Assignment 1 - Theory Part

Mandeep
Singh
200312488
CS340

Assignment #1

Problem 1:

Solution:

$2/N$, 37, $\sqrt{N}$, N, $N \log\log N$, $N\log^2 N$, $N \log N$, $N \log(N^2)$, $N^{1.5}$, N^2, $N^2 \log N$, N^3, $2^{N/2}$, 2^N

$N \log(N)$ and $N \log(N^2)$ grow at the same rate:

$\Rightarrow N \log N^2 = 2N \log N = O(N \log N)$ #

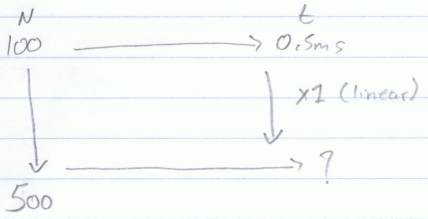
$\Rightarrow N \log\log N = N \log^2 N$

Problem 3:

a) $O(N)$
b) $O(N^2)$
c) $O(N^3)$
d) $O(N^2)$
e) $N \cdot N^2 \cdot N^2 = N^5 \Rightarrow O(N^5)$
f) $O(N^2) \cdot O(N^2) = O(N^4)$

Problem 4:

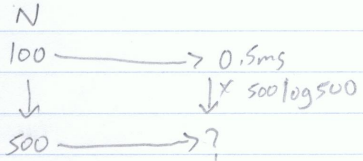
a)



$$\frac{100}{0.5ms} = \frac{500}{t}$$

$$t = \frac{500}{100} \times 0.5ms = \boxed{2.5ms}$$

b)



$$\frac{100\log 100}{0.5ms} = \frac{500\log 500}{t}$$

$$t = \frac{500\log 500}{100\log 100} \times 0.5ms$$

$t = 3.3737ms$
$t = 3.4ms$

c) $\frac{100^2}{0.5ms} = \frac{500^2}{t} \Rightarrow t = \frac{500^2}{100^2} \times 0.5ms = \boxed{12.5ms}$

d) $\frac{100^3}{0.5ms} = \frac{500^3}{t} \Rightarrow t = \left(\frac{500}{100}\right)^3 \times 0.5ms = \boxed{62.5ms}$

Problems:

a) $100 \longrightarrow 0.5\,ms$

$\downarrow \qquad \downarrow$

$?N \longrightarrow 60\times10^{-3}\,ms$

$$\frac{100}{N} = \frac{0.5\,ms}{60\times10^{-3}\,ms} \Rightarrow N = \frac{60\times10^{-3}\,ms}{0.5\,ms}\times 100$$

$$\boxed{N = 12\,000\,000}$$

b) $$\frac{100\log 100}{N\log N} = \frac{0.5\,ms}{60\times10^{-3}\,ms} \Rightarrow N\log N = \frac{60\times10^{-3}\,ms}{0.5\,ms}\times 100\log 100$$

$$\log(N^2) = 24\,000\,000$$

$$N^2 = e^{24\,000\,000}$$

c) $$\frac{100^2}{N^2} = \frac{0.5\,ms}{60\times10^{-3}\,ms} \Rightarrow N = \sqrt{\frac{60\times10^{-3}\,ms}{0.5\,ms}\times 100^2} \Rightarrow \boxed{N = 34641}$$

d) $$\frac{100^3}{N^3} = \frac{0.5\,ms}{60\times10^{-3}\,ms} \Rightarrow N = \sqrt[3]{\frac{60\times10^{-3}\,ms}{0.5\,ms}\times 100^3} \Rightarrow \boxed{N = 4932}$$

Problem 6:

a) If we implement F_n as follows:

$F(N) = F(N-1) * F(N-1)$, then the different calls corresponds to a binary tree. The complexity would be exponential, $O(2^n)$.

However, if we implemented F_n as:

$F(N) =$ power$(F(N-1), 2)$, we only have $O(N)$.

b) $F_n = 2^{2^{(n-1)}}$

We can attempt to prove this by induction:

$F_1 = 2^{2^{(n-1)}} = 2^{2^{(1-1)}} = 2^1 = 2$

$F_2 = 2^{2^{(2-1)}} = 2^{2^1} = 4$ Let $n+1 = k$.

$F_k = 2^{2^{(k-1)}} = F_n \therefore \boxed{F_n = 2^{2^{(n-1)}}}$ #

This will have a run-time of:

$O(\log\log N)$ #

Problem 7:

a) similar (*tree1, *tree2) {

```
    while (tree1 != NULL && tree2 != NULL) {
        if (tree1 -> node == tree2 -> node) {
            similar (tree1 -> left, tree2 -> left);
            similar (tree1 -> right, tree2 -> right);
        }
    }
}
```

b) Tree traversal => O(N)