3.3 Kompleksitet: *tids*-kompleksitet.

Eksempel: kompleksitet af algoritme lineær søgning

Algoritme 2: lineær søgning Side 196

procedure $linear\ search(x$:heltal, $a_1, \ldots, a_n$: forskellige heltal)
$i := 1$
while $i \leq n$ and $x \neq a_i$
    $i := i + 1$
if $i \leq n$ **then** $location := i$
else $location := 0$
return $location$
{ hvis $location = 0$ så er x ikke i listen, ellers er $a_{location} = x$ }

$f(n)$ = den tid det tager at finde i liste af n tal ved lineær søgning i værste tilfælde (worst-case): når vi søge gennem hele listen.

average-case

Lettere beregning (uafhængig af hvilken computer der bruges):

$f(n)$ = antal "operationer" der bruges

"operationer" kan eventuelt begrænses til sammenligninger, da det er den vigtigste operation i denne algoritme, og det er den der foretages flest gange, i den konkrete algoritme.

Ikke nødvendigt at finde $f(n)$ eksakt.
Nok at vise at $f(n)$ er $O(g(n))$. (Eller rettere $\Theta(g(n))$.)
Kompleksiteten er så: $O(g(n))$.

procedure $linear\ search(x$:heltal, $a_1, \ldots, a_n$: forskellige heltal)
$i := 1$
while $i \leq n$ and $x \neq a_i$
    $i := i + 1$
if $i \leq n$ **then** $location := i$
else $location := 0$
return $location$
{ hvis $location = 0$ så er x ikke i listen, ellers er $a_{location} = x$ }

procedure $binary\ search(x$: heltal, $a_1, \ldots, a_n$: voksende følge af heltal)
$i := 1$
$j := n$
while $i < j$
    $m := \lfloor (i + j)/2 \rfloor$
    if $x > a_m$ **then** $i := m + 1$
    else $j := m$
if $x = a_i$ **then** $location := i$
else $location := 0$
return $location$
{ hvis $location = 0$ så er x ikke i listen, ellers er $a_{location} = x$ }

procedure $bubblesort(a_1, \ldots, a_n$: reelle tal med $n \geq 2)$
for $i := 1$ **to** $n - 1$
    for $j := 1$ **to** $n - i$
        if $a_j > a_{j+1}$ **then** ombyt a_j og a_{j+1}
$\{$ $a_1, \ldots, a_n$ er nu i voksende rækkefølge$\}$