Bevisteknikker

Bevisteknikker

(relevant både ved design og verifikation)

Bevisførelse ved modstrid (indirekte bevis)

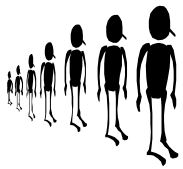
> **Antag, at det givne teorem er falsk**
> **Konkluder, at dette vil føre til en modstrid**

Teorem: Der findes uendeligt mange primtal

Bevis: Antag at der findes et endeligt antal primtal, $p_1, p_2, ..., p_k$
Betragt nu tallet $N = p_1 \cdot p_2 \cdots p_k + 1$
N er forskellig fra alle kendte primtal
Men ingen af de kendte primtal går op i N (resten ved division er 1)
Så N er et primtal
Vi har dermed en modstrid, hvilket beviser sætningen

Matematisk induktion
uformel beskrivelse

Nogle personer står i kø:



Hvis
(1) jeg kender den forreste person, og
(2) hvis jeg for enhver person, jeg kender i køen,
også kender dennes efterfølger,
så kender jeg **alle** personer i køen. Også selv om køen er uendelig lang

Matematisk induktion
formel beskrivelse

Lad T være et teorem, der skal bevises, og lad T være udtrykt i termer af heltalsparameteren n

Teoremet T gælder da for enhver værdi af $n \geq c$, hvor c er en konstant, hvis følgende to betingelser er opfyldt:

1. **Basistilfældet**:
   T gælder for $n = c$, og
2. **Induktionsskridtet**:
   Hvis T gælder for n-1, så gælder T for n

Antagelsen i induktionsskridtet kaldes **induktionshypotesen**.

Eksempler på induktion

Teorem:
Summen $S(n)$ af de første n naturlige tal er $n(n+1)/2$

Bevis:
(1) **Basistilfældet**.
For $n = 1$ er $S(1) = 1$, hvilket stemmer med formlen

(2) **Induktionsskridtet**.
Antag at sætningen gælder for n-1, d.v.s. $S(n\text{-}1) = (n\text{-}1)n/2$
$S(n) = S(n\text{-}1) + n = (n\text{-}1)n/2 + n = n(n+1)/2$
Dermed gælder sætningen også for n

Møntveksling

Teorem: Ethvert beløb ≥ 4 kroner kan veksles i et antal 2-kroner og et antal 5-kroner.

Bevis:
(1) **Basistilfældet**.
4 kroner kan veksles ved hjælp af to 2-kroner.

(2) **Induktionsskridtet**.
Antag at n-1 kroner kan veksles. Vi kan vise, at denne veksling kan benyttes som udgangspunkt til at veksle n kroner
Enten indeholder vekslingen en 5-krone, eller også gør den det ikke
I første tilfælde erstattes 5-kronen med tre 2-kroner
I andet tilfælde erstattes to 2-kroner med en 5-krone

Stærk induktion

Teoremet T gælder for enhver værdi af $n \geq c$, hvor c er en konstant, hvis følgende to betingelser er opfyldt:

1. **Basistilfældet**:
   T gælder for $n = c$, og
2. **Induktionsskridtet**:
   Hvis T gælder for **ethvert** k, $c \leq k < n$, så gælder T for n

Induktion kan benyttes ved design af algoritmer

Induktionsprincippet kan benyttes konstruktivt
Løsning af små problemer benyttes til at løse større problemer

(1) Start med en vilkårlig instans af problemet

(2) Prøv at løse dette under antagelse af, at det samme problem - men af mindre størrelse - er blevet løst

At ræsonnere om algoritmer

Evnen til at drage logiske slutninger er nyttig ved

- algoritmedesign
- fejlfinding
- forbedringer
- verifikation (bevis for korrekthed)

Verifikation af algoritmer

En algoritme *A* siges at være **partielt korrekt**, når der gælder følgende:
Hvis A **terminerer** for et givet legalt input, så vil dens output være korrekt

En algoritme *A* siges at være **korrekt** (eller **totalt korrekt)**, hvis *A* er partielt korrekt, og *A* **terminerer** for ethvert legalt input

Bevisførelse af partiel korrekthed foretages ved brug af **påstande** (assertions)

En **påstand** er et logisk udsagn, der er knyttet til et givet punkt i algoritmen, og som er opfyldt, hver gang algoritmen når til dette punkt

Påstande

- **Før-betingelse (precondition)**
  Et udsagn, som gælder før udførelsen af en eller flere sætninger
- **Efter-betingelse (postcondition)**
  Et udsagn, som gælder efter udførelsen af en eller flere sætninger
- **Løkke-invariant (loop invariant)**
  Et udsagn, som gælder umiddelbart før løkke-testen i en løkke

Eksempler på påstande

```
{ m ≤ n }                                   ← før-betingelse
sort(a, m, n);
{ a[m] ≤ a[m+1] ≤ . . . ≤ a[n] }            ← efter-betingelse
```

```
i = m;
while
  { a[m] ≤ a[m+1] ≤ . . . ≤ a[i-1] ^
    a[m .. i-1] ≤ a[i .. n] }               ← løkke-invariant
    (i < n) {
    min = i;
    for (j = i + 1; j <= n; j++)
        if (a[j] < a[min]) min = j;
    x = a[i]; a[i] = a[min]; a[min] = x;
    i++;
}
```

Regler for verifikation

- **Tildeling:**

$\{ P_w \}$ $\{ w > 2 \}$

```
v = w;
```

$\{ P_{w \to v} \}$ $\{ v > 2 \}$

- **Valg:**

$\{P\}$

```
if (B)    { P ^ B }
   S1;
else      { P ^ ¬ B }
   S2;
```


Eksempel på verifikation

(heltalsdivision)

Givet algoritmen

```
q = 0; r = x;
while (r >= y) {
    r = r - y;
    q = q + 1;
}
```

hvor x, y, q og r er heltal, $x \geq 0$ og $y > 0$

Bevis, at algoritmen bestemmer kvotienten q og resten r ved heltalsdivision af x med y, d.v.s.

$r = x - qy \qquad (0 \leq r < y)$

Bevisførelsen

(1) Løkke-invarianten er opfyldt ved indgangen til løkken

(2) Hvis løkke-invarianten er opfyldt i en given iteration, så er den også opfyldt i den efterfølgende iteration

} induktion

(3) Hvis algoritmen terminerer, så er dens efter-betingelse opfyldt

(4) Algoritmen terminerer

$\{ x \geq 0, y > 0 \}$ ← algoritmens før-betingelse

```
q = 0;
r = x;
while   { r = x - qy ^ r ≥ 0 }   ← løkke-invariant
    (r >= y) {
    { r = x - qy ^ r ≥ y } ≡ { r - y = x - (q+1)y ^ r - y ≥ 0 }
    r = r - y;
    { r = x - (q+1)y ^ r ≥ 0 }
    q = q + 1;
}
```

$\{ r = x - qy \wedge 0 \leq r < y \}$ ← algoritmens efter-betingelse

Terminering


```
q = 0; r = x;
while (r >= y) {
    r = r - y;
    q = q + 1;
}
```

Algoritmen terminerer

Bevis:

Da $\mathbf{y} > 0$, mindskes $\mathbf{r}$ ved hvert gennemløb af løkken
Løkketesten må derfor blive falsk efter et endeligt antal gennemløb