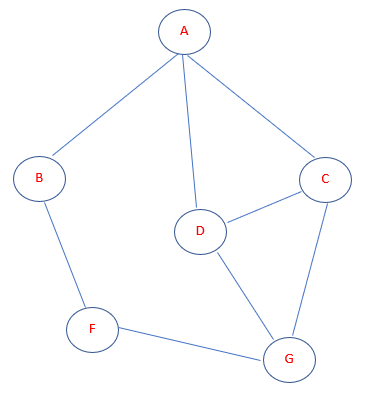
Buenaventura Aristotle C.  
IT12S1

Question 2

Instruction:

* Traverse the graph using the BFS and DFS methods.
* Follow the table in traversing the graph.



**Breadth-first search**

|  |  |  |
| --- | --- | --- |
| Node | Description | Output  -Queue/Stack  -Output |
|  | Initialize the queue | Queue: Output: |
| A | Start from visiting A (Starting node) and mark it as visited. | Queue: Output: A |
| B | We have 3 nodes but alphabetically we choose B, mark it as visited and enqueue it. | Queue: B Output: A |
| C | Next, the unvisited adjacent node from A is C. We mark it as visited and enqueue it | Queue: C, B Output: A |
| D | Next, the unvisited adjacent node from A is D. We mark it as visited and enqueue it. | Queue: D, C, B Output: A |
|  | Now, A is left no unvisited adjacent nodes. So, we dequeue and find B | Queue: D, C Output: A, B |
| F | From B we have F as unvisited adjacent node. We mark it as visited and enqueue it. | Queue: F, D, C Output: A, B |
|  | Now, B is left no unvisited adjacent nodes. So, we dequeue and find C | Queue: F, D Output: A, B, C |
| G | From C we have G as unvisited adjacent node. We mark it as visited and enqueue it. | Queue: G, F, D Output: A, B, C |
|  | At this stage, we are left no unmarked (unvisited) nodes. But as per algorithm we keep on dequeuing in order to get all unvisited nodes. | Queue: G, F, D Output: A, B, C |
|  | Dequeue D from the queue | Queue: G, F Output: A, B, C, D |
|  | Dequeue G from the queue | Queue: G Output: A, B, C, D, F |
|  | Dequeue F from the queue | Queue:  Output: A, B, C, D, F, G |
|  | Completed breadth-first search | Queue:  Output: A, B, C, D, F, G |

**Depth-first search**

|  |  |  |
| --- | --- | --- |
| Node | Description | Output  -Queue/Stack  -Output |
|  | Initialize the stack | Stack:  Output: |
| A | Mark A as visited and put it into the stack. Explore any unvisited adjacent node from A. We have three nodes and we can pick any of them but we shall take the node in an alphabetical order. | Stack: A  Output: A |
| B | Mark B as visited and put it into the stack. Explore any unvisited adjacent node from B. Both A and F are adjacent to B but we are concerned for unvisited nodes only. | Stack: A, B  Output: A, B |
| F | Mark F as visited and put it into the stack. Explore any unvisited adjacent node from F. Both B and G are adjacent to F but we are concerned for unvisited nodes only. | Stack: A, B, F  Output: A, B, F |
| G | Mark G as visited and put it into the stack. Explore any unvisited adjacent node from G. F, D, and C are adjacent to G but we are concerned for unvisited nodes only. | Stack: A, B, F, G  Output: A, B, F, G |
| C | Mark C as visited and put it into the stack. Explore any unvisited adjacent node from C. A, D, and G are adjacent to C but we are concerned for unvisited nodes only. | Stack: A, B, F, G, C  Output: A, B, F, G, C |
| D | Mark D as visited and put it into the stack. Explore any unvisited adjacent node from D. D does not have any unvisited nodes. So, we keep popping the stack until we find a node that has an unvisited node. | Stack: A, B, F, G, C, D  Output: A, B, F, G, C, D |
|  | Pop D from the stack | Stack: A, B, F, G, C,  Output: A, B, F, G, C, D |
|  | Pop C from the stack | Stack: A, B, F, G  Output: A, B, F, G, C, D |
|  | Pop G from the stack | Stack: A, B, F  Output: A, B, F, G, C, D |
|  | Pop F from the stack | Stack: A, B  Output: A, B, F, G, C, D |
|  | Pop B from the stack | Stack: A  Output: A, B, F, G, C, D |
|  | Pop A from the stack | Stack:  Output: A, B, F, G, C, D |
|  | Completed depth-first search | Stack:  Output: A, B, F, G, C, D |