Strukturel induktion.

Mængden S er defineret rekursivt ved:
Basisskridt: vi har angivet et eller flere elementer, der tilhører S
Rekursionsskridt: vi har angivet en eller flere regler, der hver ud fra et eller flere elementer i S konstruerer et nyt element i S.

Lad $P(x)$ være et åbent udsagn, $x \in S$.

For at bevise at $P(x)$ er sand for alle $x \in S$ skal vi:
Basisskridt: bevise at $P(x)$ er sand for ethvert x indført i basisskridtet af definitionen af S
Rekursionsskridt: bevise at hvis x er konstrueret fra $x_1, \ldots, x_\ell$ i rekursionsskridtet af definitionen af S og hvis $P(x_1), \ldots, P(x_\ell)$ er sande så er $P(x)$ sand.

Strenge

Σ : et alfabet, altså en endelig mængde af symboler.

Definition. Σ^*, mængden af strenge over Σ defineres ved:
Basisskridt: Den tomme streng $\lambda \in \Sigma^*$.
Rekursionsskridt: Hvis $w \in \Sigma^*$ og $x \in \Sigma$ så er $wx \in \Sigma^*$.

Definition. Konkatenering af strenge $w_1 \cdot w_2$ af to strenge $w_1, w_2 \in \Sigma^*$ defineres ved:
Basisskridt: $w_1 \cdot \lambda = w_1$
Rekursionsskridt: Hvis $w_2 \in \Sigma^*$ så er

$$w_1 \cdot (w_2 x) = (w_1 \cdot w_2)x.$$

Definition. Længden $\ell(w)$ af en streng w defineres ved
Basisskridt: $\ell(\lambda) = 0$
Rekursionsskridt: Hvis $w = w_1 x$, hvor $w_1 \in \Sigma^*$ og $x \in \Sigma$ så er $\ell(w) = \ell(w_1) + 1$.

Sætning. $\ell(w_1 \cdot w_2) = \ell(w_1) + \ell(w_2)$.

Binære træer

Definition. Mængden af fulde binære træer kan defineres ved:
Basisskridt: Et træ, der består ét punkt r, er et fuldt binært træ med rod r.
Rekursionsskridt: Hvis T_1 og T_2 er fulde binære træer så er $T_1 \cdot T_2$ et fuldt binært træ, der består af T_1, T_2 og en rod r samt en kant fra r til roden af T_1 og en kant fra r til roden af T_2.

$n(T)$: antal punkter i et fuldt binært træ T opfylder:
Basisskridt: Hvis T består af en rod så er $n(T) = 1$.
Rekursionsskridt: Hvis $T = T_1 \cdot T_2$ så er

$$n(T) = n(T_1) + n(T_2) + 1.$$

$h(T)$: højden af et fuldt binært træ T defines ved
Basisskridt: Hvis T består af en rod så er $h(T) = 0$.
Rekursionsskridt: Hvis $T = T_1 \cdot T_2$ så er

$$h(T) = \max\{h(T_1), h(T_2)\} + 1.$$

Sætning. Et fuldt binært træ T opfylder

$$n(T) \leq 2^{h(T)+1} - 1.$$

Algoritme: Iterativ beregning af n!

procedure iterativ factorial(n: ikke-negativt heltal)

fac := 1

for $i := 1$ **to** n

    fac := $i \cdot$ *fac*

return *fac*

{*fac* $= n!$}

Algoritme 1: Rekursiv beregning af n! Side 354

procedure *factorial*(n: ikke-negativt heltal)

if $n = 0$ **then return** 1

else return $n \cdot$ *factorial*$(n - 1)$

{outputtet er $n!$}

Algoritme 3:

Rekursiv beregning af støste fælles divisor Side 355

procedure gcd(a, b: heltal, hvor $0 \leq a < b$)
if $a = 0$ **then return** b
else return gcd($b \mod a, a$)
$\{$outputtet er gcd(a, b)$\}$

Algoritme 8:

Iterativ beregning af Fibonaccital

procedure iterativ fibonacci (n: ikke-negativt heltal)

if $n = 0$ **then** **return** 0

else $x := 0$

$y := 1$

for $i := 1$ **to** $n - 1$

$z := x + y$

$x := y$

$y := z$

return y

$\{y$ er det n'te Fibonacci tal.$\}$

Algoritme 7:

Rekursiv beregning af Fibonaccital Side 358

procedure fibonacci (n: ikke-negativt heltal)
if $n = 0$ **then** fibonacci(0) := 0
else
    if $n = 1$ **then** fibonacci(1) := 1
    else return fibonacci($n-1$) + fibonacci($n-2$)
{outputtet er det n'te Fibonacci tal.}

Ved rekursiv beregning af f_n beregnes f_1 i alt f_n gange.

procedure mergesort($L = a_1, \ldots, a_n$ liste af tal)
if $n > 1$ **then**
    $m := \lfloor \frac{n}{2} \rfloor$
    $L_1 := a_1, \ldots, a_m$
    $L_2 := a_{m+1}, \ldots, a_n$
    $L :=$ merge(mergesort(L_1), mergesort(L_2))

{ L er sorteret i ikke-aftagende rækkefølge}

Lemma 1 For at flette to sorterede lister med henholdsvis m og n elementer bruges højst $m+n-1$ sammenligninger.

Sætning 1. Antal sammenligninger, der i alt bruges af Mergesort for at sortere en liste med $n = 2^k$ elementer med er højst

$$2^k(k-1)+1 = n(\log(n)-1)+1.$$

Mergesort har kompleksitet $O(n\log(n))$.