DMA uge 6, løsninger til nogle af fredagens opgaver

Nogle af disse løsninger er skrevet ned meget kortfattet. Her må man selv lave "mellemregningerne". Hensigten er at man kan tjekke at man ikke er helt ved siden af.

Der kan også have sneget sig fejl ind. Hvis du mener der er en fejl, så rapportér det på Discussions på kursussiden, så det kan blive afklaret og andre kan få glæde af det, hvis du har ret.

1 (a) Nej, INSERTION-SORT laver aldrig en overflødig sammenligning. Betragt situationen hvor vi sammenligner elementet key med $A[i]$. Vi ved da kun at $key < A[i']$ for alle $i' \in \{i+1, \ldots, j-1\}$, hvor

$$A[1] \leq A[2] \leq \ldots \leq A[j].$$

Ud fra dette kan vi ikke fastlægge resultatet af sammenligningen $A[i] > key$.

1 (b) Nej, MERGE-SORT laver aldrig en overflødig sammenligning. Vi laver sammenligninger i linje 13, CLRS side 31. Her er det både muligt at $L[i] \leq R[i]$ og $L[i] > R[i]$.

1 (c) Ja, HEAP-SORT laver ofte overflødige sammenligninger. Et eksempel er når $A = [2, 1, 3]$. Da vil 2 sammenlignes med 1 både når vi kalder BUILD-MAX-HEAP og når vi kører MAX-HEAPIFY.

1 (c) Når vi når et blad i beslutningstræet, skal vi have fastslået den sorterede rækkefølge. Dette kræver at vi har lavet mindst $n-1$ sammenligninger, da dette er antallet af sammenligninger som er nødvendige for overhovedet at bekræfte at et array er sorteret.

2 (d) Vi har $\lg(n!/2) = \Omega(n \log n)$, så hvis vi kigger på halvdelen af de $n!$ input, må der være nogle som fører til en sti af længde $\Omega(n \log n)$ i beslutningstræet, ved at følge et argument ligesom det fra CLRS eller forelæsningen.

Ligeledes har vi $\lg(n!/n) = \lg((n-1)!) = \Omega(n \log n)$, så der må igen være nogle input der fører til en sti af længde $\Omega(n \log n)$.

Vi har $\frac{n!}{2^n} = \frac{n}{2} \cdot \frac{n-1}{2} \cdot \frac{n-1}{2} \cdots \frac{1}{2} \geq \frac{n}{2} \cdot \frac{n-1}{2} \cdot \frac{n-1}{2} \cdots \frac{n/2}{2} \geq (\frac{n}{4})^{n/2}$. Vi får da $\lg(\frac{n!}{2^n}) \geq \lg((\frac{n}{4})^{n/2}) = (n/2) \lg \frac{n}{4} = \Omega(n \log n)$. Dermed kan der ikke være en brøkdel af $1/2^n$ af de mulige input som alle har lineær dybde.

3 (a) Vi viser ved induktion at kaldet `SlowSort(A, i, j)` fører til at $A[i \ldots j]$ bliver sorteret. Basistilfældet er når $i == j$. Her er $A[i \ldots j] = A[i]$ altså et enkelt tal, som trivielt er sorteret. Antag induktivt at hvis $j - i + 1 \leq n - 1$, så vil kaldet `SlowSort(A, i, j)` føre til at $A[i \ldots j]$ bliver sorteret. Betragt nu tilfældet hvor $j - i + 1 = n$. Induktionsantagelsen giver at kaldene `SlowSort(A, i, m)` og `SlowSort(A, m+1, j)` fører til at $A[i \ldots m]$ og $A[m+1 \ldots j]$ bliver sorteret. Derfor er det største tal i $A[i \ldots j]$ gemt på plads $A[m]$ eller $A[j]$ efter disse to kald. Hvis disse tal er i omvendt rækkefølge bytter vi dem om, så derfor ender vi med at have det største tal på den rigtige plads $A[j]$. Dernæst kalder vi funktionen rekursivt på arrayet $A[i \ldots j-1]$. Induktionsantagelsen giver så at hele $A[i \ldots j]$ er sorteret til sidst.

3 (b) Vi laver to rekursive kald på arrays af størrelse $n/2$ og dernæst et på et array af størrelse $n-1$. Heraf følger rekursionsligningen for køretiden.

3 (c) Når $n > 2$, laver vi tre rekursive kald til arrays af størrelse $2n/3$. Heraf følger rekursionsligningen.

3 (d) Basistilfældet er $n \leq 2$. Her fører det første if-statement til at A bliver sorteret. Antag nu at hvis vi kalder `StoogeSort` på et delarray $A[i \ldots j]$ af længde højst $j - i + 1 \leq n - 1$, så vil funktionen sortere vores array korrekt. Betragt et tilfælde hvor vi kalder `StoogeSort(A, i, j)` og $j - i + 1 = n$. Lad A_0, A_1, A_2, A_3 være vores array A efter hhv. $0, 1, 2, 3$ rekursive kald. Det første rekursive kald fører til at de t største værdier i $A_0[i \ldots j - t]$ er placeret i $A_1[i \ldots j - 2t + 1]$. Derfor fører det andet rekursive kald til at de t største værdier i hele $A_0[i \ldots j]$ er placeret i $A_2[j - t + 1 \ldots j]$, og at disse værdier er i sorteret rækkefølge. Det tredje rekursive kald sorterer alle de tal som er mindre, således at $A_3[i \ldots j - t]$ også vil være sorteret. Dermed er hele A_3 sorteret.

3 (e) Definér $c_3 = 3c_1 + c_2$. Da gælder for $n \leq 3$ at $T(n) \leq c_3 \cdot n^3$. Antag induktivt at $T(n) \leq c_3 \cdot n^3$ når $n < N$, hvor $N \geq 4$. Vi har da $T(N) = 3T(2N/3) + c_2 = 8c_3N^3/9 + c_2$. Idet $c_2 < c_2 \cdot 4^3/9 \leq c_2N^3/9 < c_3N^3/9$, får vi $T(N) = 8c_3N^3/9 + c_2 < c_3N^3$.