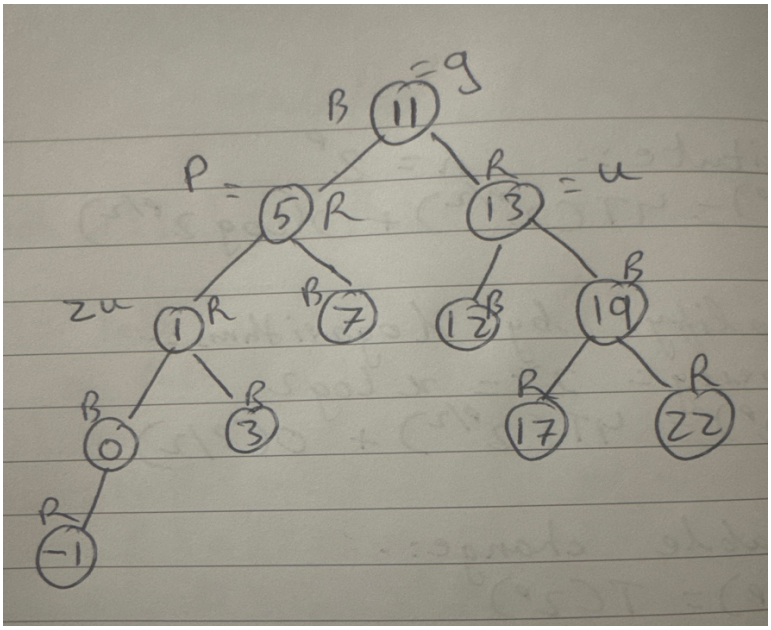
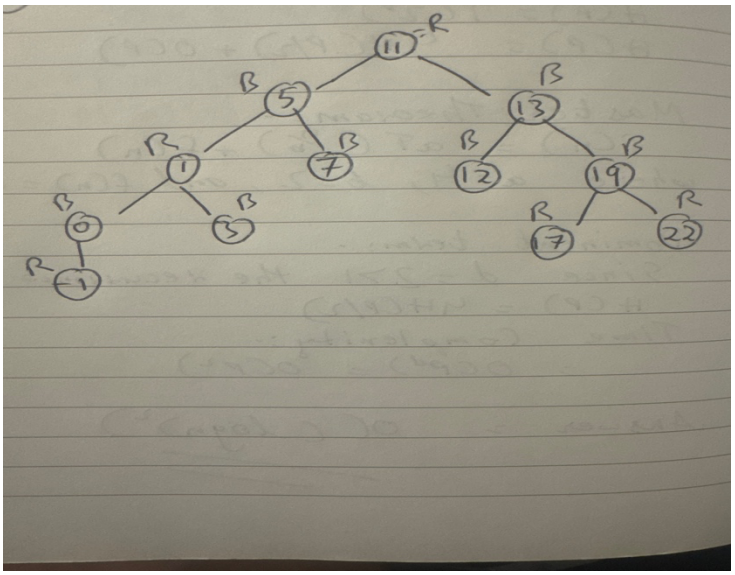
Question 1 - 20 points
When inserting a node into a red-black tree, we may encounter three possible cases where at least one red-black property is violated. Given two red-black tree with a new node just inserted (marked as 'z'), (1) Identify which case each of the following tree encountered at node z. (2) Show step by step how to fix the red-black trees.

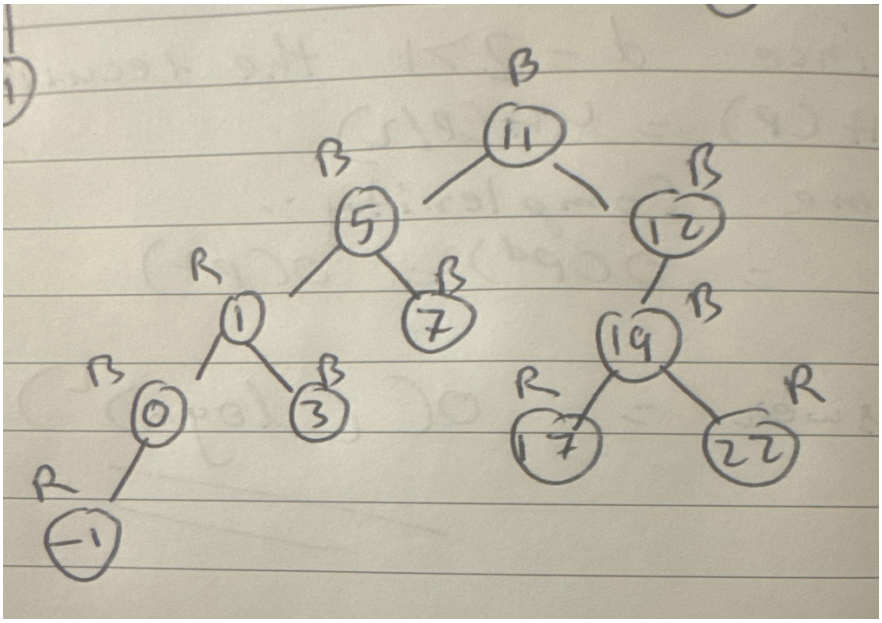
Answer1:
u = 3, g = 1, Z = -1, and p = 0.
U is red, and G's left child is P.
Case 1: The parent and grandparent levels are now black instead of red.
When p is red or z is not the root:



Grandparents are red, whereas the uncle and parents are black.
Turn your grandmother into the new Z.
Make the root black.

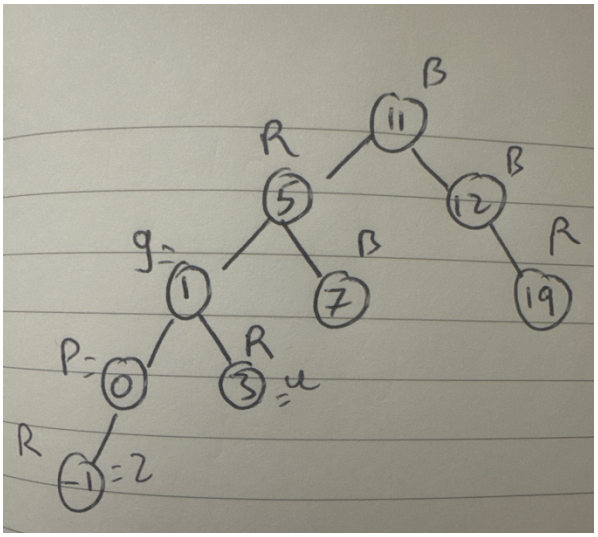


z = 1, p = 5, u = 13, and g = 11 in this case.
It qualifies as case 1.
Make the root black last.

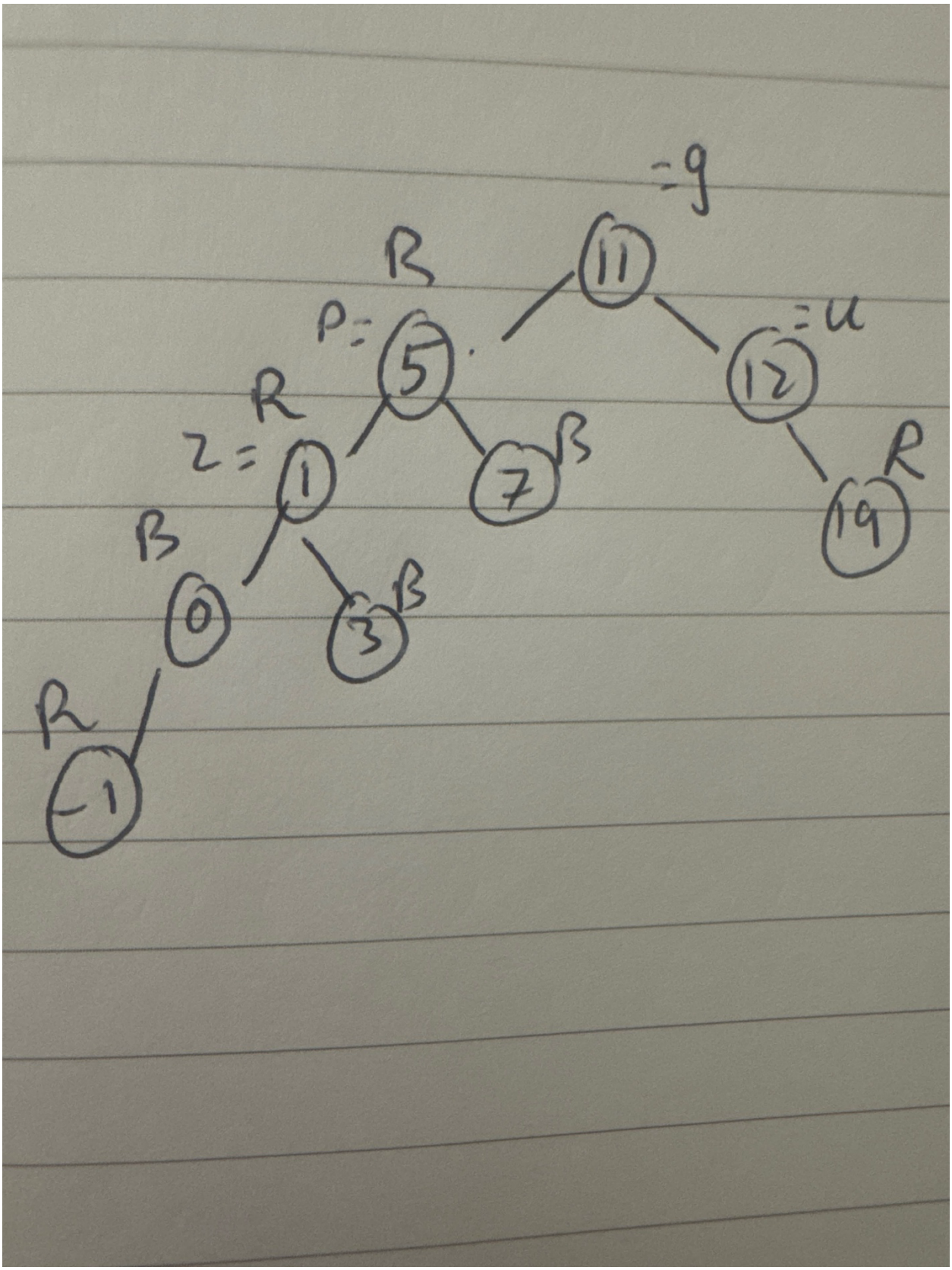


The provided tree has been fixed and now meets the requirements for a red-black tree.
In this case, g = 1, u = 3, p = 0, and z = -1.
U is red, and G's left child is P.

Case 1: The parent and grandparent levels are now black instead of red.
When p is red or z is not the root, the grandparents are red and the parents and uncle are black.
Turn your grandmother into the new Z.
Make the root black.

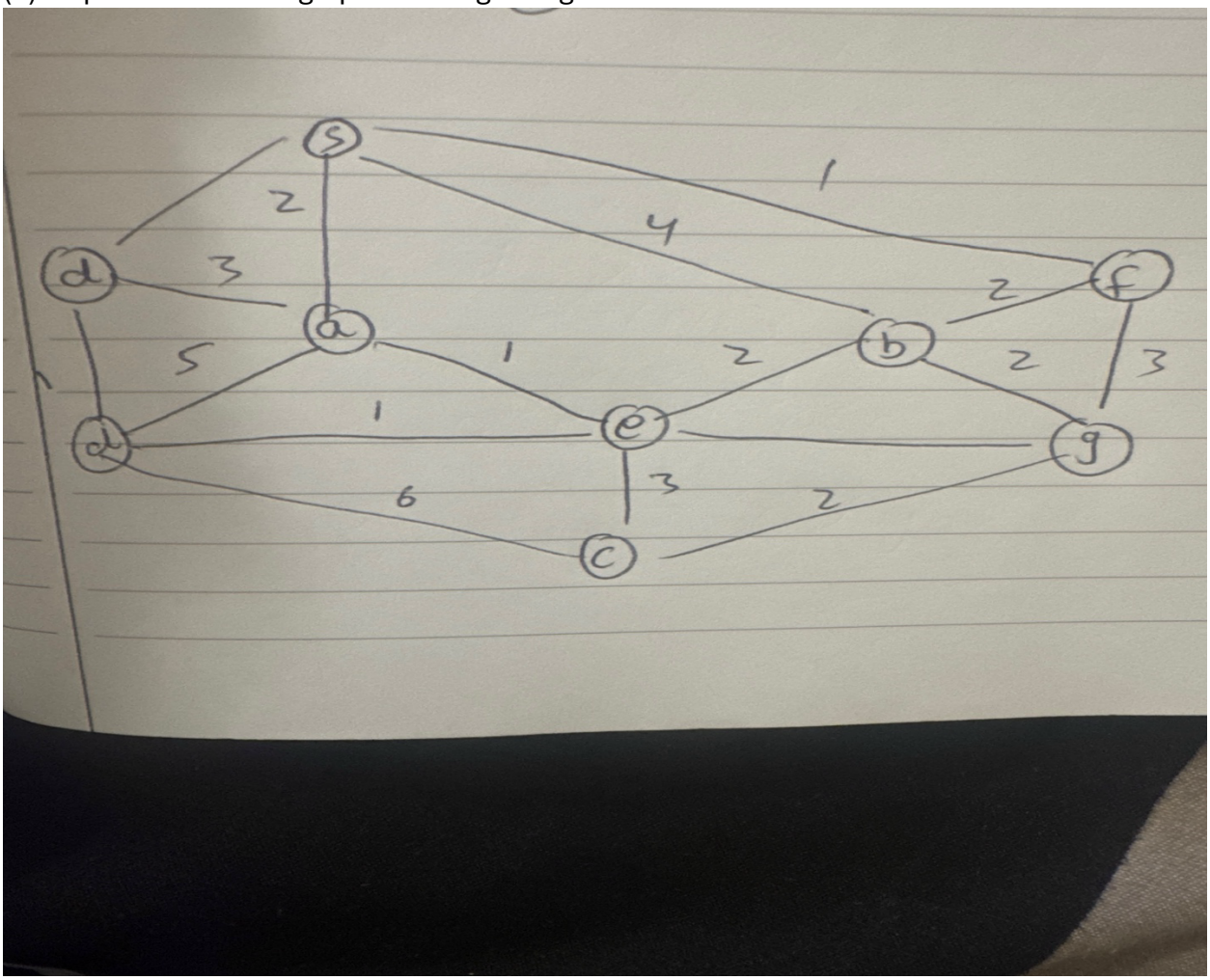


p = 5, u = 12, z = 1, and g = 11 in this case.
P is G's left child, and U is Black. Z is P's surviving child as well. This is covered by case 3.
In instance 3, we switch the colors of p and g and rotate (p, g) to the left.
The resulting tree is as follows:


=g
R
11
P=
5
=u
12
R
Z=
1
7
B
R
19
B
0
3
B
R
-1

Now that it has been fixed, the tree meets all requirements to be classified as a red-black tree.

Question 2 - 50 points Given the following adjacency matrix for an un- directed graph.
(1) - 5 points Draw the graph with edge weights.



(2) - 10 points Give the traversal order of BFS and DFS starting from node 's'. When enqueue or push unvisited neighbors into queue or stack, follow the alphabetical order.

Answer:

From the node, S

BFS:

Queue:

Back --→ [ ] --→ Front

Enqueue Source Vertex S.
Back --→ [S] --→ Front

The queue is not empty; it enqueues its unvisited neighbors and dequeues S.
D, A, B, and F are the neighbors of S that have not been visited.
Sort these vertices in alphabetical order.

Back --→ [A B D F] --→ Front

Order of traversal: S
Enqueue A's unvisited neighbors after dequeuing A.

Back --→ [I E F D B] --→Front
Order of traversal: S A
Enqueue B's unvisited neighbors after dequeuing B.

Back --→ [G I E F D] --→ Front
Order of traversal: S A B
Enqueue D's unvisited neighbors after dequeuing D.

Back --→ [G I E F] --→ Front
Order of traversal: S A B D
Enqueue F's unvisited neighbors after dequeuing F.

Back --→ [G I E] --→ Front
Order of traversal: S A B D F
Enqueue E's unvisited neighbors after dequeuing E.

Back --→ [C G I] --→ Front
Order of traversal: S A B D F E
Enqueue I's unvisited neighbors and dequeue I.

Back --→ [C G] --→ Front
Order of traversal: S A B D F E I
Enqueue G's unvisited neighbors after dequeuing G.

Back --→ [C] --→ Front
Order of transit: S A B D F E I G
Enqueue C's unvisited neighbors after dequeuing C.

Back--→ [] --→ Front
The sequence of traversal is S A B D F E I G C

There is nothing in the queue.
Order of BFS traversal: S A B D F E I G C

DFS:
Stack:
[] --→ top

Add Source S to the stack's queue.
[S] --→ top

In alphabetical order, Pop S and push its unvisited neighbors.
[A B D F] --→ top

Order of traversal: S
push F's unvisited neighbors and pop F.

[A B D G] --→ top

Order of traversal: S F
push G's unvisited neighbors after pop G.

[A B D C E] --→ top

Order of traversal: S F G
push E's unvisited neighbors after pop E.

[A B D C I] --→ top

Order of traversal: S F G E
push I's unvisited neighbors and pop I.

[A B D C] --→ top

Order of traversal: S F G E I
push C's unvisited neighbors after pop C.

[A B D] --→ top

Order of traversal: S F G E I C
push B's unvisited neighbors and pop B.

[A B] --→ top

Order of traversal: S F G E I C D
push D's unvisited neighbors after pop D.

[A] --→ top

Order of traversal: S F G E I C D B
Dequeue A and add its unvisited neighbors to the queue.

[]--→ top

The sequence of traversal is S F G E I C D B A
Now there is nothing on the stack.
DFS The sequence of traversal is S F G E I C D B A

(3) - 15 points Use Prim's algorithm to find a minimum spanning tree (MST) of the graph. You need to show step by step results and calculate the weight of the final MST.

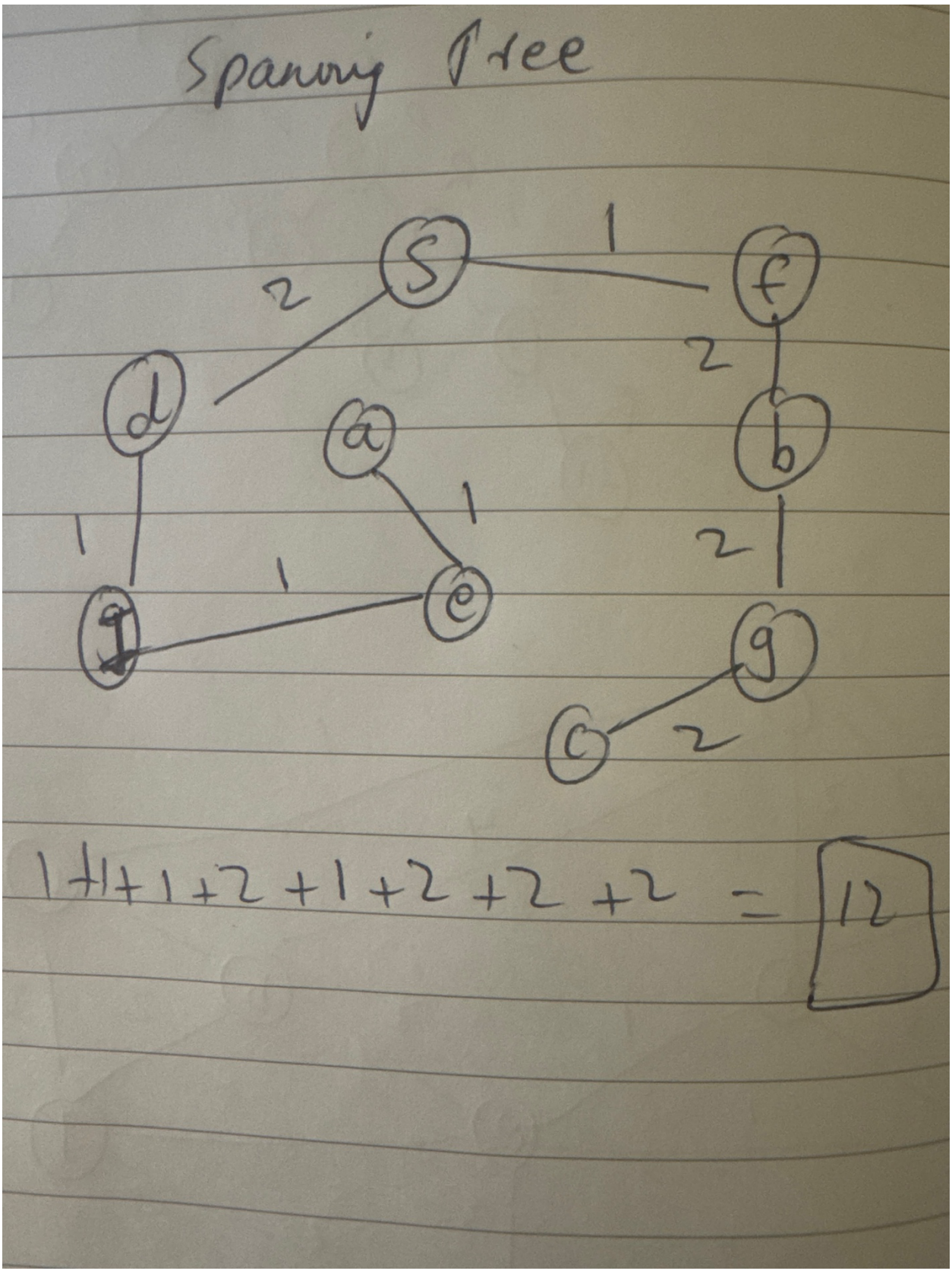
Minimum Spanning Tree:
Considering start vertex to be e

Including e in set A and remaining nodes in set Q

Nodes	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**I**	**S**
Distance	1	2	3	Inf	/	Inf	6	1	Inf
	/	2	3	3	/	Inf	6	1	2
	/	2	3	1	/	Inf	6	/	2
	/	2	3	/	/	Inf	6	/	2
	/	2	3	/	/	1	6	/	/
	/	2	3	/	/	/	3	/	/
	/	/	3	/	/	/	2	/	/
	/	/	2	/	/	/	/	/	/
	/	/	/	/	/	/	/	/	/

The distances are entered in the table each time a node that is closest to set A is added. Next, we add the next closest set to set A and remove from set Q until set Q is empty. The table's minimal spanning tree is as follows:

B -> G -> C -> E -> I -> D -> S -> F



(4) - 20 points Use Dijkstra's algorithm to find the shortest path from s to rest of the nodes in the graph. You need to show step by step results and the actual shorted paths from s to each node.

Nodes	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**I**	**S**
Distance	Inf	Inf	Inf	Inf	Inf	Inf	Inf	Inf	0
Predecessor	Nil	Nil	Nil	Nil	Nil	Nil	Nil	Nil	Nil

The shortest paths between nodes and the rest of all nodes:

Nodes	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**I**	**S**
Distance	2	4	Inf	2	Inf	1	Inf	Inf	0
Predecessor	S	S	Nil	S	Nil	S	Nil	Nil	Nil

Add node F to the group.

Nodes	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**I**	**S**
Distance	2	3	Inf	2	Inf	1	4	Inf	0
Predecessor	S	F	Nil	S	Nil	S	F	Nil	Nil

Add node A to the group.

Nodes	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**I**	**S**
Distance	2	3	Inf	2	3	1	4	7	0
Predecessor	S	F	Nil	S	A	S	F	A	Nil

Add node B to the group.

Nodes	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**I**	**S**
Distance	2	3	Inf	2	3	1	4	3	0
Predecessor	S	F	Nil	S	A	S	F	D	Nil

Add node I to the group.

Nodes	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**I**	**S**
Distance	2	3	9	2	3	1	4	3	0
Predecessor	S	F	I	S	A	S	F	D	Nil

Add node D to the group.

Nodes	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**I**	**S**
Distance	2	3	Inf	2	3	1	4	3	0
Predecessor	S	F	Nil	S	A	S	F	D	Nil

Add node E to the group.

Nodes	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**I**	**S**
Distance	2	3	6	2	3	1	4	3	0
Predecessor	S	F	E	S	A	S	F	D	Nil

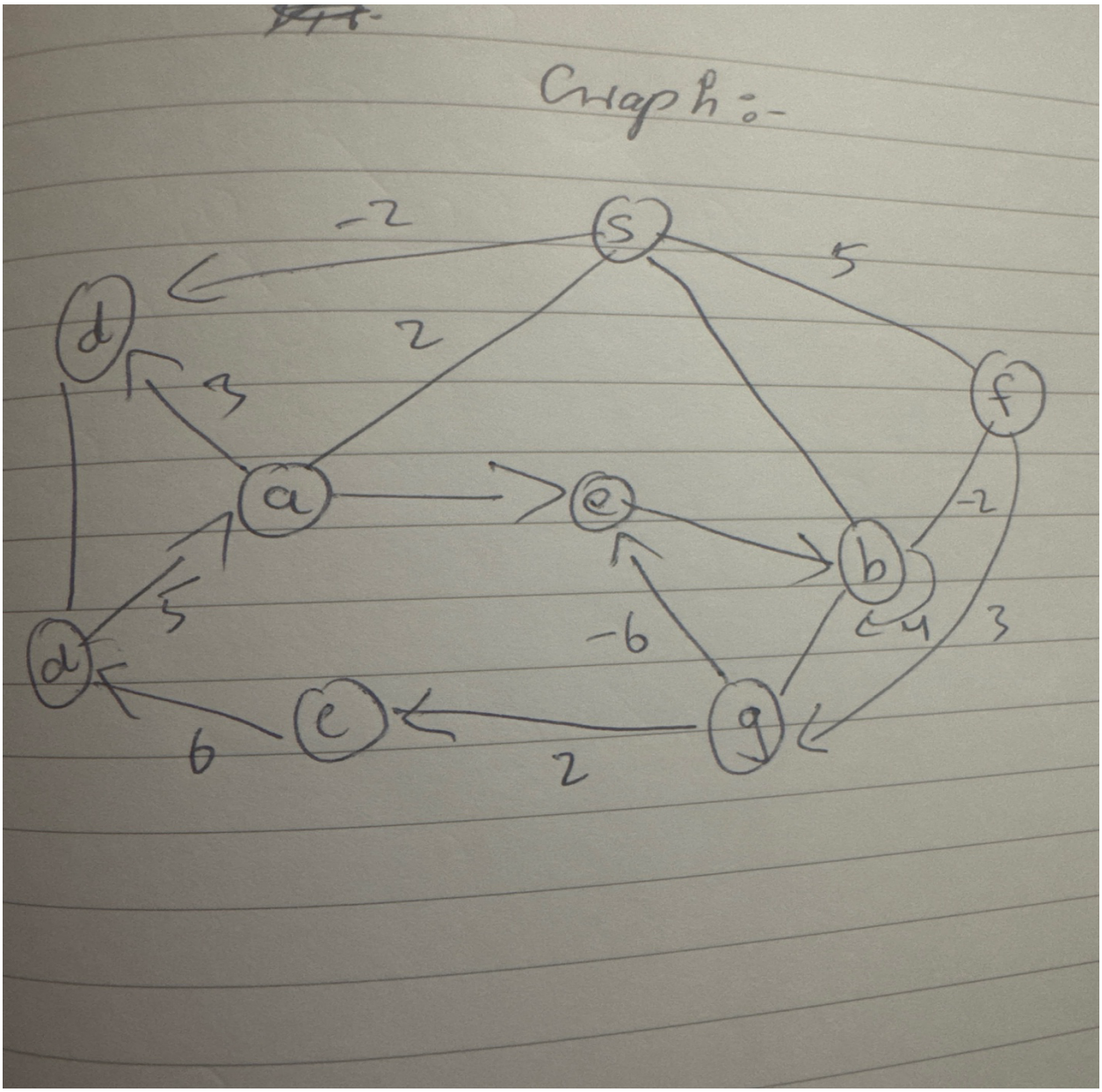
Add node G to the group.

Nodes	**A**	**B**	**C**	**D**	**E**	**F**	**G**	**I**	**S**
Distance	2	3	6	2	3	1	4	3	0
Predecessor	S	F	E	S	A	S	F	D	Nil

The shortest path from S to:
S -> A (2)
S -> F -> B (3)
S -> A -> E -> C (6)
S -> D (2)
S -> A -> E (3)
S -> F (1)
S -> F -> G (4)
A -> D -> I (3)

Question 3 - 40 points Given the following adjacency matrix for a directed graph. Assume matrix entry (i, j) is for edge i → j.

(1) - 10 points Draw the graph with edge weights.



(2) - 10 points Give the traversal order of BFS and DFS starting from node 's'. When enqueue or push unvisited neighbors into queue or stack, follow the alphabetical order.

ANSWER:

BFS:

Back -→ [ ] -→Front

Enque S:
Back -→ [S] -→Front

Dequeue S and enqueue
Back -→ [F B] -→Front

Traversal order: S
Dequeu b and enqueu
Back -→ [G F] -→Front

Traversal order: S B
Dequeue F and enqueue
Back -→ [G] -→Front

Traversal Order: S B F
Qequeue G and enqueue
Back -→ [E C] -→Front

Traversal Order: S B F G
Dequeue C and enqueue
Back -→ [I E] -→Front

Traversal Order: S B G F E
Dequeu E and enqueue
Back -→ [I] -→Front

Traversal order: S B F G C E I
Enque a and now queue is empty after node d

So the BFS traversal is SBFGCEIAD

DFS:
Stack: []
Push: []
Starting from S

Push S
[S]

Pop S
[B F]

Pop F
[B G]

Pop G
[B C E]

Pop E
[B C]

Pop C
[B I]

Pop I
[B A]

Pop A
[B D]

Pop D
[B]

Pop B and its empty

So the DFS traversal is SFGECIADB

References:

Red-Black Trees

- GeeksforGeeks: Introduction to Red-Black Trees
- Brilliant: Red-Black Trees

Graph Algorithms (BFS, DFS, Prim's, Dijkstra's)

- GeeksforGeeks: Graph Traversals (BFS and DFS)
- GeeksforGeeks: Prim’s Minimum Spanning Tree Algorithm
- GeeksforGeeks: Dijkstra’s Shortest Path Algorithm