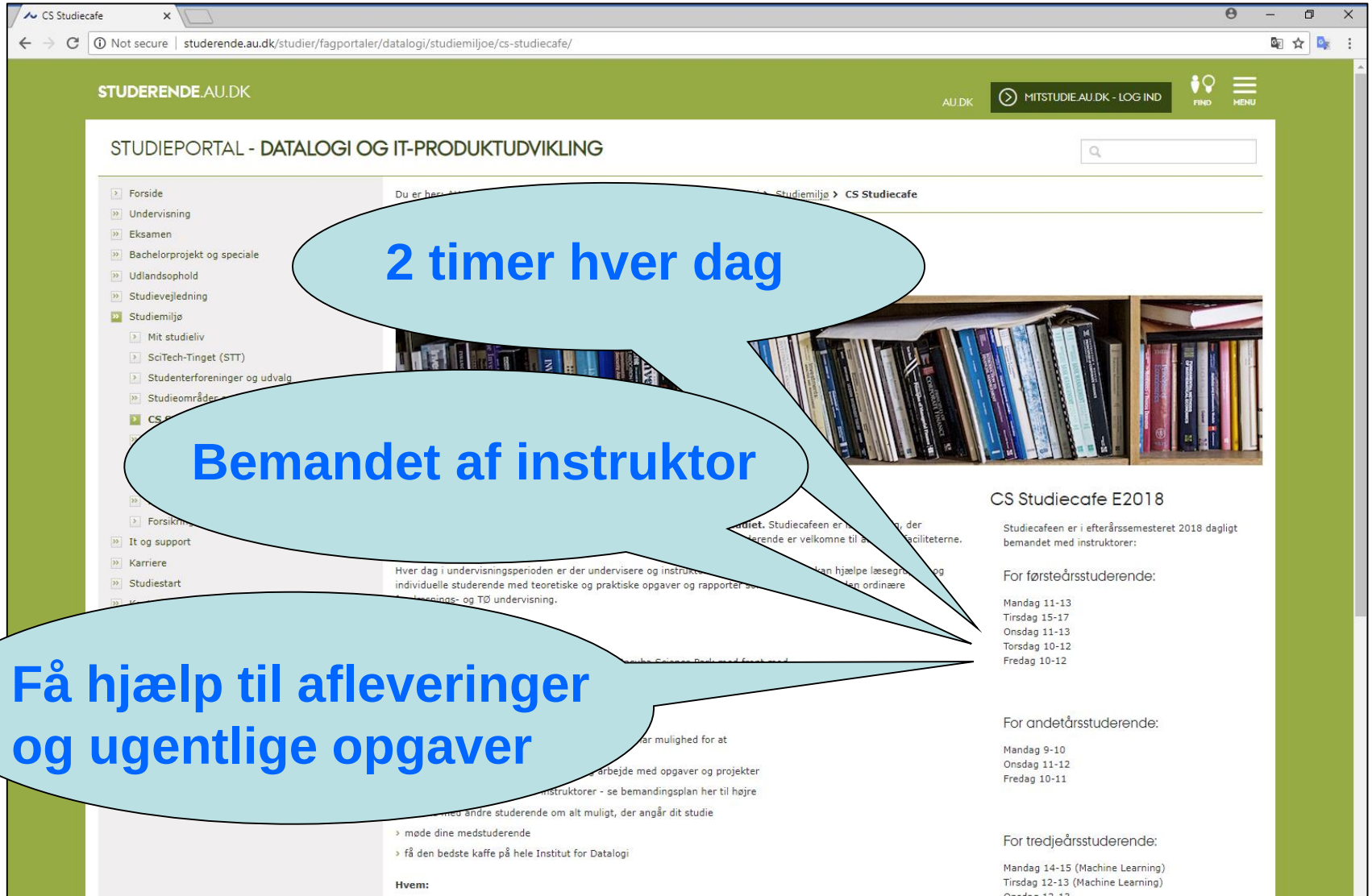
- Alle skal godkendes!
- Hvis en aflevering ikke godkendes vil man blive bedt om at **genaflevere**.

Programmeringsopgaver

- Først et par uger inde i kurset.
- Korrekt løsning af disse vil være en del af endelig karakter.

Studie Café

cs.au.dk/studiecafe



Tidsforbrug?

Fra kursushjemmesiden (Blackboard):

Forventet tidsforbrug

- Forelæsninger 4 timer pr. uge, x 14 uger = 56
- TØ timer 3 timer pr. uge, x 14 uger = 42
- Study Café 1 x 14 = 14
- Afleveringer 3 timer pr. uge, x 14 uger = 42
- Forberedelse til forelæsninger 2 timer pr. uge, x 14 = 28
- Forberedelser til TØ 2 timer pr. uge, x 14 = 28

- Forberedelse til eksamen = 45 timer
- Eksamen = 2 timer
- **I alt 257 timer**

10 ECTS = 250 – 280 timer

Læringsmål

Fra kursusbeskrivelsen:

I slutningen af kurset vil deltagerne kunne:

- **Formulere** og **udføre** algoritmer og datastrukturer i form af pseudokode,
- **Konstruere**, **implementere** og **analysere** algoritmer ved hjælp af standard algoritme paradigmer,
- **Identificere** og **sammenligne** datastrukturer og grafalgoritmer til løsning af algoritmiske problemer,
- **Identificere** gyldige invarianter for en algoritme,
- **Konstruere**, **implementere** og **evaluere** algoritmers ydeevne for simple algoritmiske problemer,
- **Analysere** og **sammenligninge** tid og pladsforbrug af algoritmer og datastrukturer,
- **Bevise** korrektheden af enkle programmer og transitionssystemer.

Spørgsmål ?

Se Blackboard for info, samt slides!

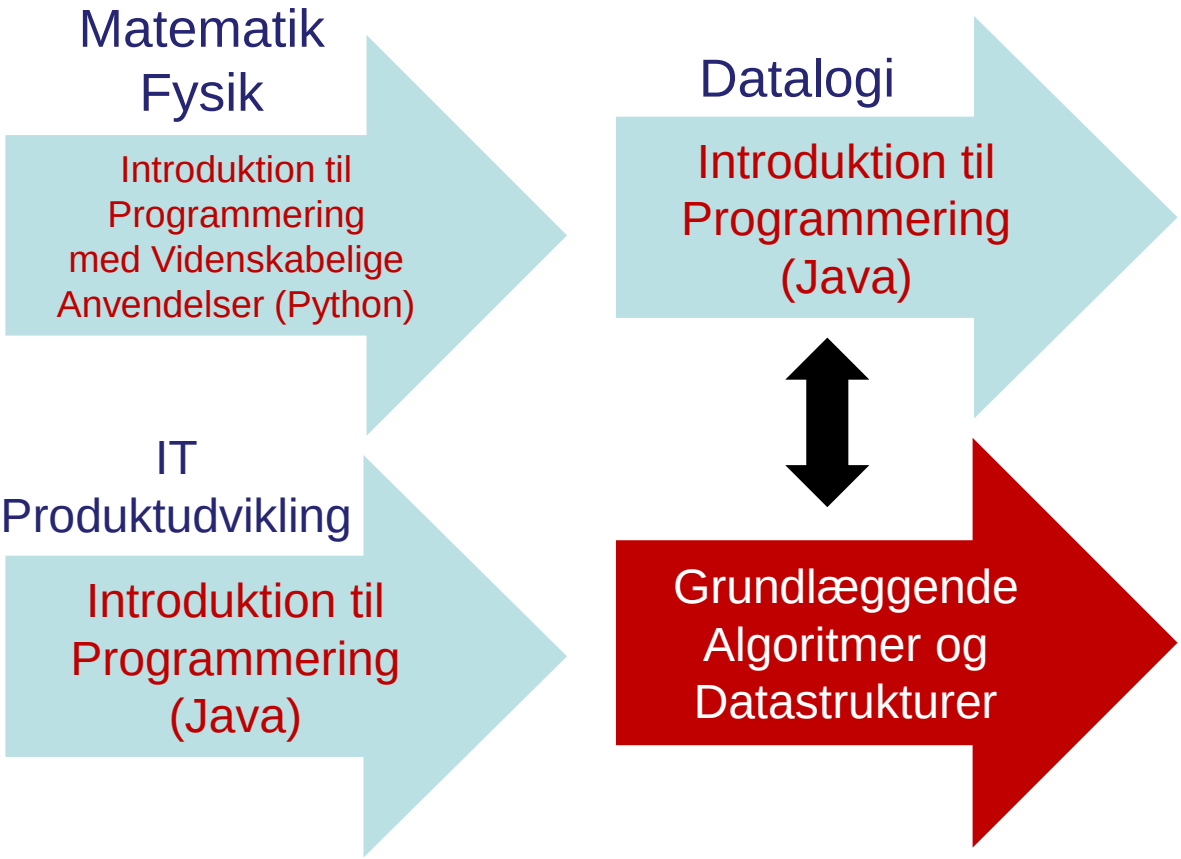
Studieretning

a) Datalogi (typisk 1. år)
b) IT produktudvikling (typisk 3. år)
c) Matematik (typisk 3. år)
d) Fysik (typisk 3. år)
e) andet (typisk tilvalg i datalogi)

Programmeringserfaring
(for det programmeringssprog du måtte kende bedst)

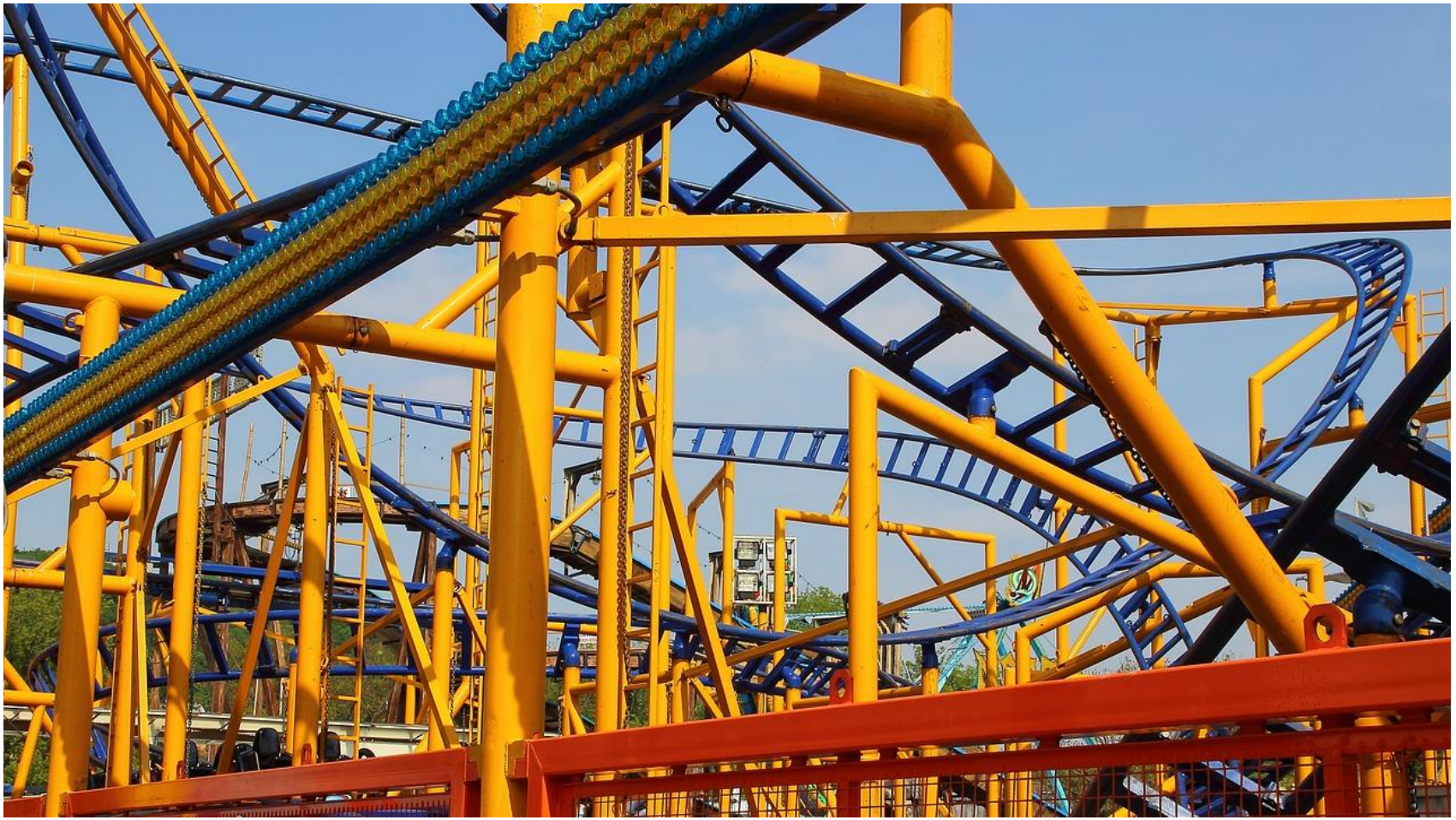
a) Ingen
b) Basal kendskab
c) Grundlæggende kendskab
d) Advanceret
e) Ekspert

- Algoritmer og Datastrukturer omhandler effektivitet af programmer
- Kræver egentlig man kan programmere...



- I dette kursus vil vi anvende basal Java

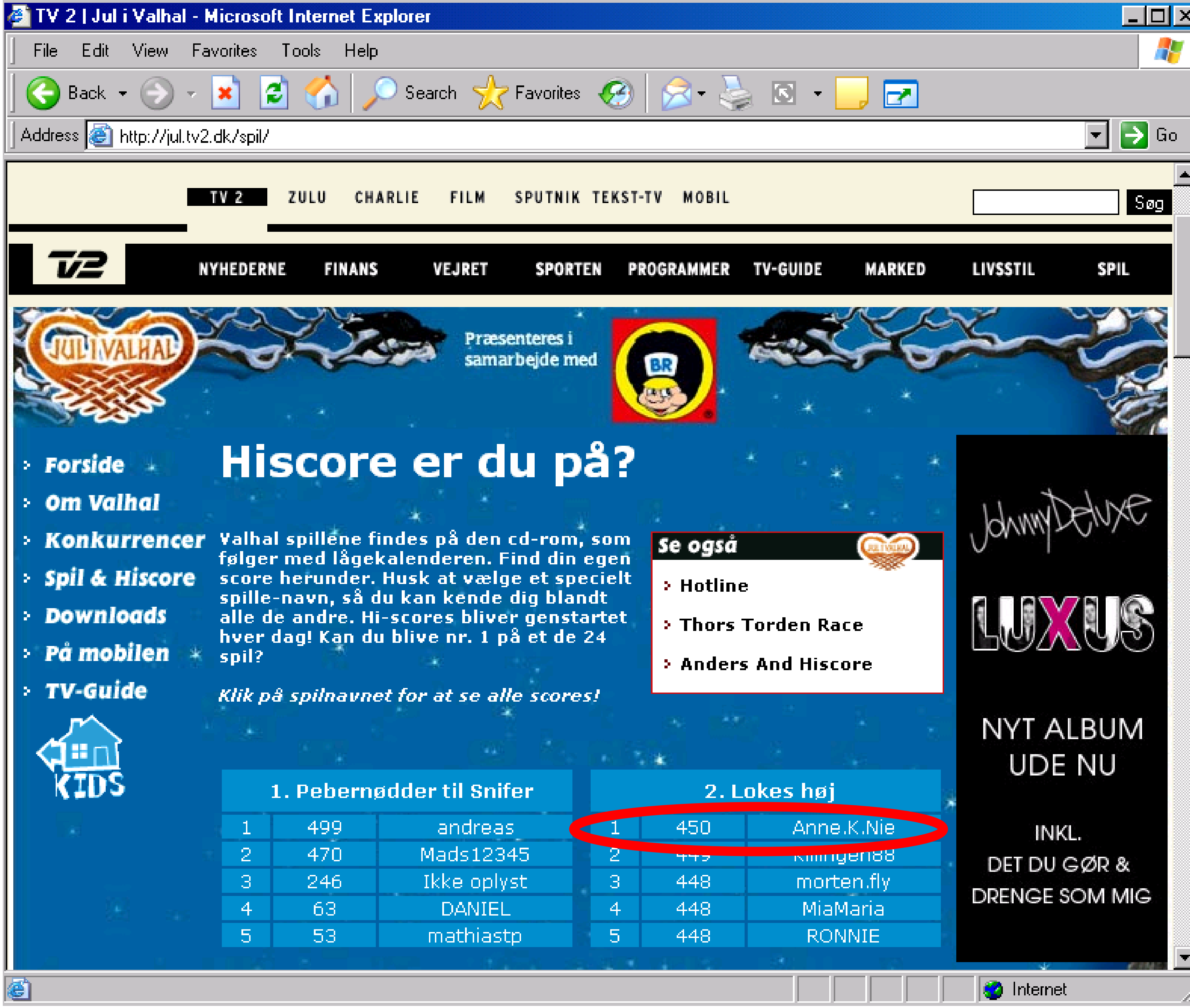
Ready ?



Eksempel på en algoritmisk problemstilling

2005


TV2
500

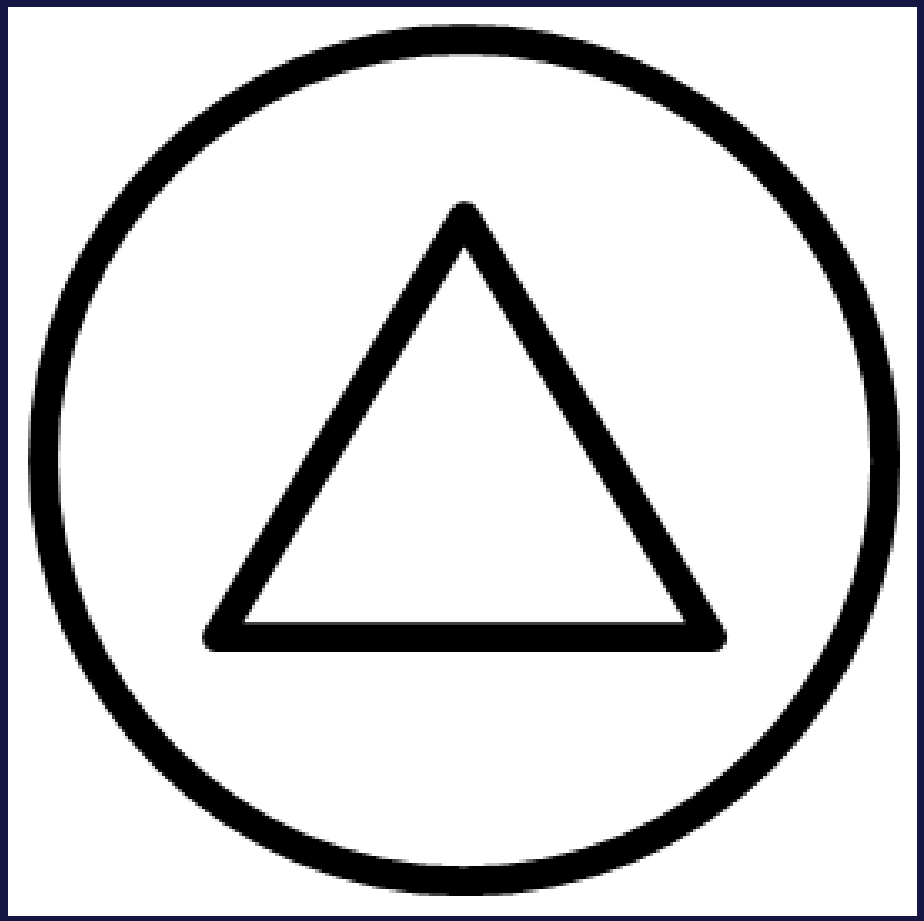

TV 2 | Jul i Valhal - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
Address http://jul.tv2.dk/spil/
Go
TV 2 ZULU CHARLIE FILM SPUTNIK TEKST-TV MOBIL
Søg
NYHEDERNE FINANS VEJRET SPORTEN PROGRAMMER TV-GUIDE MARKED LIVSSTIL SPIL
JUL I VALHAL
Præsenteres i samarbejde med
BR
Forside
Om Valhal
Konkurrencer
Spil & Hiscore
Downloads
På mobilen
TV-Guide
KIDS
Hiscore er du på?
Valhal spillene findes på den cd-rom, som følger med lågekalenderen. Find din egen score herunder. Husk at vælge et specielt spille-navn, så du kan kende dig blandt alle de andre. Hi-scores bliver genstartet hver dag! Kan du blive nr. 1 på et de 24 spil?
Klik på spilnavnet for at se alle scores!
Se også
Hotline
Thors Torden Race
Anders And Hiscore
1. Pebernødder til Snifer
1 499 andreas
2 470 Mads12345
3 246 Ikke oplyst
4 63 DANIEL
5 53 mathiastp
2. Lokes høj
1 450 Anne.K.Nie
2
3 448 morten.fly
4 448 MiaMaria
5 448 RONNIE
JohnnyDeluxe
LUXUS
NYT ALBUM UDE NU
INKL.
DET DU GØR &
DRENGE SOM MIG
Internet

”Lokes Høj”

- 64 brikker
- Hiscore 450
- Antal ombytninger 500 - 450 = 50

Hvordan opnår man et lavt antal ombytninger
– held eller dygtighed ?

Optimale antal ombytninger ?



Optimale antal ombytninger ?

a) 5

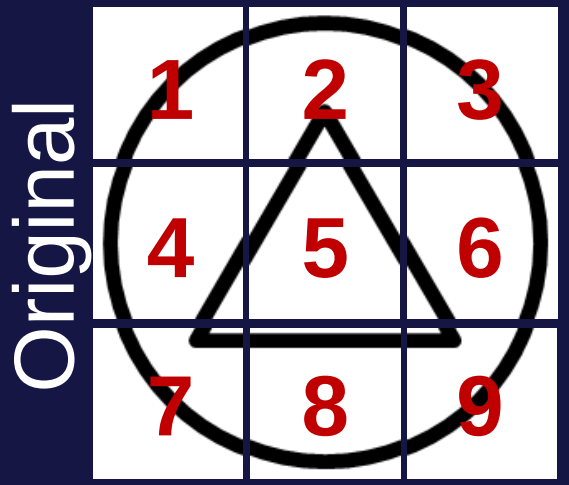
b) 6

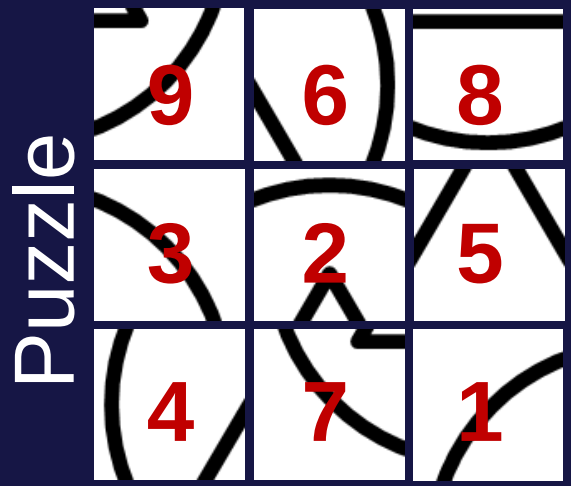
c) 7

d) 8

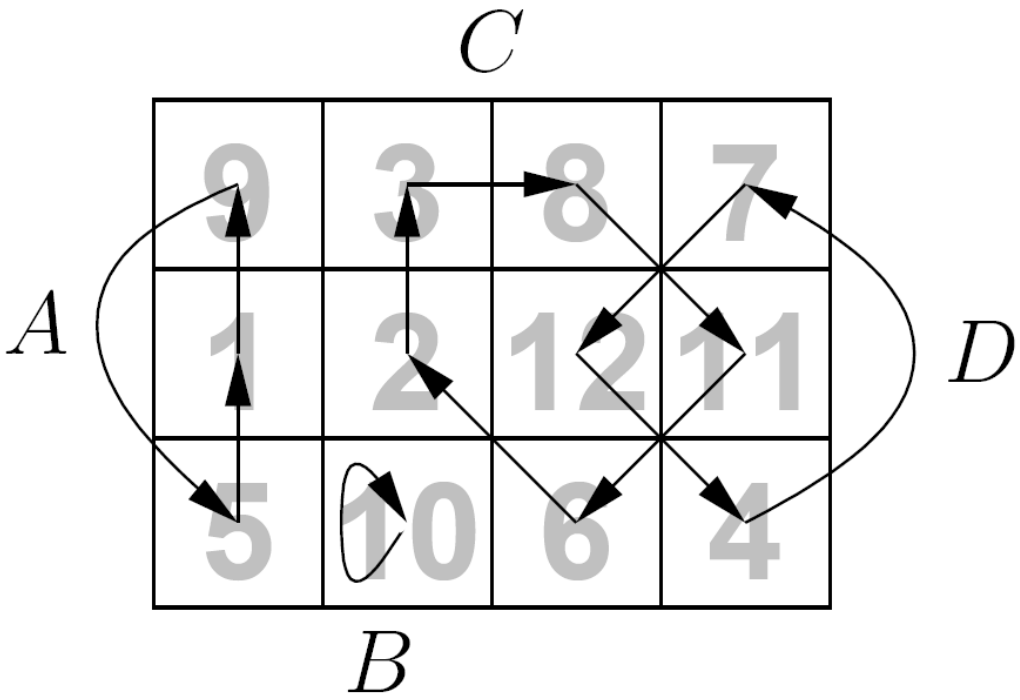
e) 9

f) Ved ikke





Cykler (Permutationer)

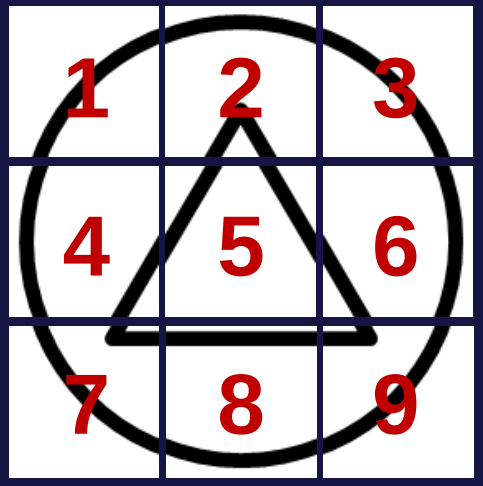


Hver pil peger på brikkens korrekte plads

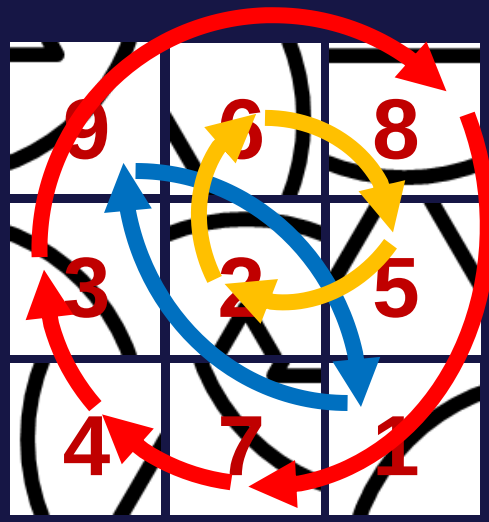
Definerer en mængde af cykler (fx cyklerne A,B,C,D)

Optimale antal ombytninger ?

Original

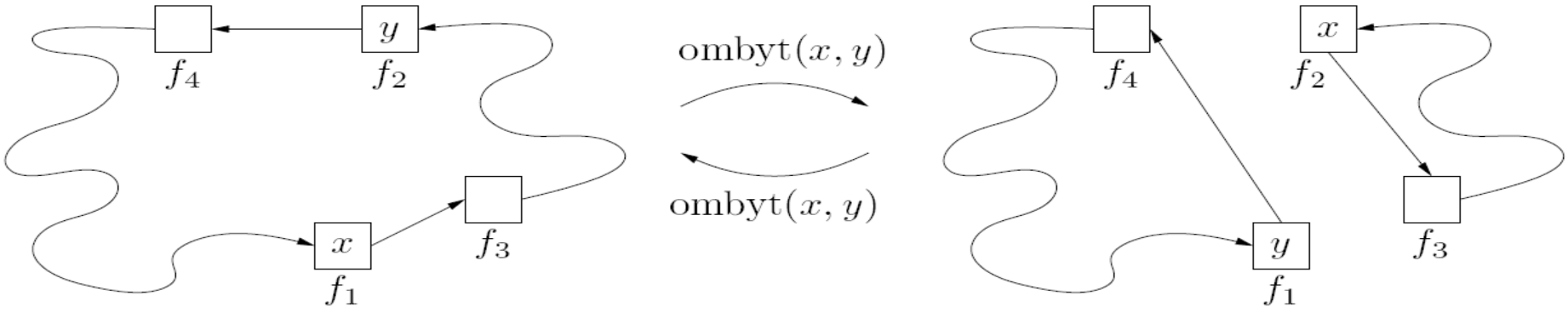


Puzzle



En løsning: 1-9 2-6 5-6 8-3 4-8 8-7

Ombytninger og Cykler



Lemma

- En ombytning af to brikker i **samme cykel** øger antallet af cykler med én.
- En ombytning af to brikker fra to **forskellige cykler** reducerer antallet af cykler med én.

Lemma
Når alle ***n*** brikker er korrekt placeret er der præcis ***n*** cykler.

Lemma
For at løse et puslespil med ***n*** brikker og ***k*** cykler i starten kræves **≥ *n* – *k*** ombytninger.

Har vist en nedre grænse for

ALLE algoritmer der løser problemet

En (grådig) algoritme

Algoritme Puslespil
while der findes en brik x som ikke er placeret korrekt **do**
    lad y være brikken på x's korrekte plads
    ombyt(x, y)
od

Lemma
Algoritmen bytter aldrig om på brikker der står korrekt.

Lemma
Algoritmen udfører $\leq n - 1$ ombytninger

Lemma
For at løse et puslespil med n brikker og k cykler i starten udfører algoritmen præcis $n - k$ ombytninger.

Har vist en øvre grænse for en konkret algoritme

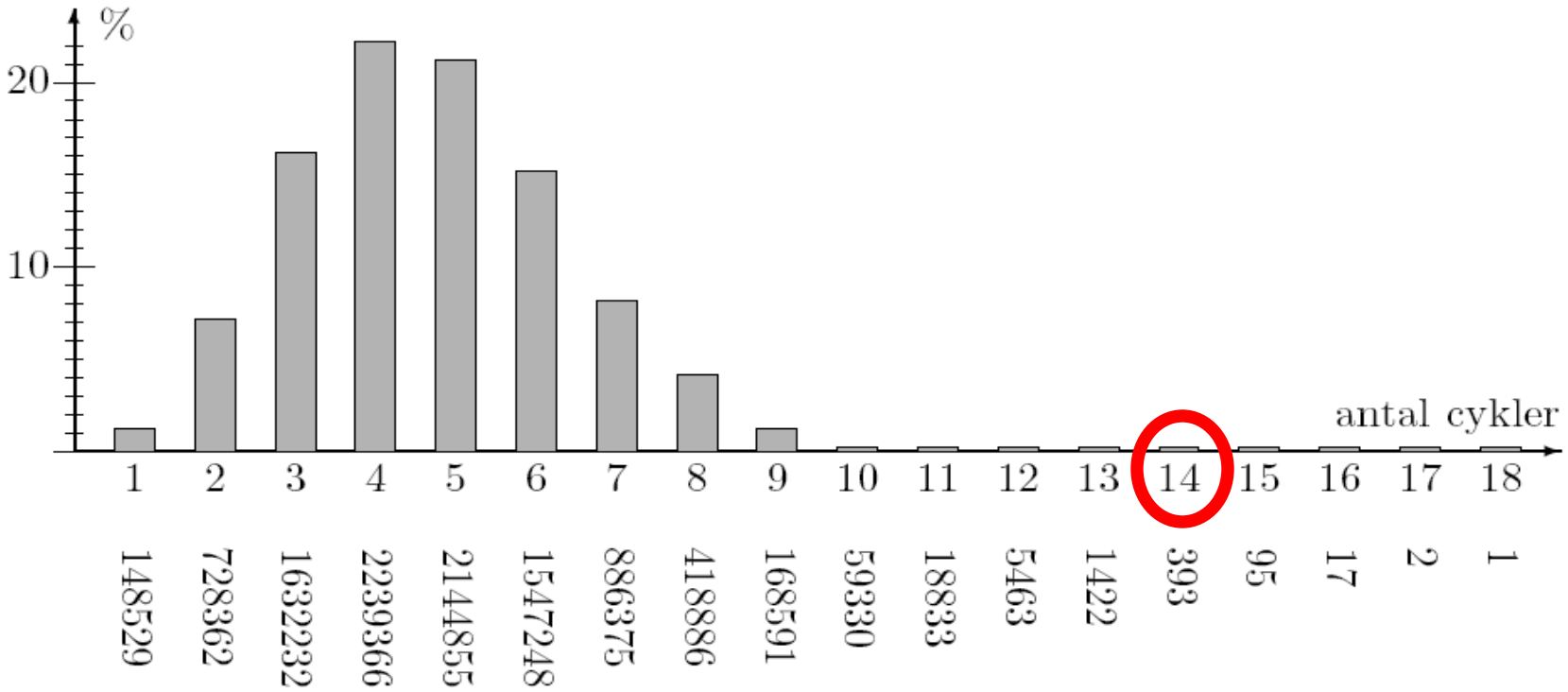
Algoritmen er optimal da antal ombytninger er bedst mulig (de viste nedre og øvre grænser er identiske)

Sætning

For at løse et puslespil med n brikker og k cykler i starten kræves præcis $n - k$ ombytninger

Fordelingen af antal cykler

n = 64, 10.000.000 permutationer

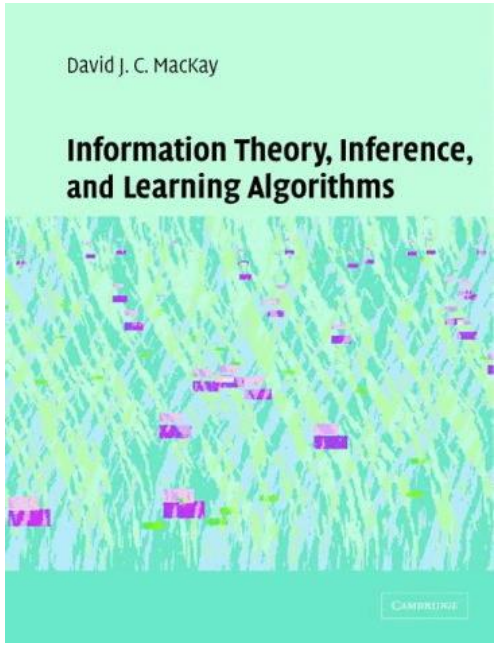


Hvad har vi så lært… ?

Algoritmisk indsigt…

- **Matematisk indsigt** (cykler)
- **Resourceforbrug** (antal ombytninger)
- **Nedre grænse** (≥ *n* - *k* ombytninger)
- **Grådig algoritme**
- **Analyseret algoritmen** (≤ *n* - *k* ombytninger)
- **Optimal algoritme** (argumenteret bedst mulig)
- **Input afhængig resourceforbrug**

Tilfældige permutationer…



Yderligere information kan findes i David J.C. MacKay, tillæg til *Information Theory, Inference, and Learning Algorithms*, om

"Random Permutations“, 4 sider.

http://www.inference.phy.cam.ac.uk/mackay/itila/cycles.pdf

Et andet eksempel på en algoritmisk problemstilling

Søgning i Sorteret Liste

3	7	9	11	13	27	33	37	42	89

$1 + \lfloor \log_2 n \rfloor$ sammenligninger

Et tredie eksempel – en algoritmisk anvendelse

Patience Diff
&
Længste Voksende Delsekvenser

A.c

```
#include <stdio.h>

// Frobs foo heartily
int frobnitz(int foo)
{
    int i;
    for(i = 0; i <
    {
        printf("You
        printf("%d\
    }
}

int fact(int n)
{
    if(n > 1)
    {
        return fact
    }
    return 1;
}

int main(int argc,
{
    frobnitz(fact(1
}
```

B.c

```
#include <stdio.h>

int fib(int n)
{
                                  fib(n-2);
                                  +)
                                  o);
                                  *argv)
```

```
$ diff A.c B.c
3,4c3
< // Frobs foo heartily
< int frobnitz(int foo)
---
> int fib(int n)
6,7c5
<     int i;
<     for(i = 0; i < 10; i++)
---
>     if(n > 2)
9,10c7
<         printf("Your answer is: ");
<         printf("%d\n", foo);
---
>         return fib(n-1) + fib(n-2);
11a9
>     return 1;
14c12,13
< int fact(int n)
---
> // Frobs foo heartily
```

11 removals 10 additions

```
1. #include <stdio.h>
2.
3. // Frobs foo heartily
4. int frobnitz(int foo)
5. {
6.     int i;
7.     for(i = 0; i < 10; i++)
8.     {
9.         printf("Your answer is: ");
10.        printf("%d\n", foo);
11.    }
12. }
13.
14. int fact(int n)
15. {
16.     if(n > 1)
17.     {
18.         return fact(n-1) * n;
19.     }
20.     return 1;
21. }
22.
23. int main(int argc, char **argv)
24. {
25.     frobnitz(fact(10));
26. }
```

```
1. #include <stdio.h>
2.
3. int fib(int n)
4. {
5.     if(n > 2)
6.     {
7.         return fib(n-1) + fib(n-2);
8.     }
9.     return 1;
10. }
11.
12. // Frobs foo heartily
13. int frobnitz(int foo)
14. {
15.     int i;
16.     for(i = 0; i < 10; i++)
17.     {
18.         printf("%d\n", foo);
19.     }
20. }
21.
22. int main(int argc, char **argv)
23. {
24.     frobnitz(fib(10));
25. }
```

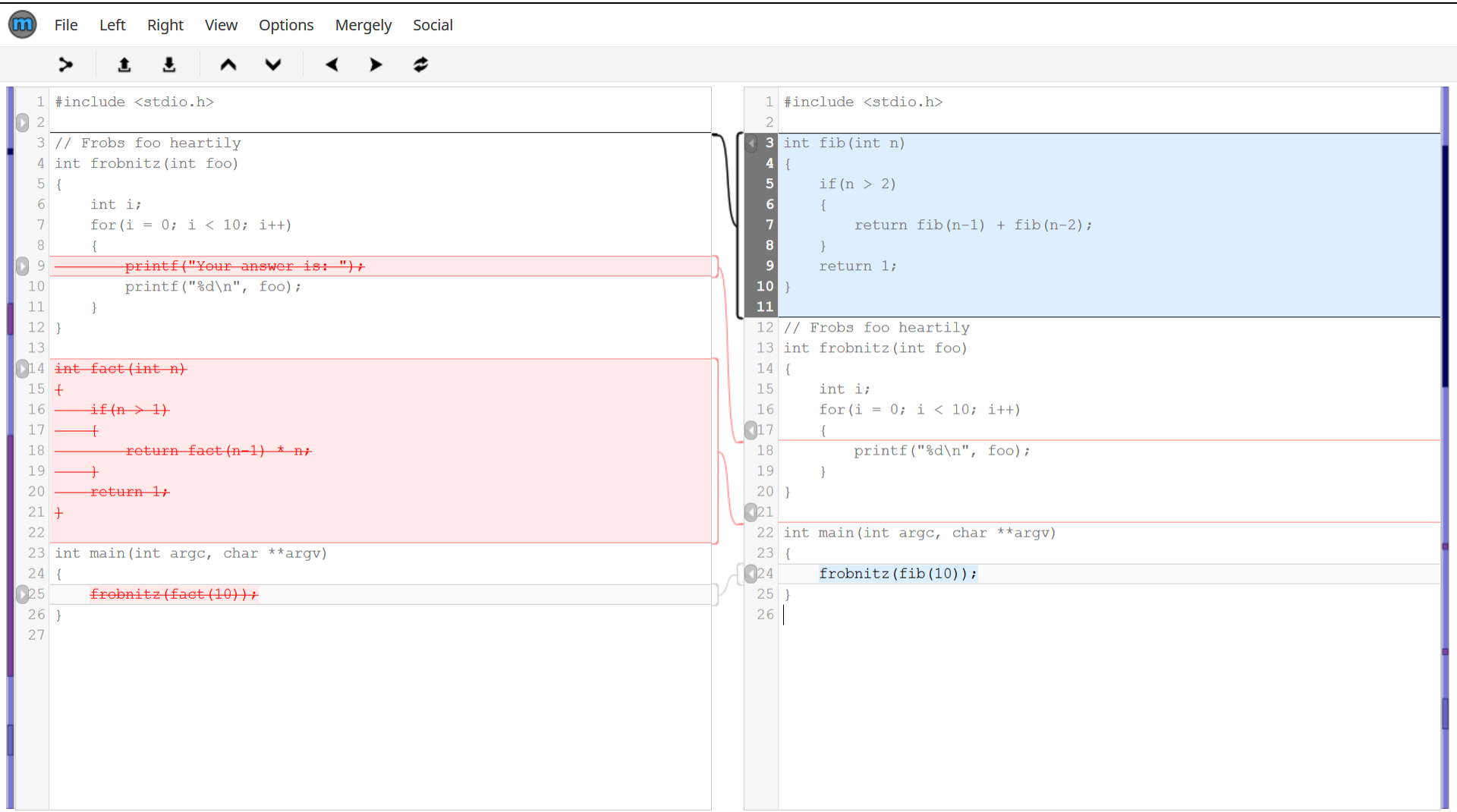
Text Compare!

Email this comparison

```
#include <stdio.h>

// Frobs foo heartily
int frobnitz(int foo)
{
    int i;
    for(i = 0; i < 10; i++)
    {
        printf("Your answer is: ");
        printf("%d\n", foo);
    }
}

int fact(int n)
{
    if(n > 1)
    {
        return fact(n-1) * n;
    }
    return 1;
}

int main(int argc, char **argv)
{
    frobnitz(fact(10));
}
```

```
#include <stdio.h>

int fib(int n)
{
    if(n > 2)
    {
        return fib(n-1) + fib(n-2);
    }
    return 1;
}

// Frobs foo heartily
int frobnitz(int foo)
{
    int i;
    for(i = 0; i < 10; i++)
    {

        printf("%d\n", foo);
    }
}

int main(int argc, char **argv)
{
    frobnitz(fib(10));
}
```

Edit texts ... | Switch texts | Compare! | Clear all

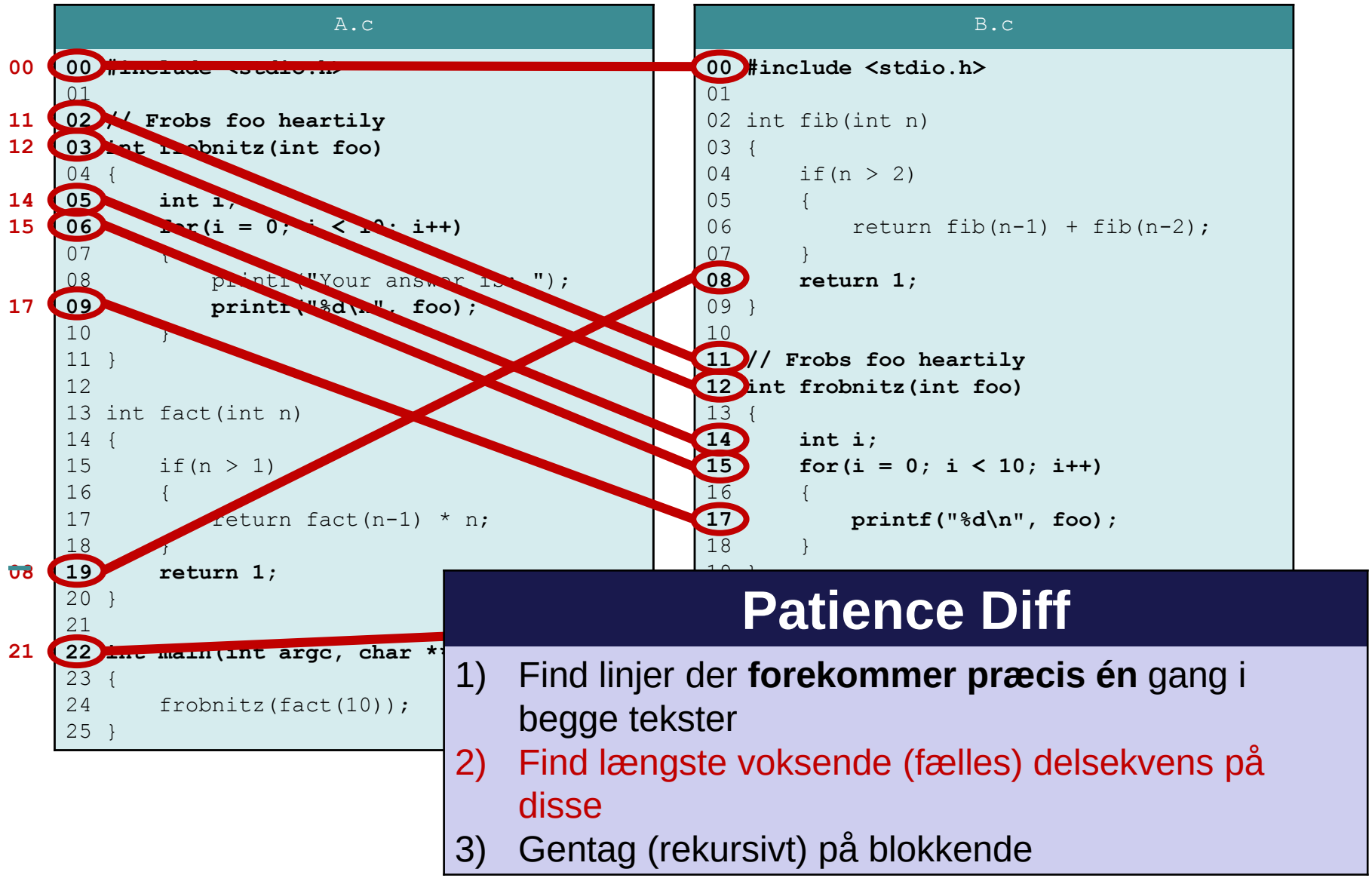

File
Left
Right
View
Options
Mergely
Social
#include <stdio.h>
// Frobs foo heartily
int frobnitz(int foo)
{
int i;
for(i = 0; i < 10; i++)
{
printf("Your answer is: ");
printf("%d\n", foo);
}
}
int fact(int n)
{
if(n > 1)
{
return fact(n-1) * n;
}
return 1;
}
int main(int argc, char **argv)
{
frobnitz(fact(10));
}
#include <stdio.h>
int fib(int n)
{
if(n > 2)
{
return fib(n-1) + fib(n-2);
}
return 1;
}
// Frobs foo heartily
int frobnitz(int foo)
{
int i;
for(i = 0; i < 10; i++)
{
printf("%d\n", foo);
}
}
int main(int argc, char **argv)
{
frobnitz(fib(10));
}

Patient Diff (Bram Cohen)

- Forsøger at lave **læsbar og meningsfuldt ouput –** frem for mindst mulig

- Anvendes i Bazaar versionskontrolsystemet (bazaar-vcs.org)

Mindst mulig løsning findes senere i kurset vha. Dynamisk Programmering


A.c
B.c
00 #include <stdio.h>
01
02 // Frobs foo heartily
03 int frobnitz(int foo)
04 {
05 int i;
06 for(i = 0; i < 10; i++)
07 {
08 printf("Your answer is: ");
09 printf("%d\n", foo);
10 }
11 }
12
13 int fact(int n)
14 {
15 if(n > 1)
16 {
17 return fact(n-1) * n;
18 }
19 return 1;
20 }
21
22 int main(int argc, char **
23 {
24 frobnitz(fact(10));
25 }
00 #include <stdio.h>
01
02 int fib(int n)
03 {
04 if(n > 2)
05 {
06 return fib(n-1) + fib(n-2);
07 }
08 return 1;
09 }
10
11 // Frobs foo heartily
12 int frobnitz(int foo)
13 {
14 int i;
15 for(i = 0; i < 10; i++)
16 {
17 printf("%d\n", foo);
18 }
00
11
12
14
15
17
08
21
Patience Diff
1) Find linjer der forekommer præcis én gang i begge tekster
2) Find længste voksende (fælles) delsekvens på disse
3) Gentag (rekursivt) på blokkende

Længste voksende delsekvens

~~30 83 73 80~~ 59 63 ~~41 78 68 82 53 31 22~~ 74 ~~6 36~~ 99 ~~57 43 60~~

Opgave

Slet så få tal som muligt fra en liste af tal,
så de resterende tal står i voksende orden

Opgave senere i kurset

30 83 73 80 59 63 41 78 68 82 53 31 22 74 6 36 99 57 43 60

Sætning (Erdős og Szekeres, 1935)

Enhver sekvens af n tal har en voksende eller aftagende delsekvens af længde mindst $\lceil \sqrt{n} \rceil$.

3 1 4 17 6 42 10 8 13 11 28 (voksende)

24 3 12 16 7 14 26 8 20 2 1 (aftagende)

Opsummering

- Designe algoritmer
- Analysere algoritmer
  - øvre grænser
  - asymptotisk analyse
- Analysere problemer
  - nedre grænse
- Invarianter
- Korrekthedsargument