Assignment-1

(a) Double the input size :-

Q-2

(i) $f(n) = n^2$

(a) double the input size :-

$f(n) = (2n)^2 = 4n^2$

$\therefore$ Run time of the algorithm get 4 times slower.

(ii) $f(n) = n^3$

$f(n) = (2n)^3 = 8n^3$

$\therefore$ Run time of the algorithm get 8 times slower.

(iii) $f(n) = 100n^2$

$f(n) = 100(2n)^2 = 400n^2 = 4(100n^2)$

$\therefore$ Run time of the algorithm get 4 times slower.

(iv) $f(n) = n\log n$

$f(n) = 2n\log(2n) = 2n(\log_2 2 + \log_2 n)$

$= 2n(1 + \log_2 n)$

$= 2n + 2n\log n$

$\therefore$ Run time of the algorithm ~~increases~~ slower by 2 time and factor $2n+2$.

(v) $f(n) = 2^n$

$f(n) = 2^{(2n)} = (2^n)^2$

$\therefore$ Run time of algorithm is ~~in~~ slower by becomes squared of itself.

(b) Increase ~~by~~ the input size by 1:

(i) $f(n) = n^2$

$f(n+1) = (n+1)^2 = n^2 + 2n + 1$

$\therefore$ Run time ~~increase~~ slower by $2n + 1$.

(ii) $f(n) = n^3$

$f(n+1) = (n+1)^3 = n^3 + 3n^2 + 3n + 1$

$\therefore$ Run time of algorithm is slower by $3n^2 + 3n + 1$.

(iii) $f(n) = 100n^2$

$f(n+1) = (100(n+1)^2) = 100n^2 + 200n + 100.$

$\therefore$ Run time of algorithm is slower by $200n + 100$.

(iv) $f(n) = n \log n$

$f(n+1) = (n+1)\log(n+1)$

$= \cancel{(n+1)\log}\ n\log(n+1) + \log(n+1)$

Now,

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

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

$\therefore$ Run time of algo. is slower by $\log(n+1) + n[\log(n+1) - \log n]$

(v) $f(n) = 2^n$

$f(n+1) = 2^{(n+1)} = 2^n \times 2$

$\therefore$ Run time of Algorithm is slower by 2 time

Q-4

(i) $\log_2 f(n)$ is $O(\log_2 g(n))$

If,

$f(n)$ is $O(g(n))$ then $f(n) \leq c * g(n)$.

Let,

$f(n) = 2$ & $g(n) = 1$.

$\log f(n) = \log 2 = 1$

$\log (g(n)) = \log 1 = 0$

$\therefore$ $\log_2 f(n)$ is not $O(\log_2 g(n))$

Hence,

False

(ii) $2^{f(n)}$ is $O(2^{(g(n))})$.

$f(n) \leq c * g(n)$.

So, let,

$f(n) = 2n$ & $g(n) = n$

$\therefore$ $2^{f(n)} = 2^{(2n)} = 4n$ & $2^{g(n)} = 2^n$

$\because (2^2)^n = 4^n$ is not in $O(2^n)$.

$\therefore$ $2^{f(g)}$ is not $O(2^{g(n)})$.

Hence, False

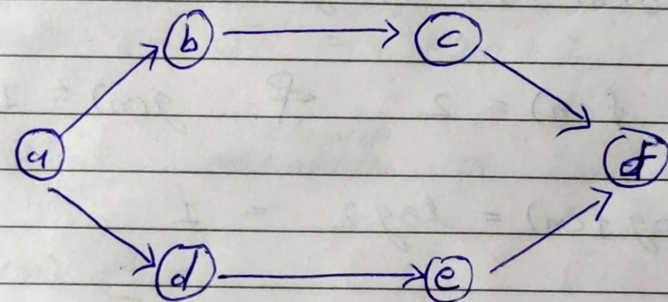
(iii) $f(n)^2$ is $O(g(n)^2)$

Here, $f(n) \leq c \cdot g(n)$

$\therefore f(n)^2 \leq (c * g(n))^2 = c^2 (g(n))^2$
$\therefore f(n)^2 \leq c * g(n^2)$
Hence,

True

Q-5



Therfore,
- a is first.
- f must come last.
- b precedes c.
- d precedes e.

Hence,
total topological orders are:
i - {a, b, c, d, e, f}
ii - {a, d, e, b, c, f}
iii - {a, b, d, c, e, f}
iv - {a, d, b, e, c, f}
v - {a, b, d, e, c, f}
vi - {a, d, b, c, e, f}

six topological order

2-6 - A cycle in a graph refer to the begin from a vertex & ends at that vertex only.

- The begining & the ending vertex are same.

- DFS-Algorithm is used to detect the cycle in the graph.

→ D_F_S (G)

for each vertex s in graph G
    do color [s] ← WHITE
    parent [s] ← NULL
time ← 0

for each vertex s in graph G
    do if color [s] ← WHITE
        then D_F_S_Visit [s]
END D_F_S

→ D_F_S_Visit [S]
    color [s] ← GREY
    time ← time + 1
    d [s] ← time

for each vertex v adjacent to s in graph G.

do if color [v] != WHITE
    then return (cycle in G)
else
    parent [v] ← s

D_F_S_Visit (v)

color [s] ← BLACK
time ← time + 1
f [s] ← time

END D_F_S_Visit

→ Thus, The complexity is O(m+n) in which n represents the nodes of the graph and m reprensents the edges of the graph.

Q-7

Suppose,
input graph G is an undirected tree T.

Graph G has edges & vertices
$G = (V, E)$
Assuming,
DFS tree T rooted at u & BFS tree T including all nodes of graph G.
Let,
Graph G contains an edge
$E = \{x, y\}$
Hence,

Contradiction proved.

Q8

The claim is true.

→ Let G be a graph.
Number of nodes in G is s,
and s is an even number.

Assume, G is not connected.
If graph G is not connected then there must exist atleast 1 node separated from other nodes.

From statement, every node should have edges to at least $n/2$ other nodes.

$\left(\frac{n}{2}+1\right)$ nodes form one component, disconnected from other components.

$$\text{No. of nodes other} = n - \left(\frac{n}{2}+1\right)$$

$$= n - \frac{n}{2} - 1$$

$$= \frac{n}{2} - 1$$

∴ These $\frac{n}{2}-1$ cannot form another different component among themselves,
Because every node in G should have edges to $n/2$ other nodes.

As there are $\left(\frac{n}{2}-1\right)$ nodes in B, every node in this component should have atleast one edge to previous component.

∴ Therefore both the components are connected.