Algoritmer og Datastrukturer

Induktion og invarianter

Bevis ved induktion

- Vi ønsker at bevise en uendelig række udsagn

$$U_1, U_2, U_3, U_4, U_5, \ldots, U_n, U_{n+1}, \ldots.$$

- F.eks. kan U_n være udsagnet: $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Udsagnene kan **bevises v.h.a. induktionsprincipet**, ved at bevise følgende to udsagn:

1. **Basis**:
   Vis at U_1 gælder
2. **Induktionsskridt**:
   Antag for et $n \geq 1$ at U_n gælder (**induktionshypotese**)

   og vis så at U_{n+1} gælder, dvs. vis at $U_n \Rightarrow U_{n+1}$

Eksempel: Bevis $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

[CLRS A.1] (A.1)

- Basis $n = 1$: $\sum_{i=1}^{1} i = 1 = \frac{1(1+1)}{2}$
- Induktionsskridt:

  Induktionshypotese: $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

  Skal vise at: $\sum_{i=1}^{n+1} i = \frac{(n+1)\left((n+1)+1\right)}{2}$

Bevis: $\sum_{i=1}^{n+1} i = (n+1) + \sum_{i=1}^{n} i = n + 1 + \frac{n(n+1)}{2}$

$$= \frac{2(n+1) + n(n+1)}{2} = \frac{(2+n)(n+1)}{2}$$

$$= \frac{(n+1)\left((n+1)+1\right)}{2}$$

$\square$

Eksempel: Bevis $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

[CLRS A.1] (A.5)

- Basis $n = 0$: $\sum_{i=0}^{0} 2^i = 2^0 = 1 = 2^{0+1} - 1$
- Induktionsskridt:

  Induktionshypotese: $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$

  Skal vise at: $\sum_{i=0}^{n+1} 2^i = 2^{(n+1)+1} - 1$

Bevis:

$$\sum_{i=0}^{n+1} 2^i = 2^{n+1} + \sum_{i=0}^{n} 2^i = 2^{n+1} + 2^{n+1} - 1$$

$$= 2 \cdot 2^{n+1} - 1 = 2^{n+2} - 1 = 2^{(n+1)+1} - 1 \qquad \square$$

Eksempel: Bevis $\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}$

[CLRS A.1] (A.5)

- Basis $n = 0$: $\sum_{i=0}^{0} x^i = x^0 = 1 = \frac{x^{0+1}-1}{x-1}$ $\qquad x \neq 1$
- Induktionsskridt:

  Induktionshypotese: $\sum_{i=0}^{n} x^i = \frac{x^{n+1}-1}{x-1}$

  Skal vise at: $\sum_{i=0}^{n+1} x^i = \frac{x^{(n+1)+1}-1}{x-1}$

  Bevis:

$$\sum_{i=0}^{n+1} x^i = x^{n+1} + \sum_{i=0}^{n} x^i = x^{n+1} + \frac{x^{n+1}-1}{x-1}$$

$$= \frac{(x-1)x^{n+1}+x^{n+1}-1}{x-1} = \frac{x^{n+2}-1}{x-1} = \frac{x^{(n+1)+1}-1}{x-1} \qquad \square$$

Eksempel: Bevis $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

[CLRS A.1] (A.3)

- Basis $n = 1$: $\sum_{i=1}^{1} i^2 = 1 = \frac{1(1+1)(2\cdot 1+1)}{6}$
- Induktionsskridt:

Induktionshypotese: $\sum_{i=1}^{n} i^2 = \frac{n(n+1)(2n+1)}{6}$

Skal vise at: $\sum_{i=1}^{n+1} i^2 = \frac{(n+1)\big((n+1)+1\big)(2(n+1)+1)}{6}$

Bevis $\sum_{i=1}^{n+1} i^2 = (n+1)^2 + \sum_{i=1}^{n} i^2$

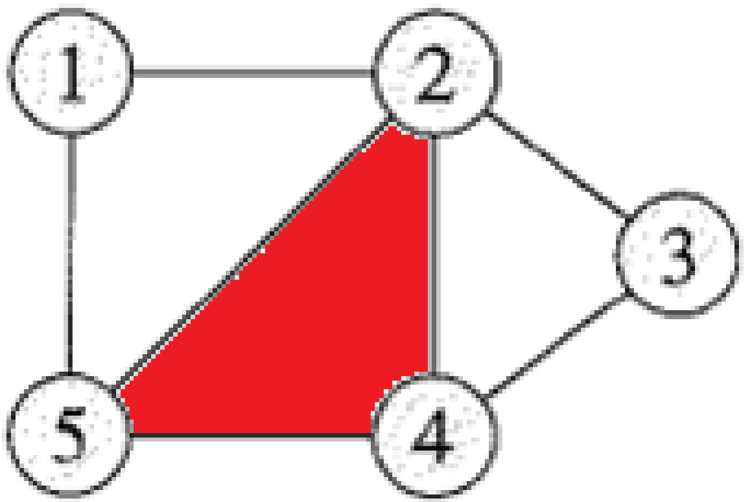
$$= (n+1)^2 + \frac{n(n+1)(2n+1)}{6} = \frac{6(n+1)^2 + n(n+1)(2n+1)}{6}$$

$$= \frac{(n+1)(2n^2+7n+6)}{6} = \frac{(n+1)(n+2)(2n+3)}{6}$$

$$= \frac{(n+1)\big((n+1)+1\big)(2(n+1)+1)}{6}$$

$\square$

Planar Grafer - Eulers formel



$|V| = 5$ knuder
$|E| = 7$ kanter
flader = 4

For en sammenhængende planar graf gælder:

Eulers formel: $\boxed{|V| - |E| + \#\text{ flader} = 2}$

Korollar: $|E| \leq 3|V| - 6$

(for $|V| \geq 3$, ingen selvløkker, ingen parallelle kanter)

DEN DANSKE ORDBOG
MODERNE DANSK SPROG

invariant adjektiv

BØJNING -, -e

UDTALE ['envɑiˌanˀd]

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**

- $i \geq 0$
- $x \geq 100$
- $i \leq x$

$x = 100 + 7 * i \wedge i \leq 11$
$\wedge$ x og i er heltal

```
i ← 0
x ← 100
```

i=0 ^ x=100 → x = 100+7*i ^ i ≤ 11

x' og i' er de nye værdier

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

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

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

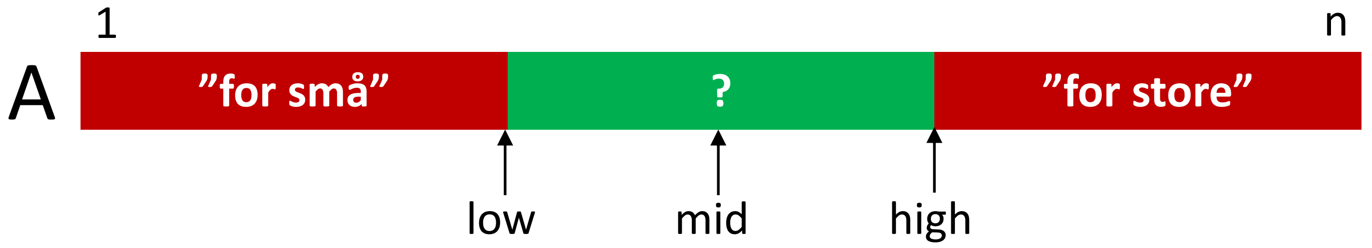
En invariant I skal opfylde

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

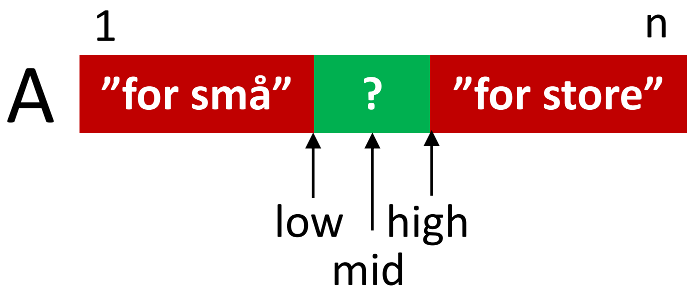
Udnyt **I** og at løkke-betingelsen er forkert når vi er færdig til at drage en konklusion

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 >>
```

$\mathbf{I_1}$: $(A[i] < x \text{ for } 1 \leq i \leq low) \wedge (x < A[i] \text{ for } high \leq i \leq n)$

$\mathbf{I_2}$: $(A[i] < x \text{ for } 1 \leq i \leq low) \wedge (x \leq A[i] \text{ for } high \leq i \leq n)$

$\mathbf{I_3}$: $(A[i] < x \text{ for } 1 \leq i < low) \wedge (x \leq A[i] \text{ for } high < i \leq n)$

$\mathbf{I_4}$: $(A[i] < x \text{ for } 1 \leq i < low) \wedge (x \leq A[i] \text{ for } high \leq i \leq n)$

...

<< initialize >>

a) low $\leftarrow$ 0; high $\leftarrow$ n+1

b) low $\leftarrow$ 1; high $\leftarrow$ n

c) low $\leftarrow$ 0; high $\leftarrow$ n

d) low $\leftarrow$ 1; high $\leftarrow$ n+1

e) ved ikke

I : $(A[i] < x \text{ for } 1 \le i < low) \wedge (x \le A[i] \text{ for } high < i \le n)$

<< loop condition >>

a) $low = high$

b) $low \neq high$

c) $low < high$

d) $low \leq high$

e) $low > high$

f) ved ikke

$\mathbf{I}$: $(A[i] < x \text{ for } 1 \leq i < low) \wedge (x \leq A[i] \text{ for } high < i \leq n)$

<< update >>

a) if $A[mid] < x$ then $low \leftarrow mid$ else $high \leftarrow mid$

b) if $A[mid] < x$ then $low \leftarrow mid+1$ else $high \leftarrow mid$

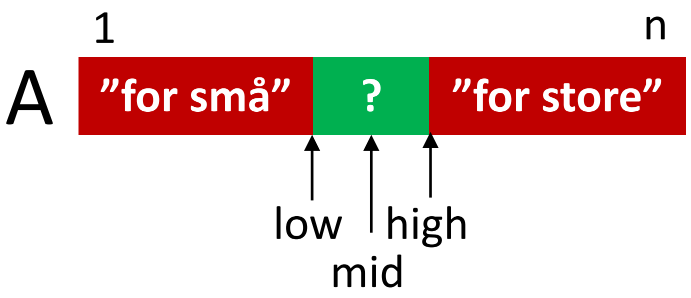
c) if $A[mid] < x$ then $low \leftarrow mid$ else $high \leftarrow mid+1$

d) if $A[mid] < x$ then $low \leftarrow mid+1$ else $high \leftarrow mid-1$

e) ved ikke

I: $(A[i] < x \text{ for } 1 \leq i < low) \wedge (x \leq A[i] \text{ for } high < i \leq n)$

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

I : $(A[i] < x \text{ for } 1 \le i < low) \wedge (x \le A[i] \text{ for } high < i \le n)$

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

Invarianter

- Værktøj til analyse af tilstandene i en løkke i en algoritme
- Designværktøj ”Invariant → kode”
- Kan indfange essentielle egenskaber ved en datastrukturs tilstand