Uge 39

Ex 1

1. Er nedenstående en algoritme?

$$\begin{array}{l} i = 0 \\ \textbf{While } i \neq 5 \\ \quad i = i + 2 \end{array}$$

Mere formelt (Fra Computer Science: An Overview Brookshear, Brylow):
An algorithm is an ordered set of unambiguous, executable steps that define a terminating process.

Nej den terminerer ikke..

Ex 2

2. Betragt listen $L = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]$.

   (a) Hvor mange sammenligninger foretages der med SequentialSearch$(L, 7)$?

   (b) Hvor mange sammenligninger foretages der med BinarySearch$(L, 7)$?

   Antag nu, at L indeholder 10.000 elementer.

   (c) Hvor mange sammenligninger foretager man i værste tilfælde med en sekventiel søgning i L?

   (d) Hvor mange sammenligninger foretager man i værste tilfælde med en binær søgning i L?

a. Sequential: 7
b. Binary: 3 (10, 5, 7)
c. Sequential: 10.000
d. Binary: $\lfloor \log_2 10000 \rfloor + 1 = 14$

Ex 3

3. Udfyld de manglende felter (undtagen øverste række) i tabellen på side 10 i forelæsningsnoterne om algoritmer.

#op input str n	$1\ ms$	$1\ s$	$1\ min$	$1\ døgn$	1 år
$\log_2 n$	10^{300000}				
n	10^6	$10^6 \cdot 10^3 = 10^9$	$60 \cdot 10^9 = 6 \cdot 10^{10}$	$9 \cdot 10^{13}$	$365{,}24 \cdot 9 \cdot 10^{13} \approx 3 \cdot 10^{16}$
$n \cdot \log_2 n$	$6 \cdot 10^4$	$4 \cdot 10^7$	$2 \cdot 10^9$	$2 \cdot 10^{12}$	$6 \cdot 10^{14}$
n^2	10^3	$\sqrt{10^9} \approx 3{,}2 \cdot 10^4$	$\sqrt{6 \cdot 10^{10}} \approx 2{,}4 \cdot 10^5$	$9 \cdot 10^6$	$\sqrt{3 \cdot 10^{16}} \approx 2 \cdot 10^8$
n^3	10^2	10^3	$\sqrt[3]{6 \cdot 10^{10}} \approx 4 \cdot 10^3$	$4 \cdot 10^4$	$\sqrt[3]{3 \cdot 10^{16}} \approx 3 \cdot 10^5$
2^n	20	30	$\log_2 6 \cdot 10^{10} \approx 36$	$\log_2 9 \cdot 10^{13} \approx 46$	55

#op input str n	$1\ ms$	$1\ s$	$1\ min$	$1\ døgn$	1 år
$\log_2 n$	10^{300000}				
n	10^6	10^9	$6 \cdot 10^{10}$	$9 \cdot 10^{13}$	$3 \cdot 10^{16}$
$n \cdot \log_2 n$	$6 \cdot 10^4$	$4 \cdot 10^7$	$2 \cdot 10^9$	$2 \cdot 10^{12}$	$6 \cdot 10^{14}$
n^2	10^3	$3{,}2 \cdot 10^4$	$2{,}4 \cdot 10^5$	$9 \cdot 10^6$	$2 \cdot 10^8$
n^3	10^2	10^3	$4 \cdot 10^3$	$4 \cdot 10^4$	$3 \cdot 10^5$
2^n	20	30	36	46	55

Det vi beregner er antallet af operationer ved input af størrelse n, ved de for skellige funktioner dvs. vi kan se hvordan de mere costly funktioner kan lave færre operationer/iterationer.

Fremgangsmåden er at beregne for størrelse n og herfra relatere til de andre funktioner.

Example for n^2 for 1 døgn:

$$n^2 = 24 \cdot 60 \cdot 60 \cdot 10^9$$

$\Updownarrow$ *Ligningen løses for n vha. CAS-værktøjet WordMat.*

$$n = 9{,}3 \cdot 10^6$$

Example for getting value for $n \cdot \log_2 n$ for 1 second.

$$10^9 = n \cdot \log_2 n$$

$$n \coloneqq 4 \cdot 10^7$$

$$n \cdot \log_2 n \approx 1{,}0 \cdot 10^9$$

$$Slet\ definitioner:$$

$$3 \cdot 10^{16} = n \cdot \log_2 n$$

$$n \coloneqq 6 \cdot 10^{14}$$

$$n \cdot \log_2 n \approx 2{,}9 \cdot 10^{16}$$

Ex 4

4. Husk på algoritmerne til, ciffer for ciffer, at addere eller gange to tal i hånden. Hvis du ikke helt kan huske dem, er her et eksempel:

$$\begin{array}{rr} & 321 \\ + & 281 \\ \hline & 602 \end{array} \qquad \begin{array}{r} 321 \times 281 \\ \hline 281 \\ 562 \\ 843 \\ \hline 90201 \end{array}$$

(a) Hvad er køretiden for at addere to tal med n cifre hver? Hvad er den karakteristiske operation?

(b) Hvad er køretiden for at gange to tal med n cifre hver? Hvad er den karakteristiske operation?

a. addere

$$O(n)$$

karakteristisk operation: læg 2 tal sammen.

b. gange

$$O(n^2)$$

karakteristisk operation: Gang to tal sammen.

Ex 5

5. Hvilke af følgende udsagn er sande?

(a) $n \in O(n)$
(b) $2n + 5 \in O(n)$
(c) $\sqrt{n} - \log(n) \in O(n)$
(d) $(\log(n))^2 \in O(n \log n)$
(e) $n^2 \in O(n)$
(f) $n \in O(n^2)$
(g) $n \log(n) \in O(n^2)$
(h) $n \log(n) \in O(n)$
(i) $3n^2 + 2n + 1 \in O(n^2)$
(j) $3n^2 + 2n + 1 \in O(n)$

Definition:
$T(n) \in O(f(n))$, hvis der findes $k, m \in \mathbb{Z}$, så
$T(n) \leq k \cdot f(n)$, for alle $n \geq m$.

Regn ikke alle.

Alternativ definition:
$T(n) \in O(f(n))$, hvis der findes $k, n' \in \mathbb{Z}$, så
$\frac{T(n)}{f(n)} < k$ for $n \geq n'$

Dvs. vi skal bare finde disse konstanter.

a. $n \in O(n)$:

$$\frac{n}{n} = 1 \leq 1, \forall n \geq 1$$

b. $2n + 5 \in O(n)$

$$\frac{2n+5}{n} \leq k, \forall n \geq m$$

$$2 + \frac{5}{n} \leq 3, \forall n \geq 5$$

c. $\sqrt{n} - \log(n) \in O(n)$

$$\frac{\sqrt{n} - \log(n)}{n} \leq k, \forall n \geq m$$

$$\frac{\sqrt{n}}{n} - \frac{log(n)}{n} \leq k$$

$$n^{0,5-1} - \frac{\log(n)}{n} \leq k$$

$$n^{-0,5} - \frac{log(n)}{n} \leq k$$

$$\frac{1}{n^{0,5}} - \frac{\log(n)}{n} \leq 1, \forall n \geq 1$$

Vi ser her at venstre side er en aftagende funktion, samt f(1)=1

d. $(\log_2 n)^2 \in O(n \cdot \log_2 n)$

$$\frac{(\log_2 n)^2}{n \cdot \log_2 n} \leq k, \forall n \geq m$$

$$\frac{\log_2 n \cdot \log_2 n}{n \cdot \log_2 n} \leq k$$

$$\frac{\log_2 n}{n} \leq 1, \forall n \geq 1$$

e. $n^2 \in O(n)$

$$\frac{n^2}{n} \leq k, \forall n \geq m$$

kan ikke findes…

f. $n \in O(n^2)$

$$\frac{n}{n^2} \leq k, \forall n \geq m$$

$$\frac{1}{n} \leq 1, \forall n \geq 1$$

g. $n \cdot \log(n) \in O(n^2)$:

$$\frac{n \cdot \log(n)}{n^2} \leq k, \forall n \geq m$$

$$\frac{\log(n)}{n} \leq 1, \forall n \geq 1$$

h. $n \cdot \log n \in O(n)$

$$\frac{n \cdot \log n}{n} \leq k, \forall n \geq m$$

$$\log n \leq k$$

kan ikke findes.

i. $3n^2 + 2n + 1 \in O(n^2)$

$$\frac{3n^2 + 2n + 1}{n^2} \leq k, \forall n \geq m$$

$$3 + \frac{2}{n} + \frac{1}{n^2} \leq k$$

$$3 + \frac{2}{n} + \frac{1}{n^2} \leq 6, \forall n \geq 1$$

eller:

$$3 + \frac{2}{n} + \frac{1}{n^2} \leq 5, \forall n \geq 2$$

j. $3n^2 + 2n + 1 \in O(n)$

$$\frac{3n^2 + 2n + 1}{n} \leq k, \forall n \geq m$$

$$3n + 2 + \frac{1}{n} \leq k$$

kan ikke findes.

Ex 6

6. Betragt følgende algoritme til at finde det mindste tal i listen L.

```
MIN(L)
n = L.length
min = L[1]
For i = 2 to n
    If L[i] < min
        min = L[i]
Return min
```

(a) Hvad er algoritmens køretid?

(b) Opskriv en løkke-invariant for algoritmen, og bevis, at den altid finder det mindste element i L.

(c) Omskriv algoritmen, så den bruger en while-løkke i stedet for en for-løkke.

(d) Bemærk, at algoritmen er iterativ. Skriv en rekursiv version af algoritmen.

a. køretid:

$$O(n)$$

b. løkkeinvariant:

Efter den k'te iteration af for loopet indeholder min den mindste værdi af $L[0 \ldots k]$

Hvilket kan bevises ved induktion

basis:

før loopet er dette sandt, da k=0 og min er sat til første element.

Induktionsatagelse:

Vi antager at invarianten overholdes hver gang vi kommer til loop begyndelsen.

Induktionsskridt:

$$L[i] < min \rightarrow \text{min} \ \ opdateres$$

$$L[i] > min \rightarrow \text{min} \ \ opdateres\ ikke$$

Vi kommer ud af loopet når der ikke er flere elementer dvs. n=k. dvs. vi har det mindste element i min

c. while-løkke:

i=2

while i<n:

…

i++

d. rekursiv:

```
Min(i, L, min):
n = L.length
if n=i:
        return min
if L[i] < min:
        return Min(i+1, L, L[i])
else:
        return Min(i+1, L, min)
```

Alternativt, kan man bruge list comprehensions, og fjerne første element i listen.

Ex 7

7. Angiv køretiden for hver af nedenstående algoritmer.

(a) $i = 1$
While $i \leq n$
$i = i + 1$

(b) $i = 1$
While $i \leq n$
$i = i * 3$

(c) $i = 1$
For $k = 1$ **to** n
For $l = 1$ **to** n
$i = i + k + l$

a. $O(n)$
b. $O(\log_3(n))$

Vi ser hvad i er:

$$i = 1$$

$$3, 9, 27 \ldots$$

Dvs. antallet af iterationer er givet af:

$$3^x = n$$

Vi husker hvad log er:

$$\log_b x = k$$

$$b^k = x$$

Dvs.:

$$\log_3 3^x = \log_3(n)$$

$$x = \log_3(n)$$

hvor x er antallet af iterationer, dvs. køretiden som funktion af n

c. $O(n^2)$

Ex 8

8. Betragt følgende algoritme.

$\underline{\text{NUMBERS}}(n)$
print n
If $n < 3$
    NUMBERS$(n + 1)$
print n

Hvilken talfølge skriver NUMBERS(1)?

$1, 2, 3, 3, 2, 1$

Ex 9

9. Fibonacci-tallene er defineret således:

$$f_0 = 0$$
$$f_1 = 1$$
$$f_i = f_{i-1} + f_{i-2}, \text{ for } i \geq 2$$

Skriv en iterativ og en rekursiv algoritme, som beregner det i'te fibonacci-tal. Implementer begge algoritmer. Hvilken version er hurtigst? Hvad kan en evt. forskel i køretid skyldes?

Tegn evt recursionstræ.

Kører ca. $O(1,6^n)$ siger Rolf.

```python
def fib_recursive(i):
    if i == 0:
        return 0
    if i == 1:
        return 1
    return (fib_recursive(i - 1) + fib_recursive(i - 2))


def fib_iterative(i):
    if i == 0:
        return 0
    if i == 1:
        return 1
    cnt = 2
    fib_i1 = 0
    fib_i2 = 1
    while cnt <= i:
        tmp = fib_i2
        fib_i2 = fib_i2 + fib_i1
        fib_i1 = tmp
        cnt += 1
    return fib_i2
```

forskellen skyldes antallet af gange vi skal beregne hver fib-værdi.