Løsningsforslag teoriøving 3

Du måtte klare 10/16 oppgaver for å få øvingen godkjent. Det beste av to forsøk er tellende. Finner du feil, mangler eller forbedringer, ta gjerne kontkat!

Oppgave 1
Sorteringsalgoritmer: Vi har en usortert liste med n elementer, og vi ønsker å finne ut hvor mange unike tall det er i lista. Etter algoritmene vi har lært til og med forelesningen om splitt og hersk, hva er den raskeste kjøretiden vi kan garantere for dette problemet (den beste worst-case-kjøretiden)?

Løsningsforslag: Den beste worst-case kjøretiden vi kjenner for sortering så langt er $\Theta(n \log n)$ fra merge sort. Den mer viderekommene kan argumentere med at worst-case for sammenligningsbasert sortering må være $\Omega(n \log n)$, og dermed at merge sort gir den best mulige worst-case kjøretiden.

Oppgave 2
Sorteringsalgoritmer: Denne oppgaven handler om quicksort og mergesort. Med sorteringsarbeid menes her den faktiske flyttingen av tall som en sorteringsalgoritme gjør. Hvilket av følgende alternativ er sant?

Løsningsforslag: Quicksort benytter seg av partition som flytter tallene på rett side av pivot-elementet, før den utfører rekursive kall. Mergesort tar først og utfører rekursive kall på hver halvdel av listen man ønsker å sortere, før den flytter tallene i sortert rekkefølge under flette-steget. Ergo er påstanden «Quicksort gjør sorteringsarbeidet før det rekursive kallet, mens mergesort gjør det etterpå» korrekt.

Oppgave 3
Sorteringsalgoritmer: La liste A være en liste sortert i stigende rekkefølge, og B en liste sortert i synkende rekkefølge. Hvilken påstand stemmer da om kjøretiden til quicksort?

Løsningsforslag: I begge tilfeller får vi rekkurensen $T(n) = T(n-1)+\Theta(n)$ som har løsning $T(n) = \Theta(n^2)$. I A sitt tilfelle vil partition ikke flytte noen elementer og vi ender opp med å kjøre quicksort på de $n-1$ første elementene. I B sitt tilfelle vil partition flytte det bakerste tallet først i listen, og vi vil da kjøre quicksort på de $n-1$ siste elementene.

Oppgave 4
Sorteringsalgoritmer: Hvilket alternativ under er korrekt?

Løsningsforslag: Korrekt svar er «Quicksort er en inplace sorteringsalgoritme».

Mergesort har kjøretid $O(n \log n)$ og kan dermed ikke ha værre kjøretid enn dette.
Mergesort krever også hjelpe lister under flette-steget og er dermed ikke en in-place sorteringsalgoritme.
Quicksort har worst-case $\Theta(n^2)$ som sett i oppgave 3.

Oppgave 5
Sorteringsalgoritmer: Følgende er sant for randomized quicksort

Løsningsforslag: Korrekt svar er «en sortert liste vil ikke alltid ha kjøretid $\Theta(n^2)$». Pivot-elementet velges tilfeldig i listen og bytter plass med bakerste element, før vi så utfører partition på vanlig måte.

Dette gjør at vi kan få «jevne» partisjoner også på lister som allerede er sorterte, og at vi dermed unngår at kjøretiden alltid er $\Theta(n^2)$.

Oppgave 6
Substitusjonsmetoden: Du ønsker å teste om kjøretiden til fire ulike, rekursive algoritmer er $O(n^2)$ ved hjelp av substitusjonsmetoden. Først setter du opp rekurrenser for algoritmene, og så forutsetter du for hver av dem den induktive hypotesen at $T(n) \leq c \cdot n^2$. Etter å ha gjennomført substitusjonsmetoden for hver av rekurrensene får du resultatene

$T1(n) \leq c \cdot n^2 - 5n$
$T2(n) \leq c \cdot n^2 + 5n$
$T3(n) \leq c \cdot n^2 + 1$
$T4(n) \leq c \cdot n^2 - 1$

Hvilke(n) av algoritmene har du greid å bevise at har kjøretid $O(n^2)$? Anta at grunntilfellene i den matematiske induksjonen også stemmer.

Løsningsforslag: Korrekt svar er $T1$ og $T4$. For å ha et gyldig induktivt bevis må vi bevise at vår induktiv hypotese $T(n) \leq cn^2$ er gyldig for n hvis den er gyldig for $n' < n$. Eksempelvis i rekurrensen $T(n) = T(n-1) + n$, antar vi at $T(n-1) \leq c(n-1)^2$ og så viser at $T(n) \leq cn^2$ gitt denne antagelsen. I vårt tilfelle ser vi at kun resultatene til $T1(n)$ og $T4(n)$ har høyresider som er $\leq cn^2$. $T2$ og $T3$ sine høyresider er $> cn^2$, noe som gjør at vi ikke kan bekrefte at vår induktivehypotese er gyldig for samtlige n.

Oppgave 7
Masterteoremet: La $T(n) = 27 \cdot T(n/3) + n^3$. Hvilket tilfelle tilhører rekurrensen når du benytter master-teoremet?

Løsningsforslag: Korrekt svar er case 2. Her må man kjenne masterteoremets tre caser fra læreboken. $f(n) = n^3$ og $\log_b a = \log_3 27 = 3$ og da får vi at $n^{\log_b a} = n^3$. Ergo er $f(n) = \Theta(n^{\log_b a})$ noe som betyr at case 2 er gjeldende.

Oppgave 8
Masterteoremet: La $T(n) = 27 \cdot T(n/3) + n^3$. Løs rekurrensen.

Løsningsforslag: Korrekt svar er $\Theta(n^3 \log n)$. Her benytter vi oss av løsningen fra forrige oppgave som sier at vi skal benytte case 2. Da følger det at $T(n) = \Theta(n^{\log_b a} \log n) = \Theta(n^3 \log n)$

Oppgave 9
Masterteoremet: La $T(n) = 4 \cdot T(n/2) + n^3$. Løs rekurrensen.

Løsningsforslag: Korrekt svar er $\Theta(n^3)$. Vi ser at masterteoremet kan benyttes siden rekurrensen er på formen $T(n) = aT(n/b) + f(n)$. Vi må først avgjøre hvilket case som er gjeldende. $\log_b a = \log_2 4 = 2$, noe som gir at $f(n) = n^3 = \Omega(n^{\log_b a + \epsilon}) = \Omega(n^{2+\epsilon})$, der $\epsilon \in (0, 1]$. Dermed er case tre det eneste mulige caset, dog har dette caset en ekstra betingelse som må undersøkes, nemlig $af(\frac{n}{b}) < cf(n)$ for $c < 1$ og stor nok n. $af(\frac{n}{b}) = 2\left(\frac{n}{2}\right)^3 = \frac{n^3}{4} < cn^3$. Denne ulikheten holder for $c \in (\frac{1}{4}, 1)$, der øvre grense på intervallet kommer fra betingelsen $c < 1$. Dermed er case tre den gjeldende casen, og løsningen blir da

$T(n) = \Theta(f(n)) = \Theta(n^3)$

Oppgave 10
Rekursjonstrær: La $T(n) = T(n/10) + T(n/5) + T(n/\pi) + n^3$ der $T(1) = 1$. Hva blir høyden til rekursjonstreet?

Løsningsforslag: Korrekt svar er $\Theta(\log n)$. For å finne høyden må vi finne den lengste kjeden av kall. $T(n/\pi)$ vil være leddet som gir opphav til denne lengste kjeden, siden $\pi < 5 < 10$ og vil dermed redusere n minst for hvert rekkursivekall. Etter k kall har vi at $T(n_k/\pi) = T(\frac{n}{\pi^k})$. Kjeden stopper når $T(\frac{n}{\pi^k}) = T(1)$, ergo $\frac{n}{\pi^k} = 1$. Noe som gir $k = \log_\pi n = \Theta(\log n)$. Den observante vil se at rekurrenser på formen $T(n) = aT(n/b) + f(n)$ har høyde $\Theta(\log n)$, og dermed at vi ikke trenger å finne den lengste kjeden, siden alle kjedene har samme asymptotiske høyde.

Oppgave 11
Variabelskifte: Løs rekurensen gitt ved $T(n) = 4T(\sqrt{n}) + (\log n)^2$ ved hjelp av variabelskifte.

Hint: $\sqrt{n}$ er det samme som $n^{\frac{1}{2}}$.

Løsningsforslag: Korrekt svar: $T(n) = \Theta((\log n)^2 \log\log n)$. Sett $m = \lg n$ noe som gir $2^m = n$. Rekurrensen kan da skrives som $T(2^m) = 4T\left((2^m)^{\frac{1}{2}}\right) + m^2 = 4T\left(2^{\frac{m}{2}}\right) + m^2$. Vi definerer så $S(m) = T(n) = T(2^m)$ noe som gir $S(m) = 4S(\frac{m}{2}) + m^2$. Vi ser at masterteoremet case 2 gjelder (se oppgave 7 for forklaring) og at løsningen dermed blir $T(n) = S(m) = \Theta(m^2 \log m) = \Theta((\log n)^2 \log\log n)$.

Oppgave 12
Rekurrens i kode: Funksjonen gjoerNoe(n) under har kjøretid $\Theta(\sqrt{n})$. Hva blir kjøretiden til funksjonen $f1(n)$?

Hint: Sett opp to rekurrenser $T1(n)$ og $T2(n)$ og finn først en løsning på lukket form for $T1(n)$

Løsningsforslag: Korrekt svar: $T(n) = \Theta(n^{\log_3 2})$. Vi setter opp to rekkurenser,

$T1(n) = T1(n/3) + T2(n-2) + \Theta(\sqrt{n})$, og $T2(n) = T1(n/3) + \Theta(\sqrt{n})$. Merk at i $T2(n)$, blir $T1(n/3)$ kun kallt en gang, men resultatet av funksjonen blir multiplisert med to. Vi setter inn i $T1$ og får $T1(n) = T1(n/3) + T1(\frac{n}{3} - \frac{2}{3}) + \Theta(\sqrt{n})$. Vi ignorer mindre konstantledd inne i T1, noe som gir $T1(n) = 2T1(n/3) + \Theta(\sqrt{n})$. Merk at $\log_3 2 > \frac{1}{2}$, så masterteoremet case 1 er gjeldende. Dette gir $T(n) = \Theta(n^{\log_3 2})$.

Oppgave 13
Rekurrens i kode:Hva blir kjøretiden til funksjonen f3(n)?

Løsningsforslag: Korrekt svar: $T(n) = \mathrm{O}(n)$. Her får vi rekurrensen

$T(n) = T(n-42) + \mathrm{O}(1), T(n' \leq 42) = \mathrm{O}(1)$. Siden $f(n) = \mathrm{O}(1)$ (arbeidet per iterasjon) trenger vi kun å se på hvor mange ganger vi trenger å utvide $T(n-42)$ før $n' - 42 \leq 42$ la oss kalle dette antallet k. Da følger det at $n - 42k = 42$ noe som gir $k = \frac{n}{42} - 1$, som igjen gir $k = \mathrm{O}(n)$. Dermed må $T(n) = \mathrm{O}(n)$.

Oppgave 14
Rekurrens i kode: Hva blir kjøretiden til funksjonen $f4(n)$? println tar konstant tid.

Løsningsforslag: Korrekt svar: $T(n) = \mathrm{O}(n^2 \log n)$. Den første løkken utfører $\Theta(n^2)$ arbeid, dette kan ses ved at rekken $n+(n-1)+(n-2)+...+2+1 = \frac{n\cdot(n+1)}{2} = \Theta(n^2)$. Neste løkke kaller så $f4(n/6)$ 36 ganger. Vi får dermed rekurrensen $T(n) = 36T(n/6) + \Theta(n^2)$, som kan løses ved masterteoremet case 2. Vi får dermed $T(n) = \mathrm{O}(n^2 \log n)$

Hva blir kjøretiden til funksjonen f4(n)? println tar konstant tid.

Oppgave 15
Rekurrens i kode: Funksjonen gjoerNoeAnnet(n) under har kjøretid $\Theta(n^2)$. Hva blir kjøretiden til funksjonen $f5(n)$?

Løsningsforslag: Korrekt svar: $T(n) = \mathrm{O}(n^2 \log n)$. Hvis vi ser bort i fra små konstantledd og husker at å gange et resultat med to ikke kaller funksjonen to ganger, så får vi rekurrensen $T(n) = 4T(n/2)+\Theta(n^2)$. Vi kan da bruke masterteoremet case 2, og får at $T(n) = \mathrm{O}(n^2 \log n)$.

Oppgave 16
Variabelskifte/ukas nøtt: Løs rekurensen gitt ved $\sqrt{\lg n} + 2T(\sqrt[\pi]{n}) = T(\sqrt[e]{n}) - \lg\lg n$ ved hjelp av variabelskifte. Hva blir kjøretiden?

Løsningsforslag: Korrekt svar: $T(n) = \Theta((\log n)^{\log_{\pi/e} 2})$. Sett først $r = n^e$, vi får da
$\sqrt{e \lg r} + 2T(r^{\frac{e}{\pi}}) = T(r) - \lg e - \lg\lg r$. Ignorer konstanter og løs for $T(r)$. Dette gir

$T(r) = 2T(r^{\frac{e}{\pi}}) + \sqrt{\lg r} + \lg\lg r$. Sett så $m = \lg r$, som gir
$T(r) = T(2^m) = 2T(2^{\frac{e\cdot m}{\pi}}) + \sqrt{m} + \lg m = 2T(2^{\frac{e\cdot m}{\pi}}) + \Theta(\sqrt{m})$. Så setter vi $T(r) = S(m)$ som gir $S(m) = 2S(\frac{m}{\pi/e}) + \Theta(\sqrt{m})$. $\log_{\frac{\pi}{e}} 2 > \frac{1}{2}$, ergo kan masterteoremet case 1 benyttes. $S(m) = \Theta(m^{\log_{\pi/e} 2})$. Vi får da $T(r) = \Theta((\log r)^{\log_{\pi/e} 2})$ og følgelig $T(n) = \Theta((\log n^{1/e})^{\log_{\pi/e} 2}) = \Theta((\log n)^{\log_{\pi/e} 2}$