Homework 2

1]
```
c = 0
i = n
while i > 1 do
    for j = 1 to i do
        c = c + 1
    end for
    i = floor(i/2)
end while
return c
```

→ There are 'i' operations to be done in the for loop & while loop ends when $i = 1$.
When $n = 4$
while loops runs for $i = 4$, $i = 2$. i.e 2 times whereas the for loop runs for $j = 1$, $j = 2$, $j = 3$, $j = 4$ and again $j = 1$ & $j = 2$.

So the total time is

$$n + \frac{n}{2} + \frac{n}{4} + \dots \leq 2n$$

$\Rightarrow O(n)$ → this would be the tight bound.
&

it would be $O(n \log n)$ when

it is not a tight upper bound.

2] ~~$2^{\log n}$~~

Exponential → 2^{3n}, $n^{n \log n}$, n^{n^2}

Polynomial → $2^{\log n}$, $n \log n^2$

Logarithmic → $\log n$, $\log(\log(n^n))$

~~As we~~

$\log(n) \le 1 \cdot (\log(\log n^n))$ for $n \ge n_0$

Thus, $\log(n) = O(\log(\log n^n))$

$\log(\log(n^n)) \le \underline{2} \log(n)$ for $n \ge n_0'$

$c > 0$

$\log(\log(n^n)) = O(\log(n))$

Now since logarithmic grows slower than polynomial and polynomial grows slower than exponential:-

$O(\log(n)) \subset O(2^{\log n}) \subset O(n \log n^2)$

&

$O(n \log n^2) \subset 2^{3n} \subset O(n^{n \log n})$

$O(\log n) = O(\log(\log(n^n)) \subset O(2^{\log n}) \subset O(n \log n^2) \subset O(2^{3n}) \subset O(n^{n \log n}) \subset O(n^{n^2})$

classmate

Date

Page

3] (a) True

$f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$

(1) - $f_1(n) \leq c_1 \cdot (g_1(n))$

(2) - $f_2(n) \leq c_2 \cdot g_2(n)$

Multiplying these two

$$f_1(n) \cdot f_2(n) \leq c_1 c_2 \, g_1(n) \cdot g_2(n)$$

$$\leq C_3 \, g_1(n) \cdot g_2(n)$$

$(C_3 = c_1 \cdot c_2)$

$\therefore f_1(n) \cdot f_2(n) = O(g_1(n) \cdot g_2(n))$

(b) ~~$f_1(n)$~~ True

Adding the above equations (1) & (2)

$$f_1(n) + f_2(n) \leq \underbrace{(c_1 + c_2)}_{C_3} \max(g_1(n), g_2(n))$$

$$f_1(n) + f_2(n) \leq C_3 \max(g_1(n), g_2(n))$$

$$f_1(n) + f_2(n) = O(\max(g_1(n), g_2(n)))$$

Date ______
Page ______

(c) $f_1(n)^2 = O(g_1(n)^2)$ is true

Squaring equation (1) we get

$$f_1(n)^2 \leq c_1^2 \, g_1(n)^2$$

Thus, $f_1(n)^2 = O(g_1(n)^2)$

(d) Taking log on both sides of equation (1)

$$\log_2 f_1(n) \leq \log_2 (c_1 \cdot g_1(n))$$

$$\leq \underbrace{\log_2 c_1 + \log_2 g_1(n)}_{\downarrow}$$

$$\leq c_2 \log_2 g_1(n)$$

$$c_2 = 1 + \log_{g_1(n)} c_1$$

which is true when $\log_2 g_1(n) > 0$

$$\log_2 f_1(n) \leq c_2 \log_2 g_1(n)$$

$\log_2 f_1(n) = O(\log_2 g_1(n))$ is True when $\log_2(g_1(n)) > 0$

Scanned with CamScanner

classmate

Date

Page

4] ~~Consider~~ Using DFS and keeping a track of visited nodes.

Algo :-

Graph G (V,E), n nodes and m edges.

Array [1, ... n] keep a track of whether a node is visited or not.

'p' stores information about previous node when current node is taken into consideration.

For each $v \in V$ that's not visited
Mark v as visited and p = -1
Call a recursive function $\forall v \in adj(v)$ until all nodes are visited.
if $v \neq p$
if v was visited previously and thus cycle is present so return True.

Else set variable p = v and mark v as visited, then call the recursive function for adjacent nodes to v.
Else ignore v and consider other adjacent nodes to v.
Endif

End for
End for
Return no cycle was deducted thus False.

5] Chapter 3, Exercise 6

⇒ Proof by Contradiction: ~~If~~ lets' consider that G contains an edge e' ~~=~~ (x,y) which is not present in T. Since, T is a DFS tree, one of x ~~and~~ or y is an ancestor of the other. T is also a BFS tree that means x and y should differ at most 1 layer. Now if one of x or y are ~~can~~ ancestors of ~~each~~ the other then, they have to be on the 1 layer difference. this then ~~it~~ has to be a part of the BFS tree. which contradicts our assumption. There won't be any edge in G that doesn't belong to T.