45011 Algoritmer og datastrukturer
Løsningsforslag eksamen 10. august 1992

Oppgave 1

a) Ja.

$O(g(n))$ er asymptotisk øvre grense som ikke nødvendigvis er stram (*tight*), mens $o(g(n))$ er en øvre grense som man vet er *ikke stram*. En ikke stram øvre grense kan bli enda mindre stram når noe (positivt) legges til, men er i alle fall fortsatt en gyldig øvre grense. (Det forutsettes at $f(n) > 0$.)

b) Nei, $T(n) = \Theta(n)$

$T(n) = 3\,T(n/3) + O(\lg n) = a\,T(n/b) + f(n)$
$a = 3$, $b = 3$, $f(n) = O(\lg n)$

$c \lg n = O(n^{\log_3 3 - \epsilon})$ for $0 < \epsilon < 1$

fordi $\lg n$ vokser langsommere enn alle polynomer n^k, $k > 0$. Dette er **tilfelle 1** i masterteoremet, og

$T(n) = \Theta(n^{\log_3 3}) = \Theta(n)$

c) Ja.

SELECT (boka, s. 189) finner det i-te største/minste element på $O(n)$ tid i det verste tilfelle for *hvilken som helst* i.

Oppgave 2

Prinsippet beskrevet i 24.1 (s. 499) i boka finner alltid spenntrær med minimal dyreste kant, fordi det ved hvert valg velges den kanten *med lavest kostnad* som knytter S til $V - S$. Både Kruskals og Prims algoritmer minimaliserer altså den dyreste kanten i spenntreet.

- Kruskal:
  Inkludér den kanten med lavest kostnad som ikke skaper syklus i grafen men knytter sammen usammenhengende fragmenter av spenntreet, helt til alle gjenværende kanter skaper syklus og alle noder er forbundet.
- Prim:
  Inkludér den kanten med lavest kostnad som knytter en ny node til spenntreet, som bygges opp node for node.

Oppgave 3

a)

Sorteringen av hver av p delmengder tar $T(m)$ tid, og delmengdens størrelse er $m = n/p$. Både oppdelingen i delmengder og p-veis fletting behandler hvert element bare én gang, og tar derfor $O(n)$ tid. **if**-setningen tar konstant tid.

$$T(n) = p\,T(n/p) + O(n)$$

b)

For konstant p har vi **master-tilfelle 2**:

$$c \cdot n = \Theta(n^{\log_p p}) \quad \Rightarrow \quad T(n) = \Theta(n \lg n)$$

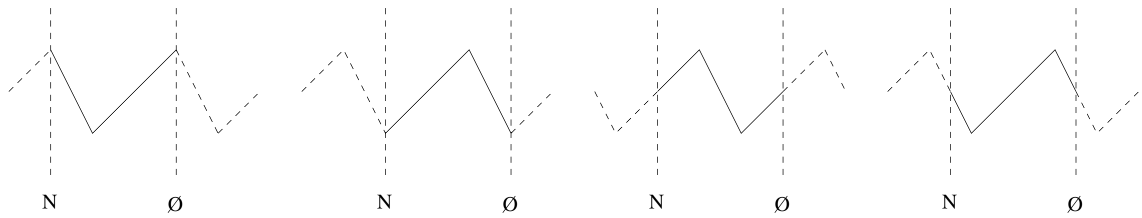
Flettingen tar egentlig $f(n) = O(n \cdot g(p))$ tid, der $g(p)$ representerer tiden det tar å velge den minste blant p elementer, som avhenger av flettealgoritmen. Med hensyn på ikke-konstant p (ikke lineær fletting) gjelder derfor ikke lenger formelen i a), og master-teoremet forutsetter også konstant p. Det er altså ikke likegyldig om p betraktes som konstant eller ikke.

For eksempel gir fletting basert på *heap*-prioritetskø $T(n) = pT(n/p) + O(n \lg p)$, mens ineffektiv fletting basert på sekvensiell søking blant de p elementene gir $T(n) = pT(n/p) + O(np)$. (Grensetilfellet er at p gjøres lik n, og da ligger hele sorteringen i flettingen av n lister av lengde 1.)

Oppgave 4

a)

Figur 1 viser unimodale sekvenser med ulike t-verdier. Sekvensen er utvidet syklisk på begge sider for å vise at den betraktes som en ring (modulo n-indekser).



Tilfelle 1 Tilfelle 2 Tilfelle 3 Tilfelle 4

Figur 1: Unimodale sekvenser med ulike t-verdier

En sekvens bestemmes som tilfelle 1, 2, 3 eller 4 ved å se på a_{n-1}, a_0 og a_1.

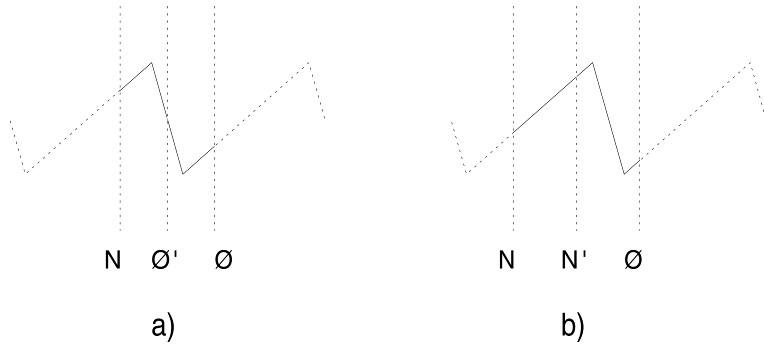
- $a_{n-1} < a_0 > a_1$: Tilfelle 1. a_0 er trivielt største verdi, $O(1)$.
- $a_{n-1} > a_0 < a_1$: Tilfelle 2.
- $a_{n-1} < a_0 < a_1$: Tilfelle 3.
- $a_{n-1} > a_0 > a_1$: Tilfelle 4.

For tilfelle 2, 3 og 4 benyttes en variant av binær søking (et intervall deles og delepunktet blir ny nedre eller øvre grense) for å finne toppunktet. For tilfelle 2 søkes ved å teste om delepunktet ligger på den stigende ($a_i < a_{i+1}$) eller fallende siden ($a_i > a_{i+1}$), og henholdsvis nedre eller øvre grense settes til delepunktet.

For tilfelle 3 og 4 testes først om delepunktet ligger på linjestykket med motsatt helling av hellingen i grensene (se fig. 2a). I så fall ligger toppunktet og bunnpunktet på hver sin side av delepunktet, og punktet brukes som en ny grense på slik måte at *toppunktet* fortsatt ligger mellom grensene (og bunnpunktet utenfor). Resten av søket gjøres som for tilfelle 2.

Dersom delepunktet ligger på samme linjestykke som en av grensene i tilfelle 3 (fig. 1b), må verdien i delepunktet testes mot verdien i en av grensene. Hvis delepunktets verdi er *større* enn grenseverdien, ligger delepunktet til *venstre* for ekstremalpunktene, og *nedre* grense flyttes til delepunktet. I motsatt fall flyttes øvre grense. Både toppunktet og bunnpunktet forblir mellom grensene, og vi har fortsatt tilfelle 3. Tilfelle 4 behandles tilsvarende.

Til slutt har man funnet det ene tallet som har stigning til venstre for seg og fall til høyre, og dette tallet må være det største.



Figur 2: Binærsøk, to typer delepunkt for tilfelle 3.

b)

Å teste et punkts stigning/fall og verdi mot grensene og flytte nedre eller øvre grense, gjøres i konstant tid. Intervallet *halveres* for hvert punkt som testes.

Tidsforbruket er derfor $T(n) = O(\lg n)$.