

## Opgaver Uge 8

### DM507/DM578/DS814/SE4-DMAD

Opsummering fra forelæsninger: der er mindst tre naturlige niveauer at beskrive en algoritme på:

1. Som idé beskrevet i ord og tegninger.
2. Som pseudo-kode.
3. Som færdig kode (i f.eks. Python eller Java).

Dette er også de naturlige stadier at passere, når man skal udvikle algoritmer til et program i den virkelige verden. Hovedidéen i en algoritme er allerede på plads på niveau 1, og korrekthedsanalyse og asymptotisk analyse med grovsortering af algoritmer efter voksehastighed kan normalt finde sted der. Niveau 2 tilføjer flere detaljer, som skal være på plads for at sikre korrektheden af koden i sidste ende. Det kan ofte tage overraskende lang tid. Det kan betale sig at være omhyggelig med at lave korrekthedsanalyse af detaljerne (initialisering og opdatering af variable, f.eks.) i ens forslag til pseudo-kode, før man går videre til næste niveau. Hvis niveau 2 er lavet grundigt, bliver niveau 3 ofte relativt nemt.

Til forelæsningerne udtrykker vi os primært på niveau 1 og til dels på niveau 2. Lærebogen udtrykker sig også på niveau 1 og niveau 2, med de fleste algoritmer angivet meget detaljeret på niveau 2. I programmeringsopgaverne til øvelsestimerne og (for DM507) i projektet arbejder vi med niveau 3 – ofte baseret på niveau 2 fra bogen, andre gange baseret på niveau 1 og niveau 2, som man selv skal lave først.

Til øvelsestimerne er det i opgaver, hvor man skal udvikle, beskrive og/eller analysere algoritmer, *nok at gøre dette på niveau 1*, medmindre andet er angivet (dvs. medmindre man direkte bliver bedt om pseudo-kode eller Python/Java-implementation).

## A: Løses i løbet af øvelsestimerne i uge 8

1. Eksamen juni 2008, opgave 2 (tidligere eksamensopgaver kan findes øverst på kursets hjemmeside).
2. Hvilke af følgende udsagn er sande?
  - (a)  $2n + 5 \in \Theta(n)$
  - (b)  $n \in o(n^2)$
  - (c)  $n \in O(n^2)$
  - (d)  $n \in \Theta(n^2)$
  - (e)  $n \in \Omega(n^2)$
  - (f)  $n \in \omega(n^2)$
  - (g)  $\sqrt{n} \log(n) \in O(\sqrt{n})$
  - (h)  $\sqrt{n} \log(n) \in \omega(\sqrt{n})$
  - (i)  $\sqrt{n} + \log(n) \in O(n)$
  - (j)  $(\log(n))^3 \in O(n \log n)$
3. Eksamen juni 2011, opgave 2. [Hint: For et sandt svar, argumenter ud fra definitionerne på  $O$  og  $\Omega$ . For et falsk svar, find modeksempler.] Opgaven bruger notationen  $f(n) \in O(g(n))$ , hvilket betyder det samme som  $f(n) = O(g(n))$ . Se evt. diskussion i Cormen et al., 4. udgave, side 55 [Cormen et al., 3. udgave: side 45].
4. Angiv for hver af følgende to algoritmer deres asymptotiske køretid i  $\Theta$ -notation som funktion af  $n$ .

ALGORITME1( $n$ )

$s = 0$

**for**  $i = 1$  **to**  $\lfloor n/2 \rfloor$

**for**  $j = i$  **to**  $n - i$

$s = s + 1$

**return**  $s$

ALGORITME2( $n$ )

$s = 0$

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

**for**  $j = 1$  **to**  $n$

**for**  $k = 1$  **to**  $n$

$s = s + 1$

**return**  $s$

5. Implementer Mergesort i Python eller Java ud fra bogens pseudokode (Cormen et al., 4. udgave, side 36 og 39 [Cormen et al., 3. udgave: side 31 og 34]). Lad input være en liste (Python) eller et array (Java) af heltal.

For Cormen et al., 3. udgave: I denne udgave optræder værdien  $\infty$  i pseudokoden på side 31. Brug her et stort tal, og sørg for at kun heltal under denne værdi optræder i input.<sup>1</sup> Alternativt (og bedre, da man undgår at skulle tage hensyn til værdien valgt som  $\infty$ ), brug til MERGE pseudokoden fra Cormen et al., 4. udgave, som kan findes i slides om sortering. Denne pseudokode anvender ikke  $\infty$ .

Test at din kode fungerer ved at generere små arrays/lister med forskelligt indhold (f.eks. stigende, faldende, tilfældige), sortere dem og checke resultatet.

## B: Løses hjemme inden øvelsestimerne i uge 9

1. Eksamen januar 2007, opgave 3. Opgaven bruger notationen  $f(n) \in O(g(n))$ , hvilket betyder det samme som  $f(n) = O(g(n))$ .
2. Angiv for hver af følgende to algoritmer deres asymptotiske køretid i  $\Theta$ -notation som funktion af  $n$ .

ALGORITME3( $n$ )

$s = 0$

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

**for**  $j = 1$  **to**  $n$

**if**  $i == j$

**for**  $k = 1$  **to**  $n$

$s = s + 1$

**return**  $s$

ALGORITME4( $n$ )

$s = 0$

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

**for**  $j = 1$  **to**  $n$

**if**  $i != j$

**for**  $k = 1$  **to**  $n$

$s = s + 1$

**return**  $s$

3. Fortsæt opgaven ovenfor med implementation af Mergesort på denne måde:

Tilføj tidtagning af din kode. Du skal kun tage tid på selve sorteringen, ikke den del af programmet som genererer array'ets/lists indhold.

Kør derefter din kode med input, som er tilfældige heltal, samt input

---

<sup>1</sup>Faktisk har både Python og Java en indbygget værdi for  $\infty$  (i Python `float("inf")`, i Java `Double.POSITIVE_INFINITY`). Disse er teknisk set kommatall, men fungerer efter hensigten, hvis de sammenlignes med heltal. I Java kan et kommatall og et heltal dog ikke optræde i samme array, så bogens pseudokode kan ikke implementeres vha. denne værdi. I Python kan et kommatall og et heltal godt optræde i samme liste, så der kan bogens pseudokode godt implementeres vha. denne værdi.

som er sorteret og omvendt sorteret. Gør dette for mindst 5 forskellige værdier af  $n$  (antal elementer at sortere), vælg værdier som får programmet til at bruge fra ca. 100 til ca. 5000 millisekunder. Gentag hver enkelt kørsel tre gange og find gennemsnittet af antal millisekunder brugt ved de tre kørsler. Divider de fremkomne tal med  $n \log_2 n$ , og check derved, hvor godt analysen passer med praksis – de resulterende tal burde ifølge analysen være konstante.

[Hint: Se seddel med opgaver til uge 7 for informationer om bogens indeksering i forhold til Pythons/Javas, samt om metoder i Python og Java til tidstagning og til generering af tilfældige heltal. For at beregne logaritmer: I Python brug metoden `math.log(x,2)` til at beregne  $\log_2(x)$  (husk at importere `math` modulet). I Java brug metoden `java.lang.Math.log(x)`, til at beregne  $\log_e(x)$ , dvs. logaritmen med grundtal  $e$ . For at beregne  $\log_2(x)$  kan man så bruge at  $\log_2(x) = \log_e(x)/\log_e(2)$  (jvf. Cormen et al., 4. udgave, formel 3.19 side 66 [Cormen et al., 3. udgave: formel 3.15 side 56]).]

4. (\*) Cormen et al., 4. udgave, opgave 2-4 (side 47) [Cormen et al., 3. udgave: opgave 2-4 (side 41)], spørgsmål **d**.