Assignment - I

Shubh Maheshwari
20161170
Algorithm

Ans-1) a) $f(n) = O(g(n))$
$= c \times g(n)$ {c = constant}

$0 \leq f(n) \leq c\,g(n)$, c & $n_0 > 0$ $\forall n > n_0$

$0 \leq (f(n))^2 \leq c^2 (g(n))^2$, $c^2 = k$

$0 \leq f(n)^2 \leq k(g(n))^2$

hence $f(n) = O(g(n)^2)$
Hence proved.

b) $f(n) + g(n) = \Theta(\min(f(n), g(n)))$

Let $f(n) = n$ $g(n) = n^2$
where $g(n) > f(n)$ $\forall n > n_0$

Thus $0 \leq c_1 f(n) \leq f(n) + g(n) \leq c_2 f(n)$

$0 \leq c_1 n \leq n + n^2 \leq c_2 n$

for $c_1 = 1$

$n \leq n + n^2$ but $n + n^2 \leq c_2 n$

$n \leq c_2 - 1$

As this is not true for $n \to \infty$

The statement is not true.

(c) Let $f(n) = n$ $g(n) = n^2$ for $n_0 = 1$

if $f(n) = \Omega(g(n))$ $0 \le c g(n) \le f(n)$

if $n = \Omega(n^2) \Rightarrow 0$

$0 \le c_1 n^2 \le c_2 n$

$0 \le n \le c_2$

hence for $n \to \infty$ $f(n) \ne \Omega(g(n))$

Hence this is disproved.

(d) $\min\{f(n), g(n)\} \in O(f(n) + g(n))$

Let us assume that both the functions are asymtotically increasing and are positive beyond n_0.

Let $\forall n > n_0$

if $f(n) \le g(n)$

$0 \le 2f(n) \le f(n) + g(n)$

if $g(n) \le f(n)$

$0 \le 2g(n) \le f(n) + g(n)$

Hence $\min(f(n), g(n)) = O(f(n) + g(n))$

Ans-2) $\log(n!) = \Theta(n \log n)$

$\log n! = \log n(n-1) \cdots 1$

$= \log n + \log(n-1) \cdots \log(1)$

$\log(n) \geq \log(k)$ if $n > k$ and $n \& k \geq 1$

$$\underbrace{\log(n) + \log n + \log(n) \cdots}_{n \text{ times}} \geq \log(n!)$$

$$n\log(n) \geq \log(n!)$$

Hence $\log(n!) = \Theta(n\log n)$

3) Both Sorts → Insertion or Merge Sort can be stable or unstable based on the algorithm used.

Array $\boxed{2\,3\,4\,3\,6}$

A sort is called unstable if it replaces the position of same elements. The eg 3 4 3 / 1 2 3

if the final value after sorting is

3 3 4 , Hence even though index 1 and
3 1 2

3 have same value their position get replaced in the final answer.

Depending on the preference given during sort it could be stable or unstable algorithm

Ans - 4)

To prove $(n+a)^3 = \Theta(n^3)$; ~~also~~ if not find the conditions in which they are true.

if $\Theta(n^b) = (n+a)^b$ for some c_1, c_2 then

$$c_2 n^b \le (n+a)^b \le c_1 n^b \quad \forall n \ge n_0$$

$$n+a \le n+|a| \le 2n$$

$$n+a \ge n-|a| \ge \frac{1}{2}n$$

Thus when $n \ge 2a$

$$0 \le \frac{1}{2}n \le n+a \le 2n$$

for $b>0$ inequality holds when raised.

$$0 \le \left(\frac{1}{2}n\right)^b \le (n+a)^b \le 2^b n^b$$

$c_1 = \left(\frac{1}{2}\right)^b \quad c_2 = 2^b$ for $n_0 = 2|a|$

These values satisfy the desired equations.

Q5) $T(n) = 2T(\sqrt{n}) + 1 \quad T(1) = 1$

Let $n = 2^m$

$$T(2^m) = 2T(2^{m/2}) + 1$$

~~for~~ Let $K(m) = T(2^m)$

Scanned by CamScanner

$K(m) = 2K(m/2) + 1$

Using masters theorm

$K(m) = \Theta(m \log^2)$

$K(m) = \Theta(m)$

$T(2^m) = \Theta \log_2 n$

$T(2^{\log n}) = \Theta(\log n)$

$T(n) = \Theta(\log n)$

Q6) a) False, as we are given

$T(n) \leq c f(n)$ $\forall n \geq n_0$ but we cannot

say that $\forall n > n_0$ $T(n) \geq 0$

(b) If $T(n) = \Omega f(n)$

$0 \leq c f(n) \leq T(n)$ $\forall n \geq n_0$ for some c

$\Rightarrow$ This Statement is false

(c) $T(n) = \Theta(f(n))$

$T(n) = \Omega f(n)$

$0 \leq T(n) \leq c_1 f(n)$ for some c_1, n_0.

$T(n) = \Omega f(n)$

$0 \leq c_2 f(n) \leq T(n)$ for some c_2, n_2

$n \to n_2$

Using the above Statements

$0 < c_2 f(n) \le T(n) \le c_n f(n)$

for some c_1, c_2

$n > \max(n_1, n_2)$

$\Rightarrow T(n) = \Theta f(n$

Hence the ~~equation~~ statement is true

(d) $T(n) = o(f(n))$ & $T(n) = \Theta(f(n))$

$0 \le c_1 f(n) \le T(n) \le c_2 f(n)$ for $c_1, c_2 \ge 0$ and $n > n_0$

$\Rightarrow T(n) \geq c_1 f(n)$

but for $f(n) = o(f(n))$

$T(n) \le c f(n) \quad \forall c > 0$

$T(n) \ne \Theta(f(n))$

Ans 7) $T(n) = T(n/10) + T\left(\frac{9n}{10}\right) + n$

$$
\begin{array}{cccc}
 & cn & & \\
c(n/10) & & c\left(\frac{9n}{10}\right) & \\
c\left(\frac{n}{10\times10}\right) & c\left(\frac{n\times9}{10\times10}\right) & c\left(\frac{9n}{10\times10}\right) & c\left(\frac{81}{100}n\right)
\end{array}
$$

The longest path is $n \to \frac{9n}{10} \to \frac{9^2 n}{10^2} \to \frac{9^3 n}{10^3} \ldots$

$\left(\frac{9}{10}\right)^k n = 1 \qquad \log_{(10/9)} n =$ height of the tree

Ans $O((n \log_{\frac{10}{9}} n) = O(n \log_{10/9} n)$

A8) We can find the median of 2 sorted in $O\log(2n) = O\log n$

eg $\boxed{1\,3\,5}$ for finding median.

$\boxed{2\,4\,6}$ we can compare middle elements of both sorted arrangments.

Now, we can safely claim that no. to the left ~~to~~ is smaller ~~that~~ than the number of elements to the right.

so we divide it into 2 parts

eg $\boxed{1\,3\,5} \to \boxed{3\,5}$ ~~Now as only~~ Now

$\boxed{2\,4\,6} \to \boxed{2\,4}$ we keep on doing

this until we have 2 elements and we find max (arr1[0], arr2[0]) & min (arr1[1], arr2[1]) as our median.

A(f) $n^{1.004} = O(n \log n)$

$0 \leq c\, n \log n \leq n^{1}$

$C \leq \frac{n^{1.004}}{n \log n}$ $\quad C \leq \frac{n^{0.004}}{\log n}$

$C \leq \lim_{n \to \infty} \frac{n^{1/250}}{\log n} \leq \frac{n^{-\frac{249}{250}}}{1/n}$

$\leq \lim_{n \to \infty} n^{1/250}$

This is not true. Hence $n^{1.004} \neq O(n \log n)$

A(10) In general large unsorted cases, merge sort would always give complexity of $O(n \log n)$ which is better than insertion sort $O(n^2)$

(a) If the array is almost sorted, insertion sort would be close to the linear complexity whereas merge sort will be $O(n \log n)$. Hence insertion sort is better

(b) We can improve merge sort by using n/k lists of k elements and sort it. Hence we use insertion sort for k elements followed by the merge sort procedure.

$O(\frac{n}{k} \cdot k^2) = O(n \cdot k)$

depth of the tree $= \log n - \log k = \log(n/k)$

So overall complexity $= O(c_1 nk + 2n \log n/k)$

$= O(nk + \log n/k)$