QUESTION # 01:

Computational Complexity:

a) $T(n) = 27T(n/3) + \theta(n^3 \log_2 n)$

b) Solving using Master Theorem for time complexity:

$T(n) = 27T(n/3) + \theta(n^3 \log n)$

$a=27, \quad b=3, \quad k=3, \quad p=1$

$\log_3 27 = 3$

$\log_b(a) = k$

Master Theorem case 2:

$\therefore \quad \theta(n) = n^3 \log_2^2(n)$

c) Proof Upper Bound using Substitution Method:

$T(n) = 27T(n/3) + \theta(n^3 \lg n)$

Let ~~$O(n) \leq cn^3 \lg^2(n)$~~ $T(n) \leq cn^3 \lg^2 n$

$T(n/3) \leq c(n/3)^3 \lg^2(n/3) \Rightarrow cn^3/27 \lg^2(n/3)$

$\leq$ ~~$c27 \lg$~~ $27 \frac{cn^3}{27} \lg^2(n/3) +$ ~~$d(n^3 \lg$~~ $dn^3 \lg n$

$\leq cn^3 \lg^2(n/3) + dn^3 \lg n$

$\leq cn^3 \lg^2 n - cn^3 \lg^2(3) + dn^3 \lg n$

$\leq n^3 \lg n (c \lg n + d) - \underbrace{cn^3 \lg^2(3)}_{\hookrightarrow}$

$c, d > 0$

$\therefore \quad O(n) = cn^3 \lg^2(n)$

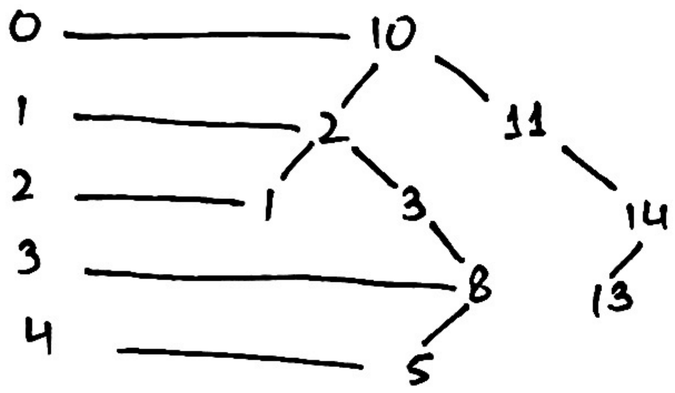
Guess function is correct

QUESTION # 02:

BINARY TREE:

A = [10, 2, 3, 8, 11, 5, 1, 14, 13]

a)



Height of B-Tree : 4

Successor of 3: 5

Predecessor of 13: 11

Inorder Tree Walk:

1 → 2 → 3 → 5 → 8 → 11 → 13 → 14

b) Red-Black Tree:

Structure of Data Node (Data, color) R=Red, B= Black

(10, B) —insert 2→ 10, B / 2, R —insert 3→ 10, B / 2, R / 3, R i.e. Red-Red conflict when 3 is inserted

fixed by recolouring the root as black after performing L-R rotation

→ (B) 3, B / 2, R / 10, R.

→ 3, B / 2, R / 10, R / 8, R →

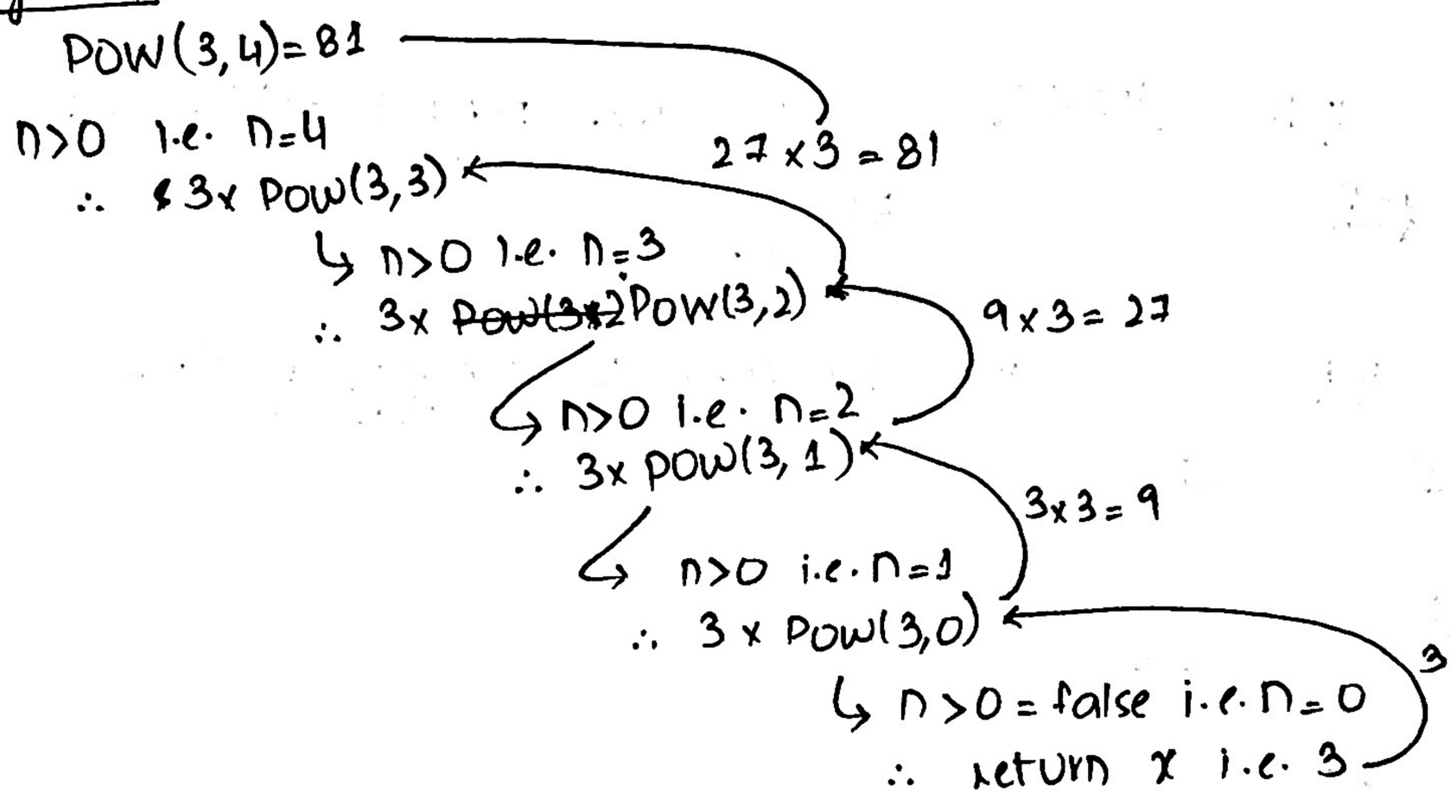
QUESTION # 03:

DIVIDE & CONQUER:

a) **PSUEDO CODE:**

```
POW(X,n):
    if (n>0) then
        return  X * POW(X,n-1)
    else
        return  1 ~~Pow(X,n-1)~~ X
```

b) **Dry-RUN**



c) **Running Time Equation:**

$$T(n) = T(n-1) + O(1)$$

d) **Time complexity:**

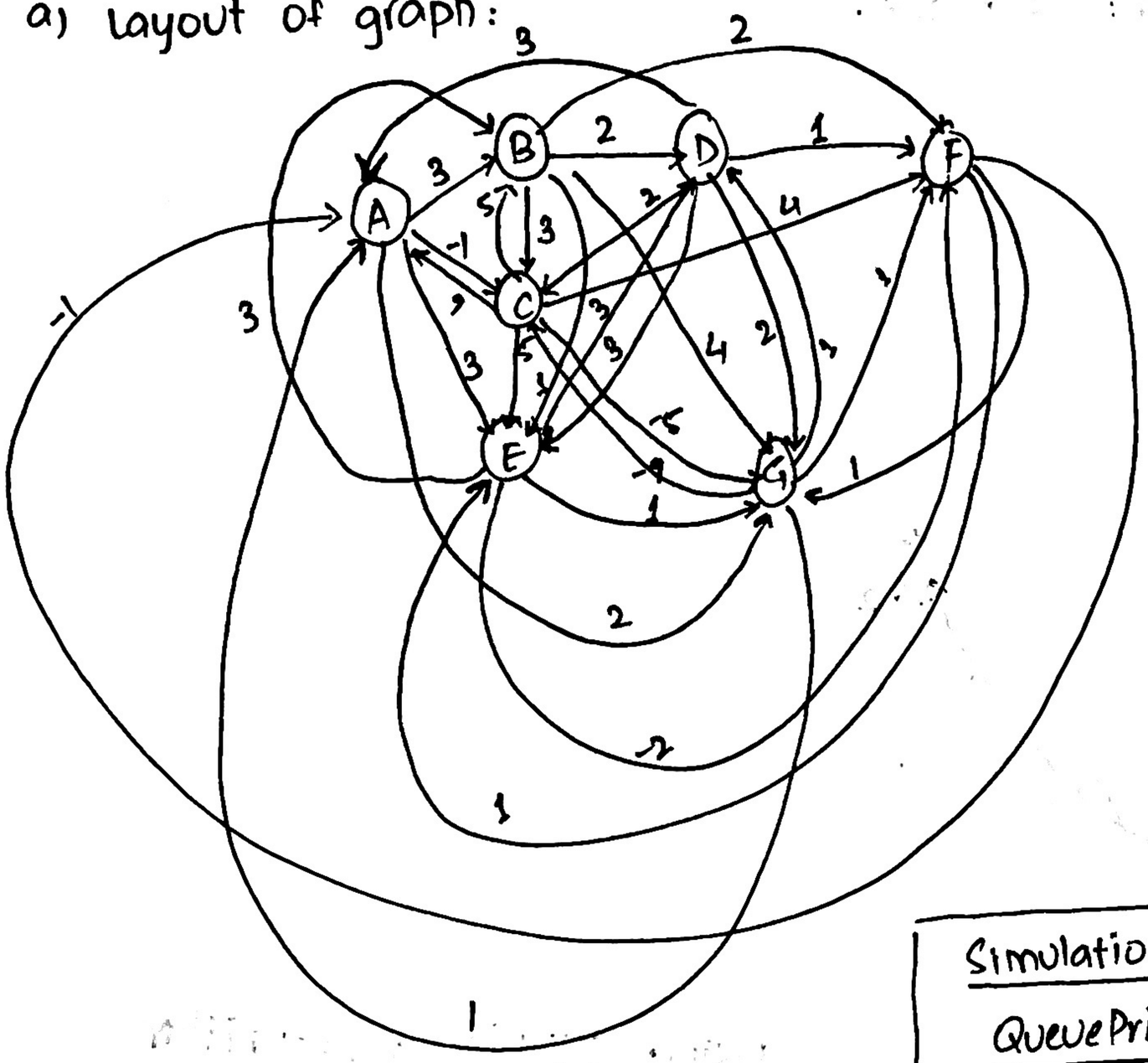
Since the sub-problem is reduced by 1 in each recursive call therefore the time complexity of function is $\theta(n)$.

Muhammad Owais Imran
(4)

QUESTION # 04:

GRAPH:

a) Layout of graph:



The type of graph is directed graph

b) BFS using A as source vertex

A→A : 0
A→B: 6
A→C: 1
A→D: 3
A→E: 5
A→F: 4
A→G: 2

Visit Order:

~~A, G, C~~
A→C→G→D→F→E→B

Simulation of BFS: Visited: A, C, G, D, F, E, B

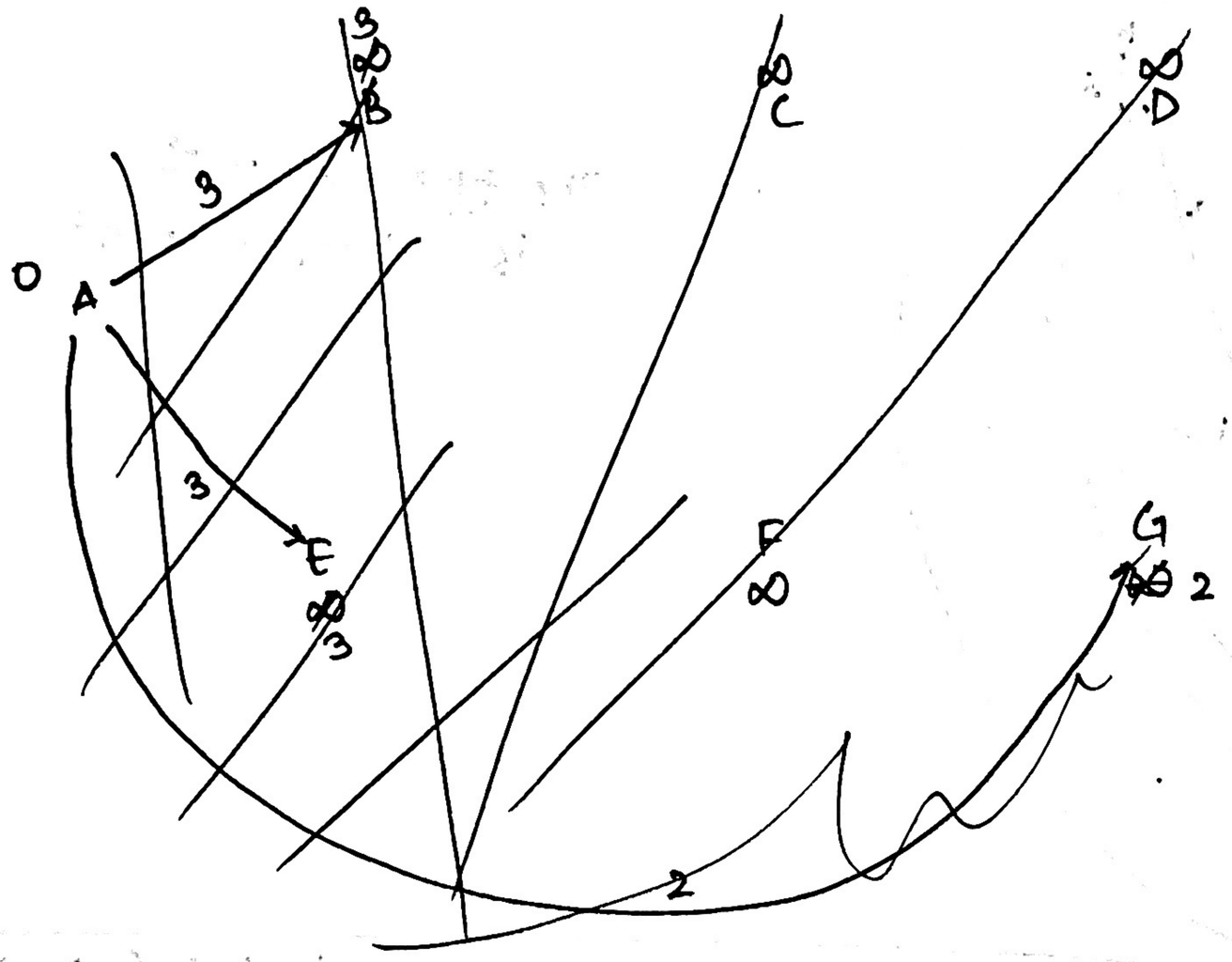
QueuePriority: (After insertion of vertex with increasing order of weight)

A → ~~A~~ C, B, E, G → G, B, E, F → D, F, E, B →
F, E, B → E, B → B → Φ

~~B, E, G~~
~~{(A,B), (A,E), (A,G)}~~ ∪ ~~{(C,A)}~~, {(C,B), (C,E), (C,F), (C,G)}

c) Path of (A,G)

~~Using Bellman-Ford Algorithm: B(A,G)=?~~



Using Dijkistra Algorithm:

Selected vertex	A	B	C	D	E	F	G
A	(0)	3	(-1)	∞	3	∞	2
AC	0	3	(-1)	∞	3	3	(-6)
ACG	⋮	⋮	⋮	⋮	⋮	⋮	⋮

Using Dijkistra Algorithm the path can be ~~AGE~~ ACG with a total cost of -6.

Although we can also use ~~B~~ Bellman Ford for each algorithm & can have different ~~result~~ weight * on vertex G as the weight would be relaxed everytime due to negative weight cycle.

Although Dijkistra can & cannot generate optimal shortest path.

d)

Bellman ford Algorithm would be a good choice as it can cater negative weighted edges as compared to Dijkistra which can or cannot extract the optimal shortest path. As in our case, Bellman ford is well suited due to prescence of negative weighted edges. Since our graph has negative weight cycles therefore both the algorith can or cannot predict optimal shortest path.

~~Bellman / Dijkstra~~

~~(A,B) (A,C) (A,E)~~

	A	B	C	D	E	F	G
B	∞	0,B	3,B	2,B	1,B	2,B	4,B
BE	∞	0	3,B	2,B	1,B	-1,E	2,E
BEF	-2,F	0	3,B	2, B	1,B	-1,E	0,F
BEFA	-2,F	0	-3,A	~~0,B~~ 2,B	1,B	-1,E	0,f
BEFAC	-2,F	0	-3,A	~~2,B~~	1,B	-1,E	-8,C
BEFACG	-2,F	0	-3,A	-1,G	1,B	-1,E	-8,C
BEFACG	-2,F	0	-3,A	-1,G	1,B	-1,E	-8,C

Shortest Path:

~~B→E : 1~~

B→A: BEFA

B→C: BEFAC

B→D: BEFACGD

B→E: BE

B→F: BEF

B→G: BEFACG