DMat-06

3.1: Algoritmer

Definition 1. En algoritme er en endelig følge af præcise instruktioner til at udføre en beregning eller løse et problem.

Yderligere egenskaber for en algoritme:
input, output, præcis defineret, korrekt, endelig, hver skridt kan udføres på endelig tid, generel

Algoritme=Procedure

procedure $insertion\ sort(a_1, \ldots, a_n$: reelle tal med $n \geq 2)$
for $j := 2$ **to** n
    $i := 1$
    while $a_j > a_i$
        $i := i + 1$
    $m := a_j$
    for $k := 0$ **to** $j - i - 1$
        $a_{j-k} := a_{j-k-1}$
    $a_i := m$

$\{a_1, \ldots, a_n$ er nu i voksende rækkefølge.$\}$

3.2: Store O.

$f(x), g(x)$ funktioner der er defineret når x er et tilstrækkeligt stort helt eller reelt tal. $f(x), g(x) \in \mathbb{R}$.

Vi siger at $f(x)$ er $O(g(x))$ hvis der findes konstanter C, k (vidner) så

$$|f(x)| \leq C|g(x)|, \text{ for alle } \; x > k.$$

Vi siger at $f(x)$ er $\Omega(g(x))$ hvis der findes konstanter $C > 0, k$ så

$$|f(x)| \geq C|g(x)|, \text{ for alle } \; x > k.$$

$$f(x) \text{ er } \Omega(g(x)) \Leftrightarrow g(x) \text{ er } O(f(x)).$$

Vi siger at $f(x)$ er $\Theta(g(x))$ hvis $f(x)$ er $O(g(x))$ og $f(x)$ er $\Omega(g(x))$.

Hvis $f(x)$ er $O(g(x))$ så "vokser $f(x)$ ikke hurtigere end $g(x)$".

Hvis $f(x)$ er $\Omega(g(x))$ så "vokser $f(x)$ mindst lige hurtigt som $g(x)$".

Idé: $f(x)$ er et kompliceret udtryk, eller funktion der ikke kan beregnes præcist. $g(x)$ er et simpelt udtryk.

Vi skriver tit $f(n)$ i stedet for $f(x)$, især hvis $f(x)$ kun er defineret når x er et helt tal.

Korollar 1
Hvis $f_1(x)$ er $O(g(x))$ og $f_2(x)$ er $O(g(x))$
så er $f_1(x) + f_2(x)$ også $O(g(x))$.

Sætning 4
Lad $f(x) = a_m x^m + a_{m-1} x^{m-1} + \ldots + a_1 x + a_0$, hvor $a_m \neq 0$
($f(x)$ er altså et vilkårligt polynomium af grad m)
Så er $f(x)$ $O(x^m)$ og x^m er $O(f(x))$.
Altså: $f(x)$ er $\Theta(x^m)$.

Eksempler.

$\log x$ er $O(x)$, og $(\log x)^{1000}$ er $O(x)$.
$\log x$ vokser meget langsomt.

x^m er $O(a^x)$ for alle m og $a > 1$, men a^x er ikke $O(x^m)$.
Exponential funktioner vokser meget hurtigt; hurtigere end polynomier.

Hvis $b > a > 1$ så gælder: a^x er $O(b^x)$ men b^x er ikke $O(a^x)$.