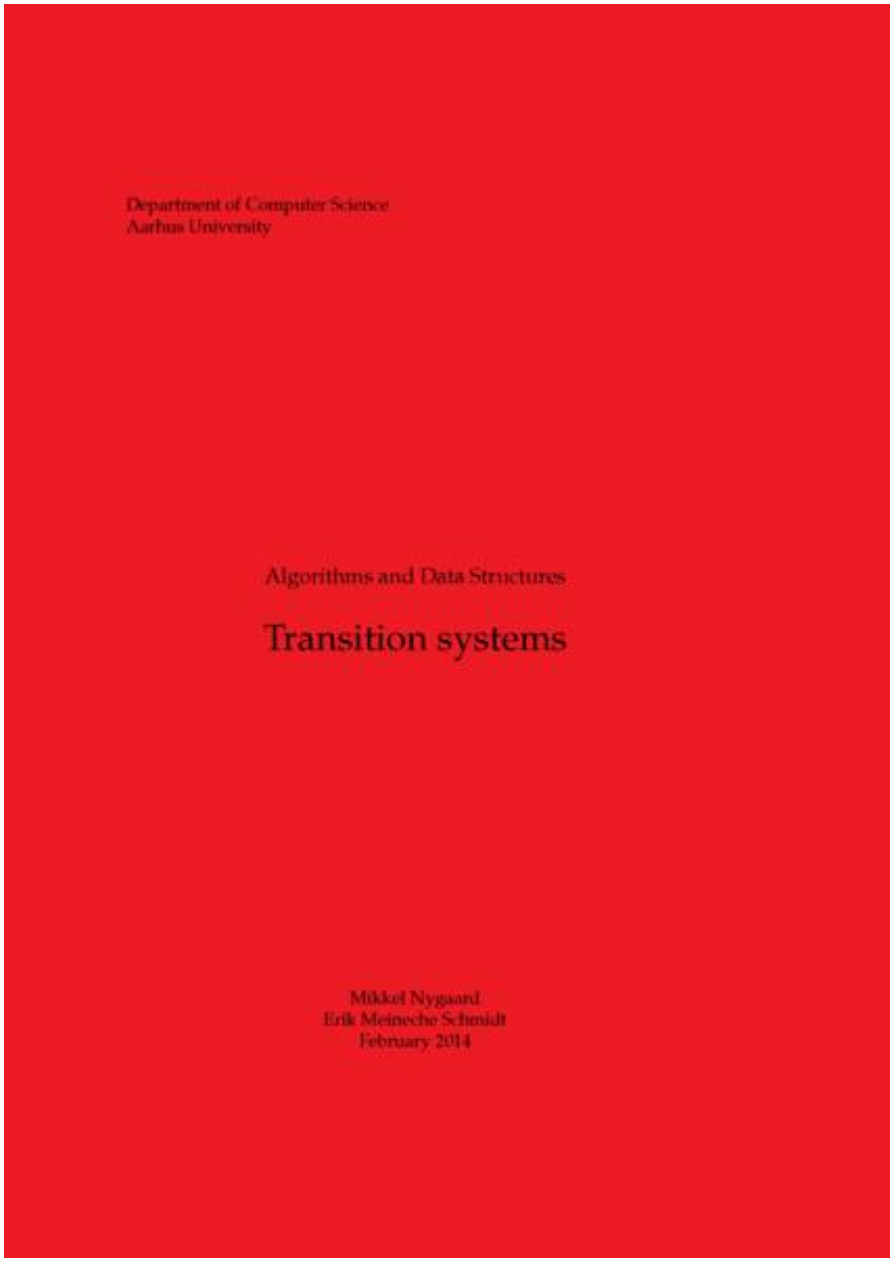
Grundlæggende Algoritmer og Datastrukturer

Invarianter

et værktøj til analyse og design af algoritmer



Note brugt indtil foråret 2017 i kurset Algoritmer og Datastrukturer 1 til at dække emnet Invarianter

Ikke pensum

Transition Systems,
Mikkel Nygaard Hansen og
Erik Meineche Schmidt.
DAIMI-FN-64, Februar 2014.

invariant adjektiv

BØJNING -, -e

UDTALE ['enva͡i̯ˌanˀd]

OPRINDELSE første led latin *in-* i betydningen 'ikke-, u-'

Betydninger

uforanderlig; konstant – bl.a. inden for matematik

GRAMMATIK almindelig som substantiv fælleskøn

```
i ← 0
x ← 100
while i ≤ 10 do
    x ← x + 7
    i ← i + 1
```

Hvilken value har `x` når programmet stopper ?

```
i ← 0
x ← 100
while i ≤ 10 do
    x ← x + 7
    i ← i + 1
```

a) 100 b) 107 c) 110 d) 111 e) 170 f) 177

Løkke-invariant = udsagn

Eksempler på **I**

- i ≥ 0
- x ≥ 100
- i ≤ x

```
i ← 0
x ← 100
{ I }  while i ≤ 10 do
    x ← x + 7
    i ← i + 1
```

x' og i' er de nye værdier

i=0 ^ x=100 → x = 100+7*i ^ i ≤ 11

x=100+7*i ^ i ≤ 11 ^ i≤10
^ x'=x+7 ^ i'=i+1
→ x'=(100+7*i)+7 =100+7*i' ^ i'≤11

x = 100 + 7 * i ∧ i ≤ 11
∧ x og i er heltal

¬(i ≤ 10) ^ (x = 100+7*i ^ i ≤ 11) → i = 11 ^ **x = 177**

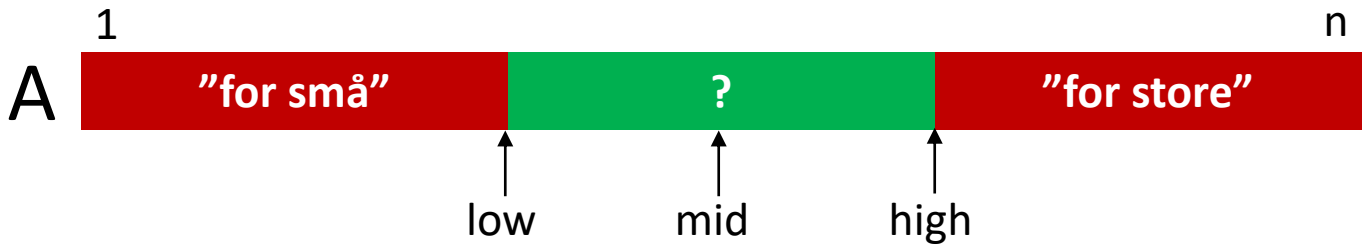
En invariant I skal opfylde

1) Når løkken nås første gang, så er **I** opfyldt
2) Hvis **I** er opfyldt før løkken udføres, så er **I** opfyldt efter en udførsel af løkken

I gælder automatisk når vi kommer ud af løkken

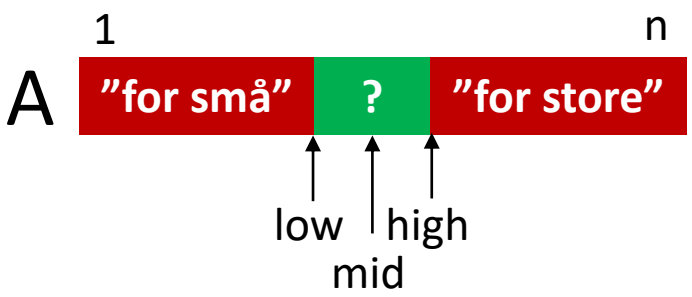
Udnyt **I** og at løkke-betingelsen er forkert når vi er færdig til at drage en konklusion

Binær søgning (i sorteret array)




```
    << initialize >>
{ I } while << loop condition >> do
        mid ← << f(low,high) >>
        << update >>
```

Binær søgning (i sorteret array)




```
                << initialize >>
{ I } while << loop condition >> do
        mid ← << f(low,high) >>
        << update >>
```

I_1: (A[i] < x for $1 \le i \le$ low) ^ (x < A[i] for high $\le i \le n$)

I_2: (A[i] < x for $1 \le i \le$ low) ^ (x $\le$ A[i] for high $\le i \le n$)

I_3: (A[i] < x for $1 \le i <$ low) ^ (x $\le$ A[i] for high < $i \le n$)

I_4: (A[i] < x for $1 \le i <$ low) ^ (x $\le$ A[i] for high $\le i \le n$)

...

<< initialize >>

a) low ← 0; high ← n+1
b) low ← 1; high ← n
c) low ← 0; high ← n
d) low ← 1; high ← n+1
e) ved ikke

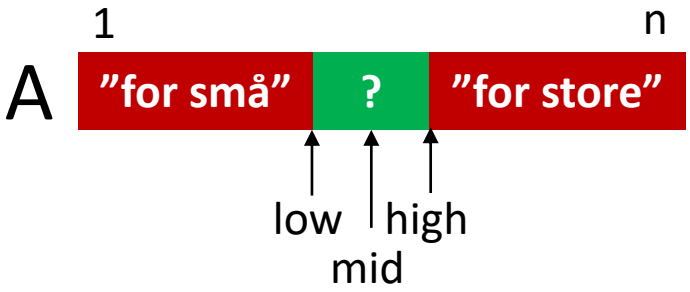
<< loop condition >>

a) low = high
b) low ≠ high
c) low < high
d) low ≤ high
e) low > high
f) ved ikke

<< update >>

a) if A[mid] < x then low ← mid else high ← mid
b) if A[mid] < x then low ← mid+1 else high ← mid
c) if A[mid] < x then low ← mid else high ← mid+1
d) if A[mid] < x then low ← mid+1 else high ← mid-1
e) ved ikke

Binær søgning (i sorteret array)




```
        low ← 1; high ← n
{ I } while low ≤ high  do
          mid ← ⌊low+(high-low)/2⌋
          if A[mid] < x then
             low ← mid+1
          else
             high ← mid-1
```

I: (A[i] < x for 1 ≤ i < low) ^ (x ≤ A[i] for high < i ≤ n)

Hvor står svaret ?

a) low - 1
b) low
c) low + 1
d) high - 1
e) high
f) high + 1
g) ved ikke

Algoritme POWER(x, p)
Inputbetingelse : Heltal $x \geq 0$ og $p \geq 0$
Outputkrav : $r = x^p$
Metode : $r \leftarrow 1;$
$\{I\}$ **while** $p \geq 1$ **do**
if p lige **then**
$x \leftarrow x * x;\ p \leftarrow p/2$
else
$r \leftarrow r * x;\ p \leftarrow p - 1$

p_0 = værdien af p i starten

	Ja	Nej
$p \leq p_0$	☒	☐
$x^p = r$	☐	☒
$x \leq x_0$	☐	☒
$r \cdot x_0^{p_0} = x^p$	☐	☒
$x_0^{p_0} = r \cdot x^p$	☒	☐

Eksamensopgave, Algoritmer og Datastrukturer 1, marts 2017, opgave 21

Algoritme LOG2(n)
Inputbetingelse : Heltal $n \geq 2$
Outputkrav : $r = \text{intlog}(n) = \lfloor \log_2 n \rfloor$
Metode : $i \leftarrow 1;$
$r \leftarrow 1;$
$p \leftarrow 2;$
$\{I\}$ **while** $2p \leq n$ **do**
if $p * p \leq n$ **then**
$p \leftarrow p * p;$
$r \leftarrow 2 * r$
else
$p \leftarrow 2 * p;$
$r \leftarrow r + 1$

	Ja	Nej
$1 \leq r < p$	☒	☐
$2p \leq n$	☐	☒
$p = 2^r$	☒	☐
$p = 2r$	☐	☒
$p = 2^{\text{intlog}(p)}$	☒	☐

Eksamensopgave, Algoritmer og Datastrukturer 1, marts 2015, opgave 21

Invarianter

- Værktøj til analyse af tilstandene i en løkke i en algoritme
- Designværktøj ”Invariant → kode”
- Kan indfange essentielle egenskaber ved en datastrukturs tilstand (f.eks. heap-order i en binær heap, eller rød-sort træ egenskaber)