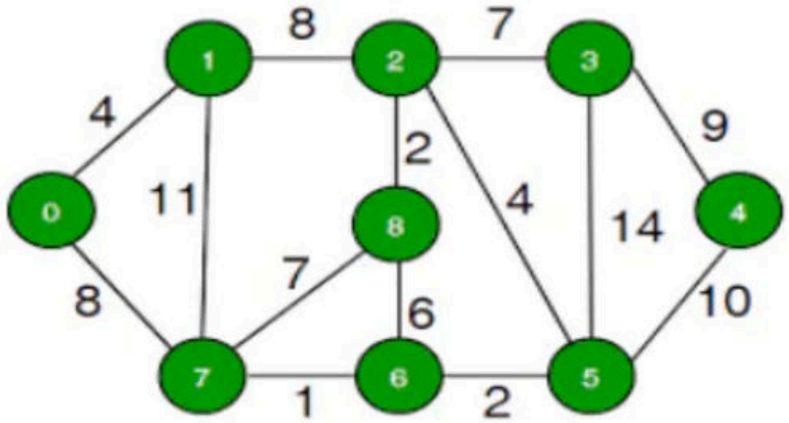
Functions in C Programming

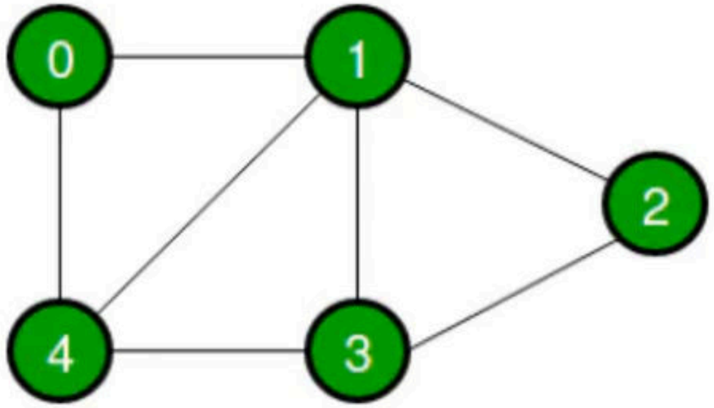
Complete Course on Data Structures for GATE

Sanchit Jain • Lesson 27 • July 18, 2022

Graph

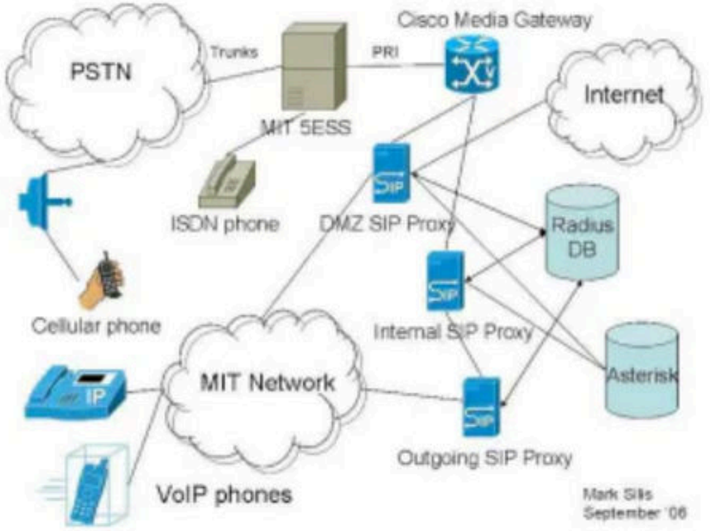
- Graph is a data structure that consists of following two components:
  - A finite set of vertices also called as nodes.
  - A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of a directed graph(di-graph).
  - The pair of the form (u, v) indicates that there is an edge from vertex u to vertex v. The edges may contain weight/value/cost.





- Graphs are used to represent many real-life applications: Graphs are used to represent networks. The networks may include paths in a city or telephone network or circuit network.

- Graphs are also used in social networks like LinkedIn, Facebook. For example, in Facebook, each person is represented with a vertex (or node). Each node is a structure and contains information like person id, name, gender and locale.



Representation of Graph in Memory

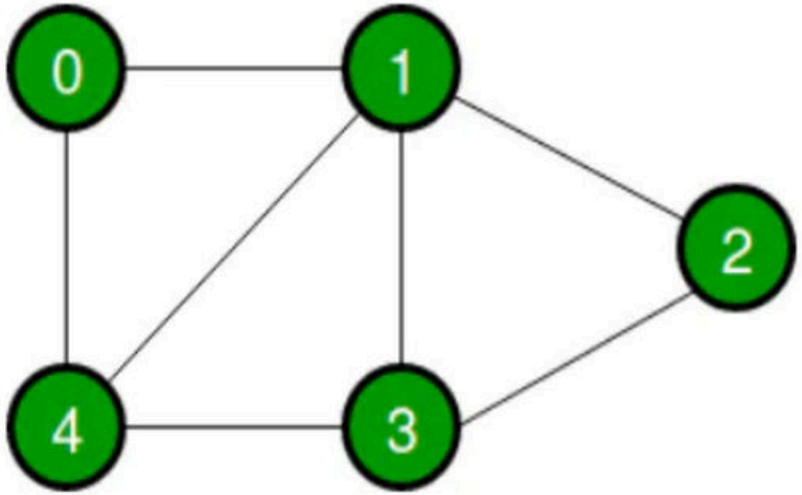
- Following two are the most commonly used representations of a graph.
  - Adjacency Matrix
  - Adjacency List

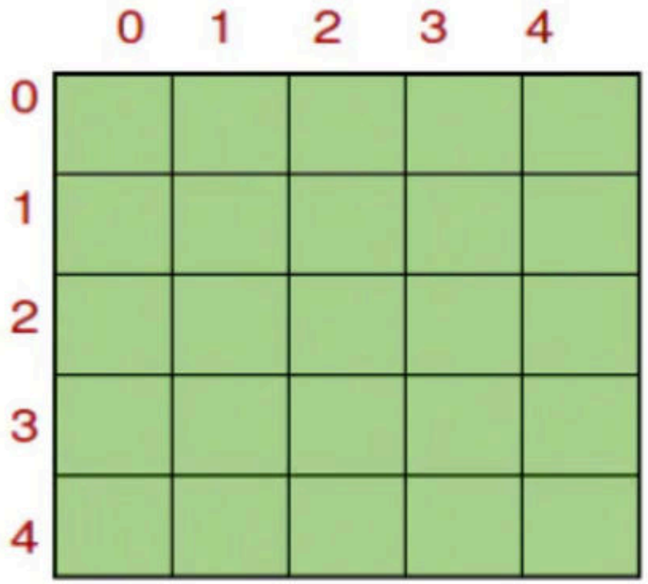
- There are other representations also like, Incidence Matrix and Incidence List. The choice of the graph representation is situation specific. It totally depends on the type of operations to be performed and ease of use.

- **Adjacency Matrix:** Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j.

- Adjacency matrix for undirected graph is always symmetric.

- Adjacency Matrix is also used to represent weighted graphs. If adj[i][j] = w, then there is an edge from vertex i to vertex j with weight w.

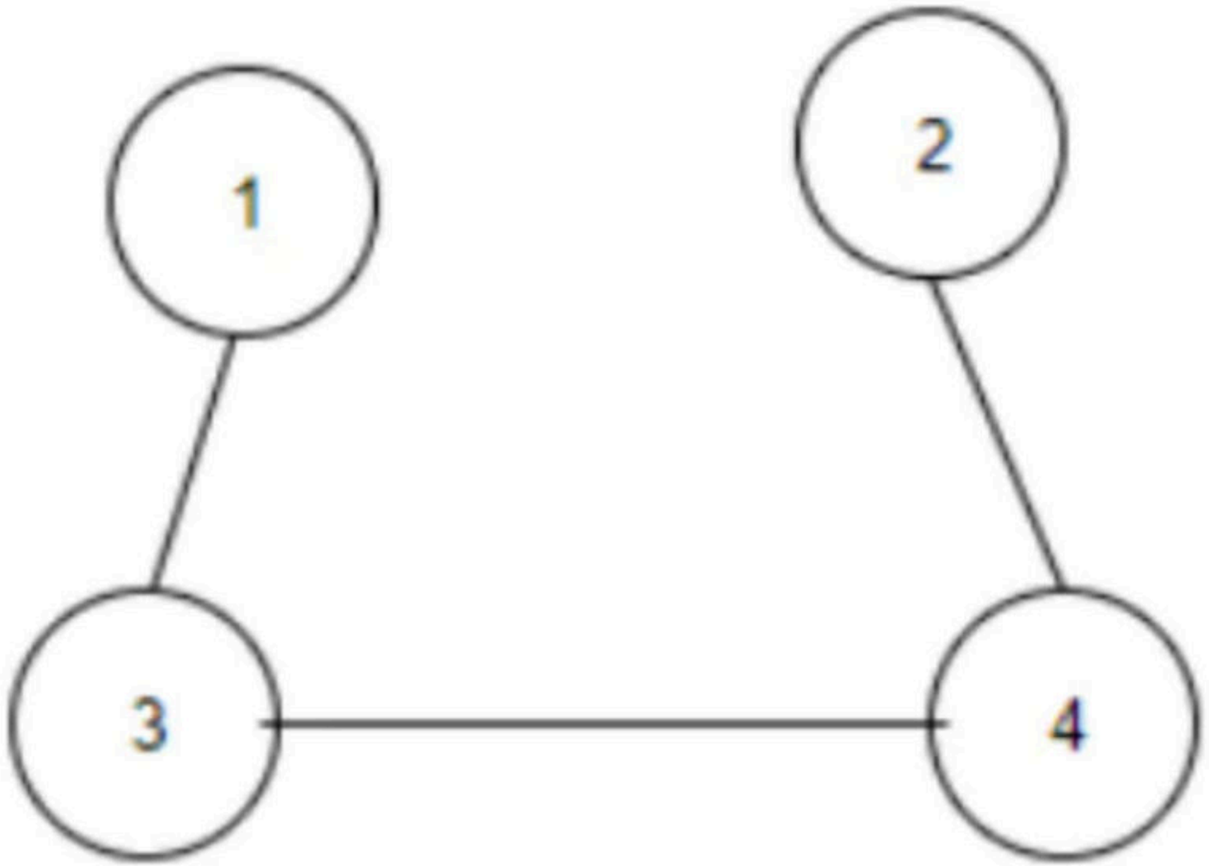




- **Pros:** Representation is easier to implement and follow. Removing an edge takes O(1) time. Queries like whether there is an edge from vertex ‘u’ to vertex ‘v’ are efficient and can be done O(1).

- **Cons:** Consumes more space $O(V^2)$. Even if the graph is sparse(contains less number of edges), it consumes the same space. Adding a vertex is $O(V^2)$ time.

Q. What would be the number of zeros in the adjacency matrix of the given graph? **[Asked in AMCAT 2016]**



a. 10

b. 6

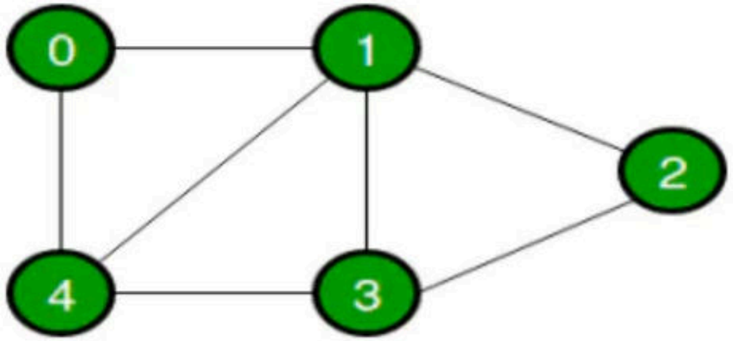
c. 16

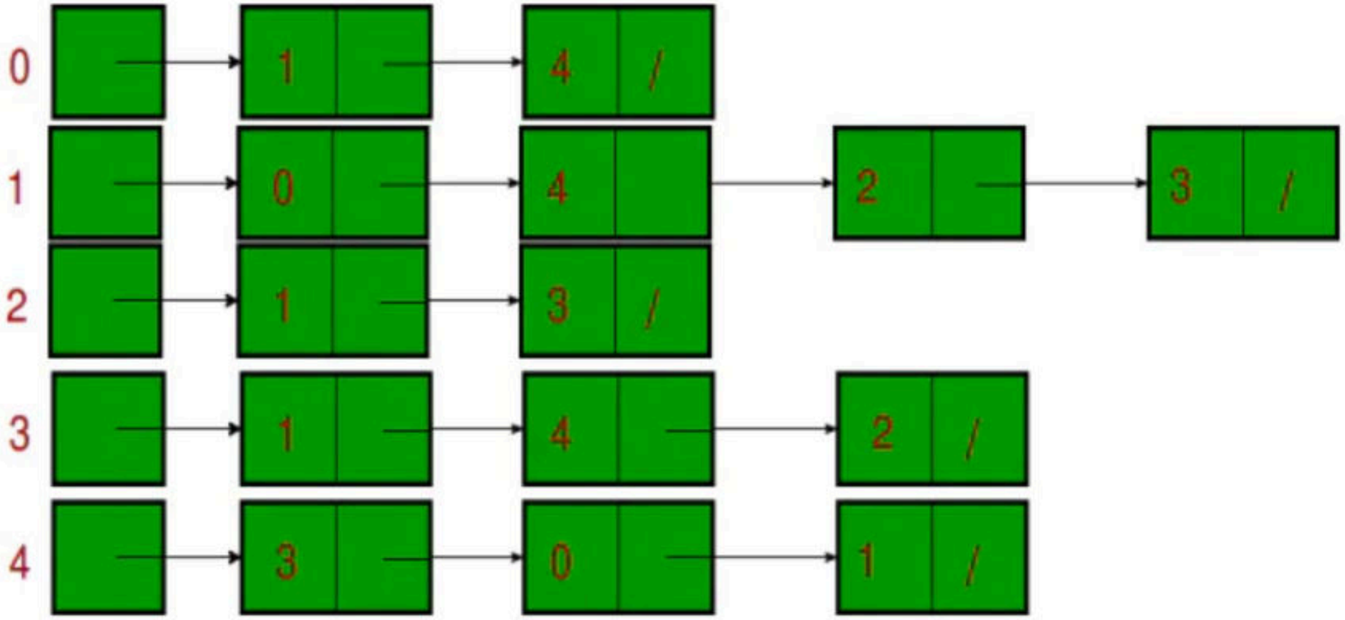
d. 0

Answer: b

Explanation: Total number of values in the matrix is 4*4=16, out of which 6 entries are non zero.

- **Adjacency List:** An array of lists is used. Size of the array is equal to the number of vertices. Let the array be array[]. An entry array[i] represents the list of vertices adjacent to the ***i***th vertex. This representation can also be used to represent a weighted graph. The weights of edges can be represented as lists of pairs.





Break

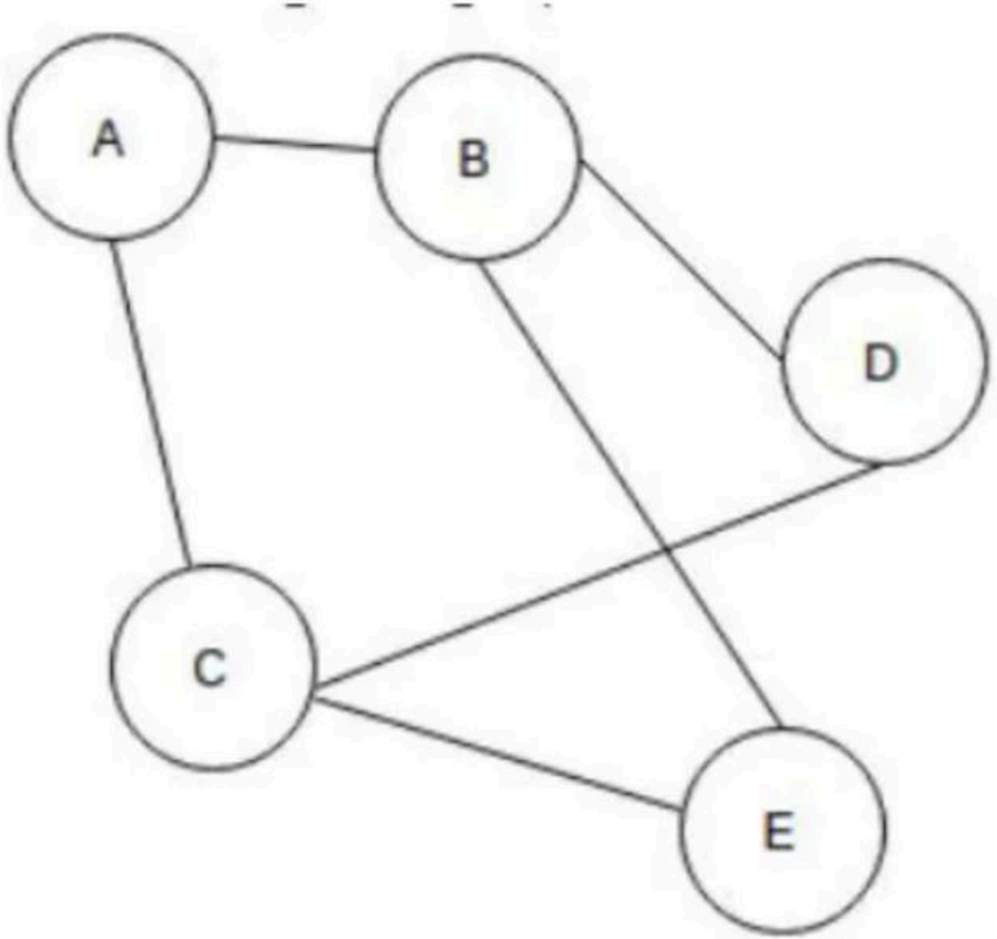
Q. Which of the following statements for a simple graph is correct? **[Asked in Capgemini]**

a) Every path is a trail
b) Every trail is a path
c) Every trail is a path as well as every path is a trail
d) Path and trail have no relation

Answer: a

Explanation: In a walk if the vertices are distinct it is called a path, whereas if the edges are distinct it is called a trail.

Q. For the given graph(G), which of the following statements is true? **[Asked in TCS NQT 2021]**



a) G is a complete graph
b) G is not a connected graph
c) The vertex connectivity of the graph is 2
d) The edge connectivity of the graph is 1

Answer: c

Explanation: After removing vertices B and C, the graph becomes disconnected.

Q. What is the number of edges present in a complete graph having n vertices? **[Asked in Hexaware 2017]**

a) (n*(n+1))/2
b) (n*(n-1))/2
c) n
d) Information given is insufficient

Answer: b
Explanation: Number of ways in which every vertex can be connected to each other is nC2.

If a simple graph G, contains n vertices and m edges, the number of edges in the Graph G'(Complement of G) is _____ **[Asked in Goldman Sachs 2019]**

a) (n*n-n-2*m)/2
b) (n*n+n+2*m)/2
c) (n*n-n-2*m)/2
d) (n*n-n+2*m)/2

Answer: a

Explanation: The union of G and G’ would be a complete graph so, the number of edges in G’= number of edges in the complete form of G(nC2)-edges in G(m).

Q. The degree sequence of a simple graph is the sequence of the degrees of the nodes in the graph in decreasing order. Which of the following sequences can not be the degree sequence of any graph?

I. 7, 6, 5, 4, 4, 3, 2, 1

II. 6, 6, 6, 6, 3, 3, 2, 2

III. 7, 6, 6, 4, 4, 3, 2, 2

IV. 8, 7, 7, 6, 4, 2, 1, 1

[Asked in TCS NQT 2018]

(A) I and II

(B) III and IV

(C) IV only

(D) II and IV

Answer (D)

In sequence IV, we have a vertex with degree 8 which is not possible in a simple graph (no self loops and no multiple edges) with total vertex count as 8. Maximum possible degree in such a graph is 7.

In sequence II, four vertices are connected to 6 other vertices, but remaining 4 vertices have degrees as 3, 3, 2 and 2 which are not possible in a simple graph (no self loops and no multiple edges).

Q. How many undirected graphs (not necessarily connected) can be constructed out of a given set V= {V 1, V 2,…V n} of n vertices ? **[Asked in E-litmus 2021]**

a. n(n-l)/2
b. 2^n
c. n!
d. 2^(n(n-1)/2

Answer : D

Explanation:

In an un-directed graph, there can be maximum n(n-1)/2 edges. We can choose to have (or not have) any of the n(n-1)/2 edges. So, total number of un-directed graphs with n vertices is 2^(n(n-1)/2).

Which of the following statements is/are TRUE for an undirected graph? P: Number of odd degree vertices is even Q: Sum of degrees of all vertices is even **[Asked in L&T Infotech (LTI) 2021]**

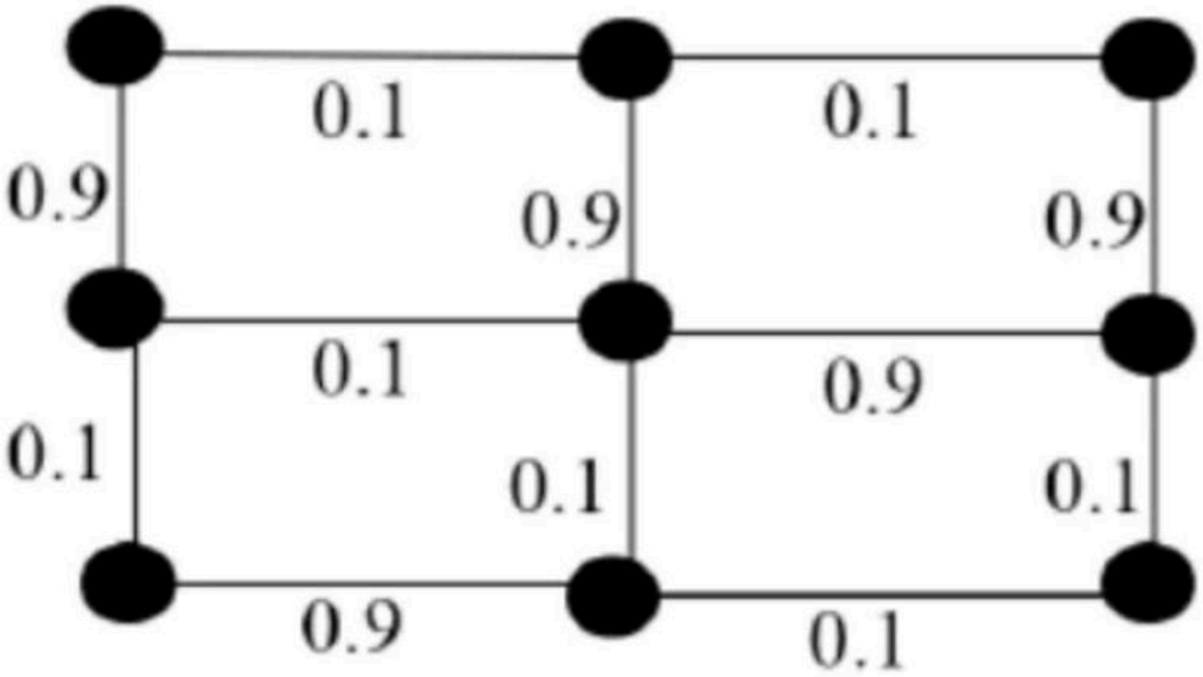
a. P Only
b. Q Only
c. Both P and Q
d. Neither P nor Q

Answer : C

Explanation:

P is true for undirected graph as adding an edge always increases degree of two vertices by 1. Q is true: If we consider sum of degrees and subtract all even degrees, we get an even number because every edge increases the sum of degrees by 2. So total number of odd degree vertices must be even.

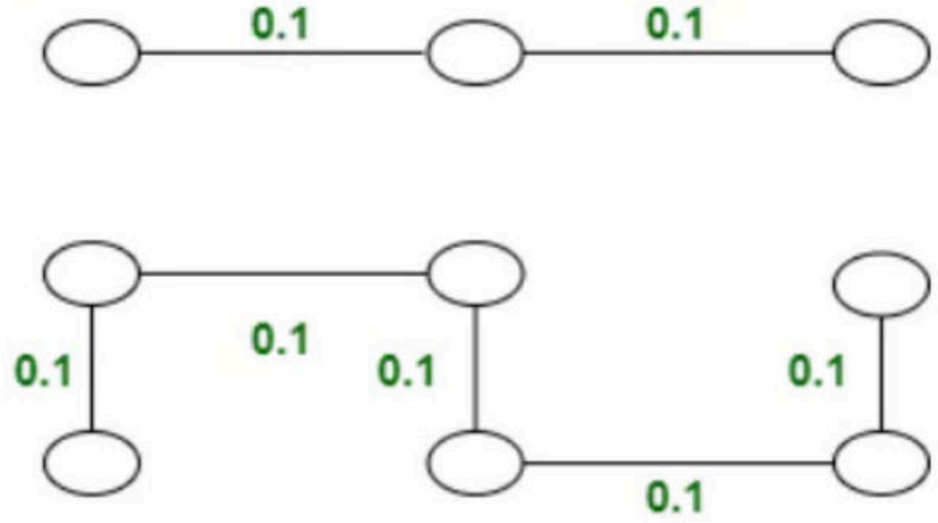
Q. Consider the following un-directed graph with edge weights as shown: The number of minimum-weight spanning trees of the graph is _____. **[Asked in Cognizant 2021]**



a. 3
b. 4
c. 5
d. 6

Ans : A

According to Kruskal’s Minimum Spanning Tree using :



Now, there are 3 edges between these components to connect them. According to Kruskal's algorithm, we will include minimum weights edges first if there is no cycle resultant. But, we need only one edge to form spanning tree, and we have 3 options for one edge. Hence, number of spanning trees are 3.

Q. Given an undirected graph G with V vertices and E edges, the sum of the degrees of all vertices is **[Asked in L&T Infotech (LTI) 2019]**

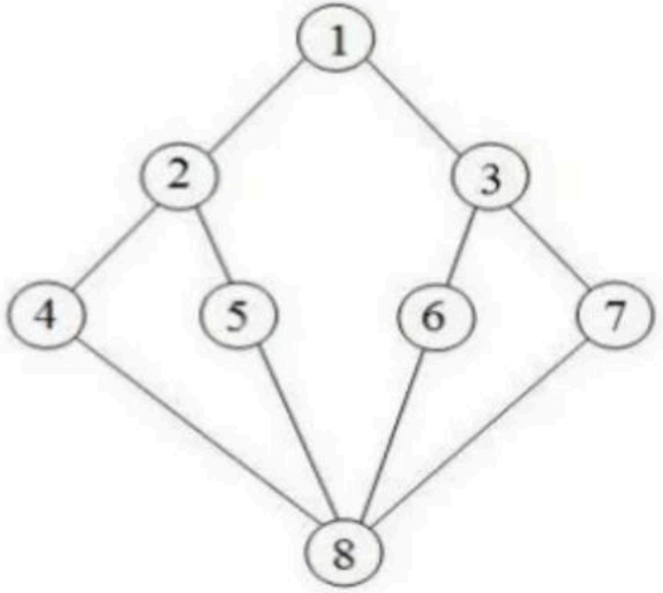
a. E
b. 2E
c. V
d. 2V

Explanation:

Since the given graph is undirected, every edge contributes as 2 to sum of degrees. So the sum of degrees is 2E.

Graph Traversal

- Traversal means visiting all the nodes of a graph.
- Depth First Traversal (or Search) for a graph is similar to Depth First Traversal of a tree.
- The only catch here is, unlike trees, graphs may contain cycles, so we may come to the same node again. To avoid processing a node more than once, we use a Boolean visited array.



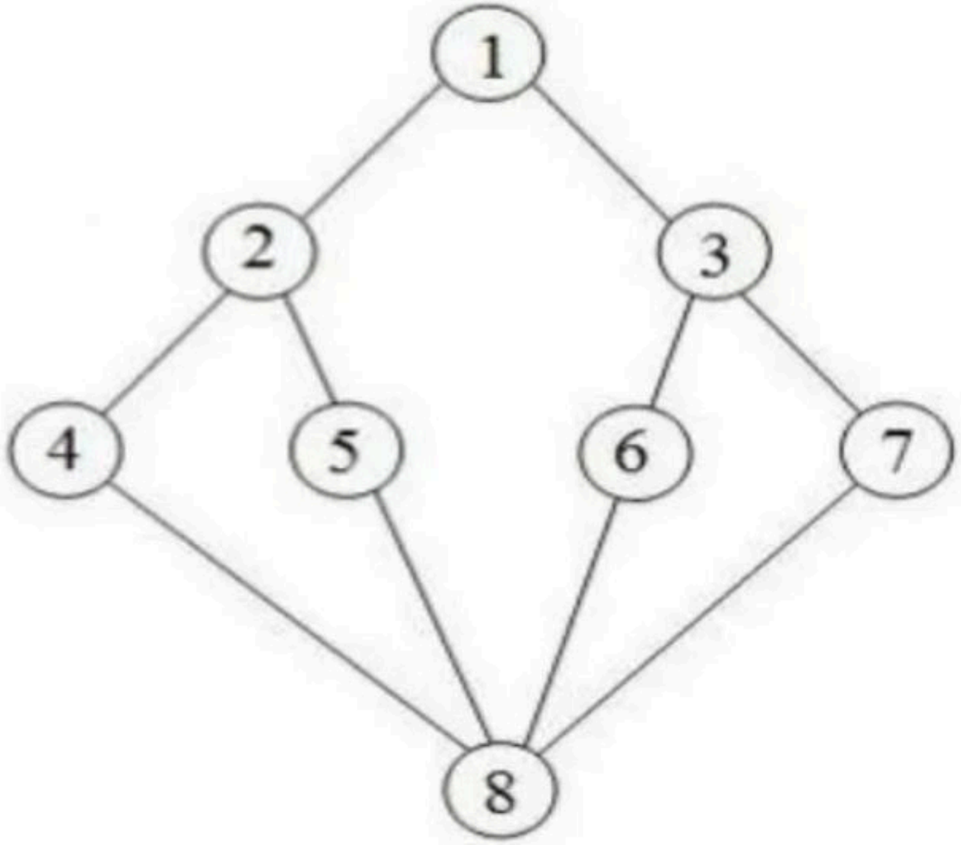
Q Which of the following are valid and invalid DFS traversal sequence

a) 1, 3, 7, 8, 5, 2, 4, 6

b) 1, 2, 5, 8, 6, 3, 7, 4

c) 1, 3, 6, 7, 8, 5, 2, 4

d) 1, 2, 4, 5, 8, 6, 7, 3

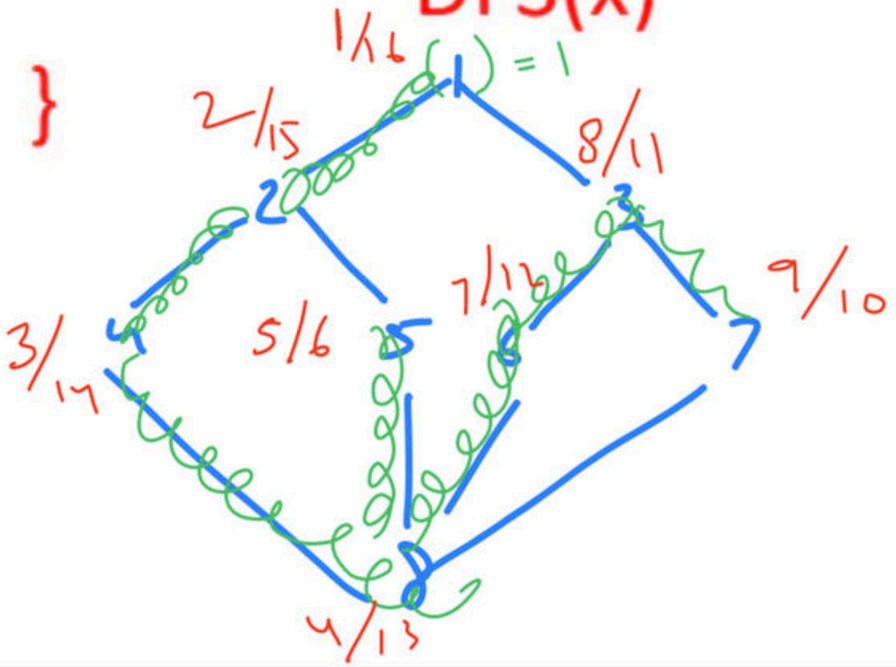


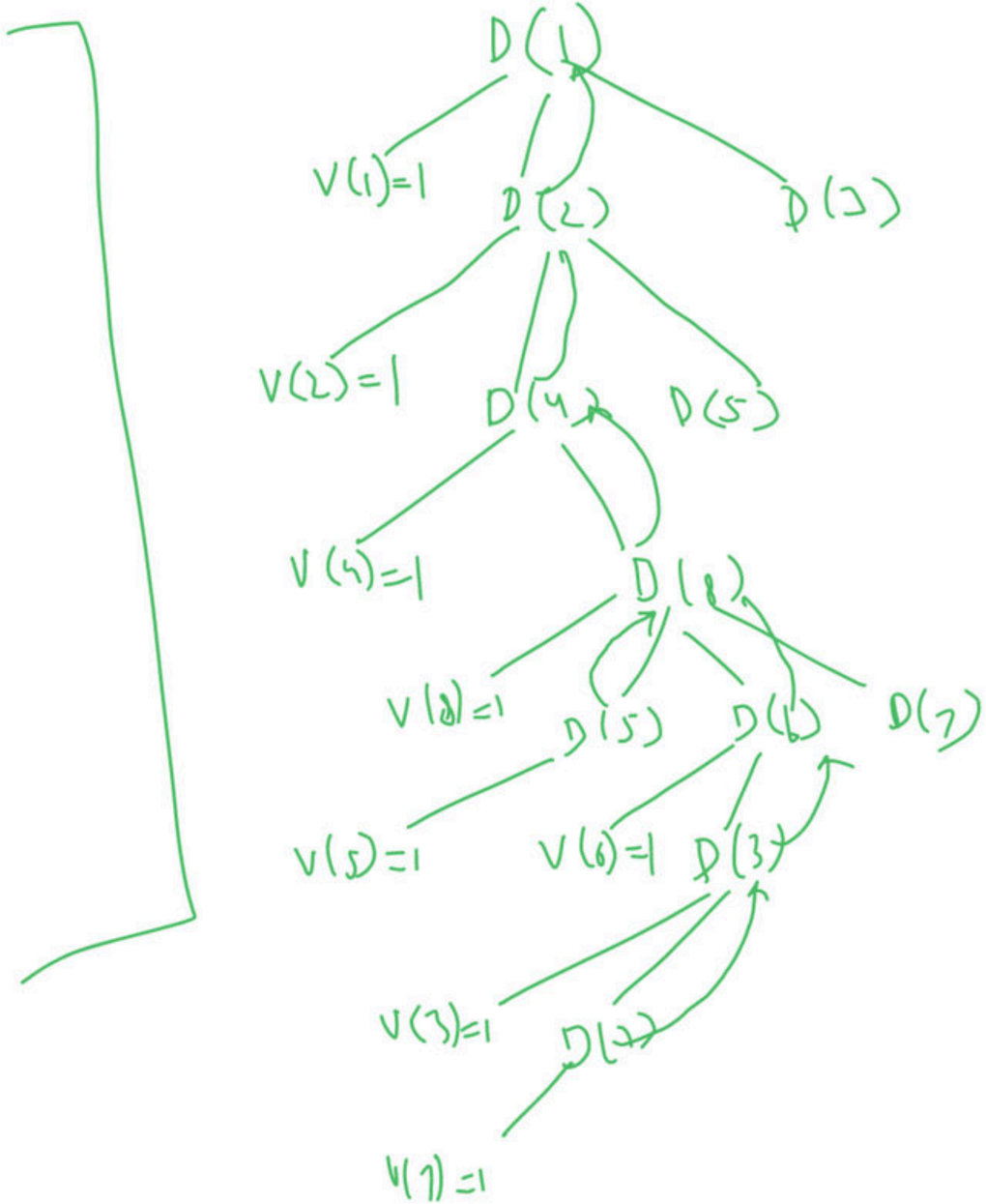
- A standard DFS implementation puts each vertex of the graph into one of two categories:
  - Visited
  - Not Visited

- The purpose of the algorithm is to mark each vertex as visited while avoiding cycles.

- The DFS algorithm works as follows:
  - Start by putting any one of the graph's vertices on top of a stack.
  - Take the top item of the stack and add it to the visited list.
  - Create a list of that vertex's adjacent nodes. Add the ones which aren't in the visited list to the top of stack.
  - Keep repeating steps 2 and 3 until the stack is empty.

```
DFS(v)
{
    visited(v) = 1
    For all x adjacent to v
    {
        if (x is not visited)
            DFS(x)
    }
}
```





```
DFS-iterative (G, s)
{
        let S be stack
        Push( s )
        while ( S is not empty)
        {
                v = pop(S)
                if v is not marked as visited
                {
                        mark v as visited
                        for all neighbors w of v in Graph G:
                        {
                                if w is not marked as visited:
                                        push( w )
                        }
                }
        }
}
```

```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)

DFS-VISIT(G, u)
 1  time = time + 1          // white vertex u has just been discovered
 2  u.d = time
 3  u.color = GRAY
 4  for each v ∈ G.Adj[u]    // explore edge (u, v)
 5      if v.color == WHITE
 6          v.π = u
 7          DFS-VISIT(G, v)
 8  u.color = BLACK          // blacken u; it is finished
 9  time = time + 1
10  u.f = time
```

What is the running time of DFS? The loops on lines 1–3 and lines 5–7 of DFS take time $\Theta(V)$, exclusive of the time to execute the calls to DFS-VISIT. As we did for breadth-first search, we use aggregate analysis. The procedure DFS-VISIT is called exactly once for each vertex $v \in V$, since the vertex u on which DFS-VISIT is invoked must be white and the first thing DFS-VISIT does is paint vertex u gray. During an execution of DFS-VISIT(G, v), the loop on lines 4–7 executes $|Adj[v]|$ times. Since

$$\sum_{v \in V} |Adj[v]| = \Theta(E) ,$$

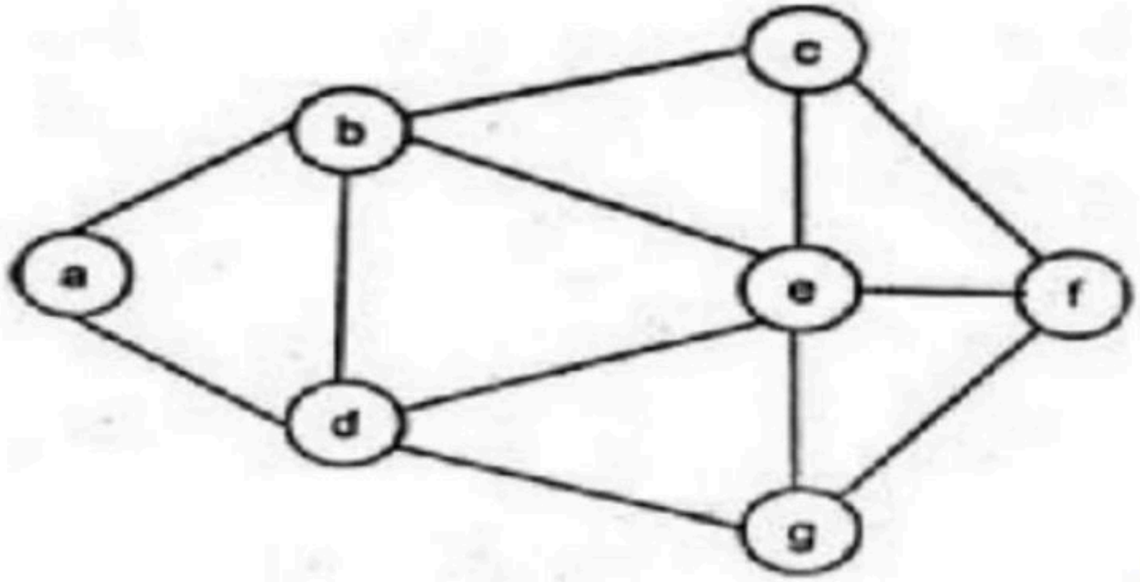
the total cost of executing lines 4–7 of DFS-VISIT is $\Theta(E)$. The running time of DFS is therefore $\Theta(V + E)$.

Break

Q Consider the following sequence of nodes for the undirected graph given below.

1) a b e f d g c **2)** a b e f c g d **3)** a d g e b c f **4)** a d b c g e f

A Depth First Search (DFS) is started at node a. The nodes are listed in the order they are first visited. Which all of the above is (are) possible output(s)? **(Gate-2008) (2 Marks)**



(A) 1 and 3 only
(B) 2 and 3 only
(C) 2, 3 and 4 only
(D) 1, 2, and 3

Q Consider the following graph

Among the following sequences

I) a b e g h f **II)** a b f e h g **III)** a b f h g e **IV)** a f g h b e

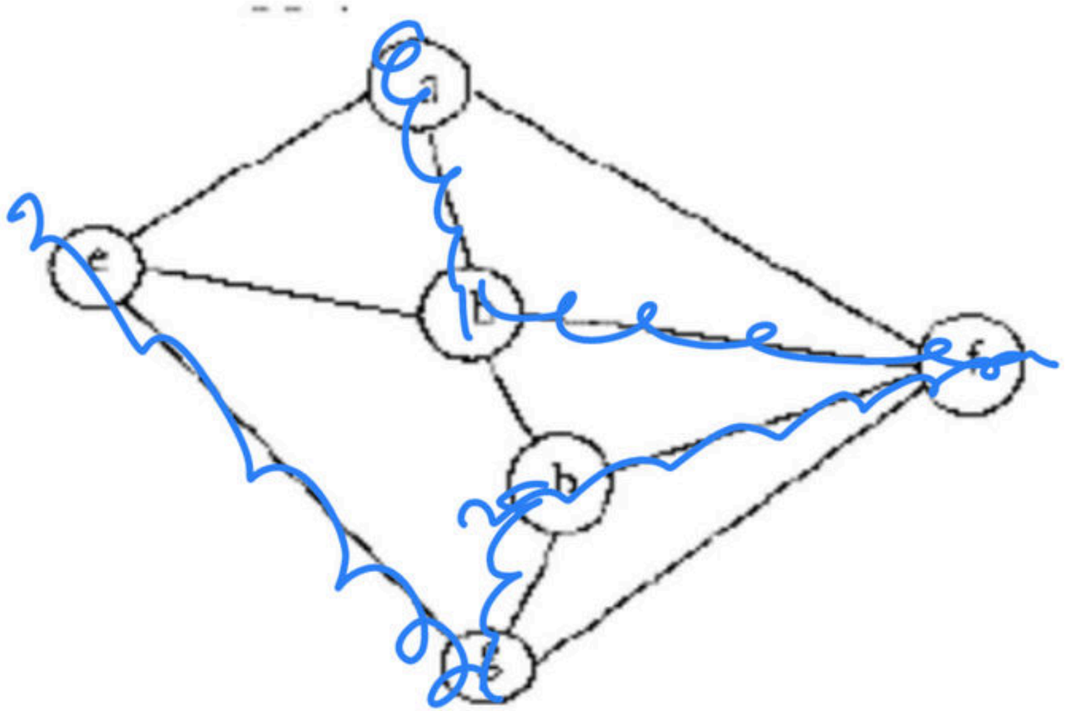
Which are depth first traversals of the above graph? **(GATE-2003) (1 Marks)**

(A) I, II and IV only

(B) I and IV only

(C) II, III and IV only

(D) I, III and IV only



Break

Q.29 An *articulation point* in a connected graph is a vertex such that removing the vertex and its incident edges disconnects the graph into two or more connected components. Let T be a DFS tree obtained by doing DFS in a connected undirected graph G. **(GATE-2021)**

Which of the following options is/are correct?

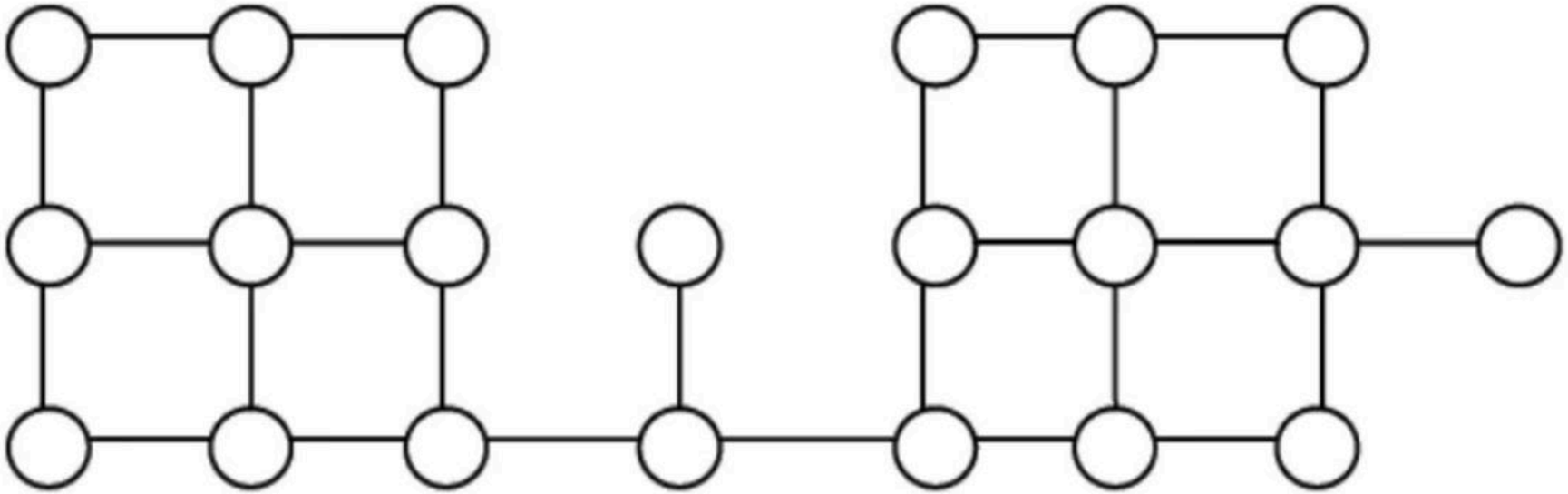
(a) Root of T is an articulation point in G if and only if it has 2 or more children.

(b) A leaf of T can be an articulation point in G.

(c) Root of T can never be an articulation point in G.

(d) If u is an articulation point in G such that x is an ancestor of u in T and y is a descendent of u in T, then all paths from x to y in G must pass through u.

Q Suppose depth first search is executed on the graph below starting at some unknown vertex. Assume that a recursive call to visit a vertex is made only after first checking that the vertex has not been visited earlier. Then the maximum possible recursion depth (including the initial call) is _____. **(Gate-2014) (2 Marks)**



Q Let G be a graph with n vertices and m edges. What is the tightest upper bound on the running time on Depth First Search of G? Assume that the graph is represented using adjacency matrix.**(Gate-2014) (1 Marks)**

(A) O(n) **(B)** O(m+n) **(C)** $O(n^2)$ **(D)** O(mn)

Q Let T be a depth first search tree in an undirected graph G. Vertices u and n are leaves of this tree T. The degrees of both u and n in G are at least 2. which one of the following statements is true? **(Gate-2006) (2 Marks)**

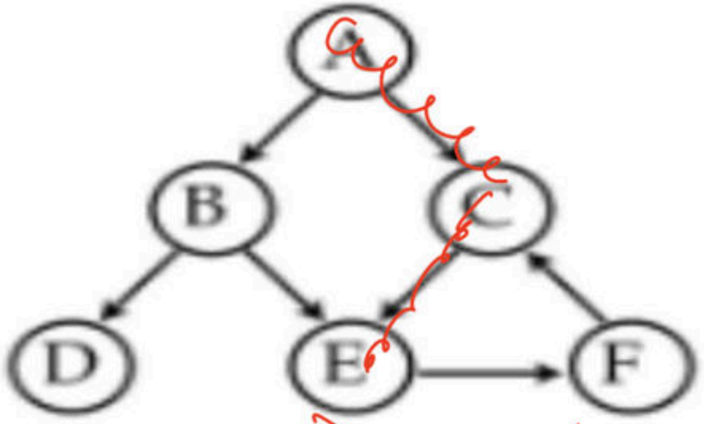
(A) There must exist a vertex w adjacent to both u and n in G

(B) There must exist a vertex w whose removal disconnects u and n in G

(C) There must exist a cycle in G containing u and n

(D) There must exist a cycle in G containing u and all its neighbors in G.

Depth ion travels of the following directed graph is : (UGC - June – 2007)



(A) A B C D E F

(B) A B D E F C

(C) A C E B D F

(D) None of the above

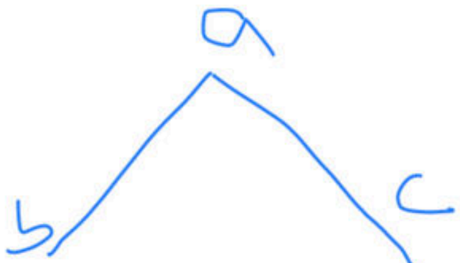
Q Let G be an undirected graph. Consider a depth first traversal of G, and let T be the resulting depth-first search tree. Let u be a vertex in G and let v be the first new (unvisited) vertex visited after visiting u in **the traversal. Which of the following statements is always true? (GATE-2000) (2 Marks)**

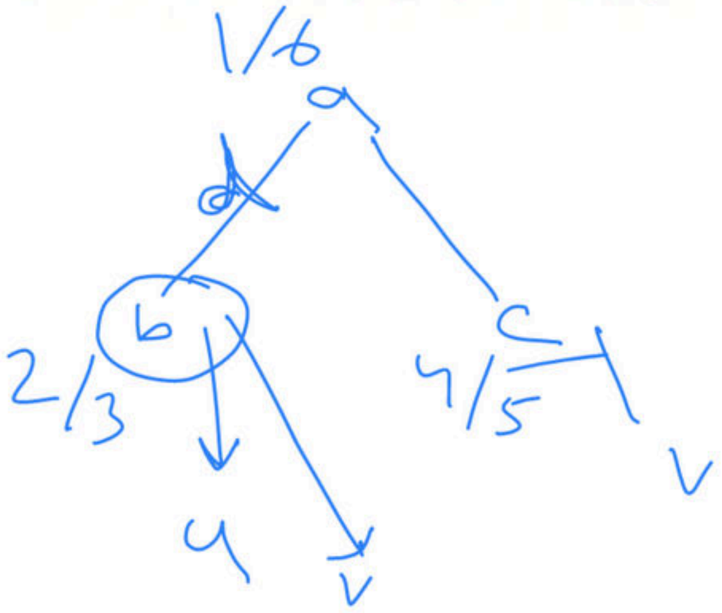
(A) {u,v} must be an edge in G, and u is a descendant of v in T

(B) {u,v} must be an edge in G, and v is a descendant of u in T

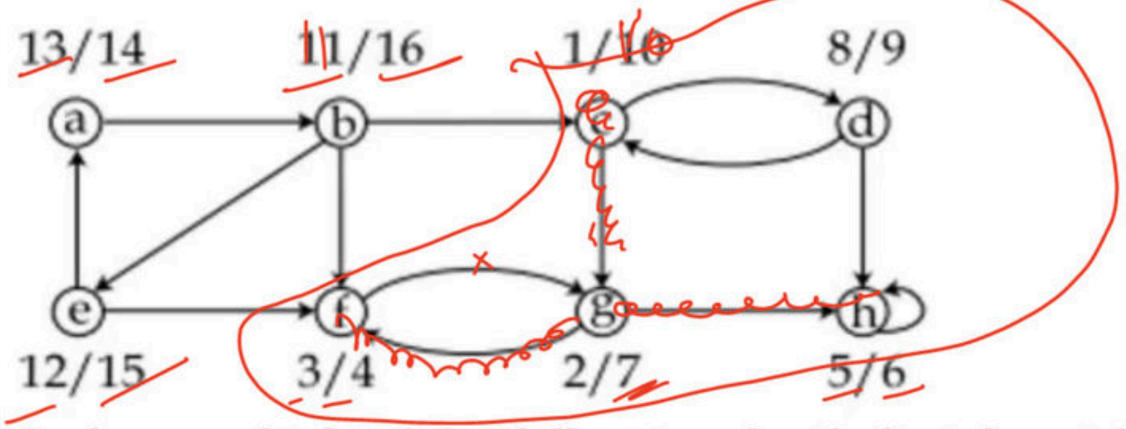
(C) If {u,v} is not an edge in G then u is a leaf in T

(D) If {u,v} is not an edge in G then u and v must have the same parent in T





In the following graph, discovery time stamps and finishing time stamps of Depth First Search (DFS) are shown as x/y, where x is discovery time stamp and y is finishing time stamp. (UGC - June – 2017)



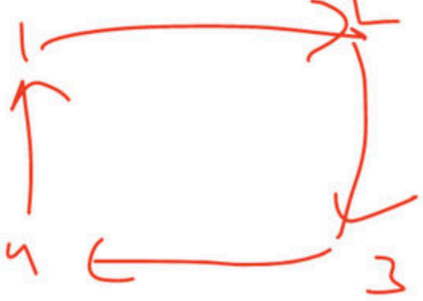
It shows which of the following depth first forest ?

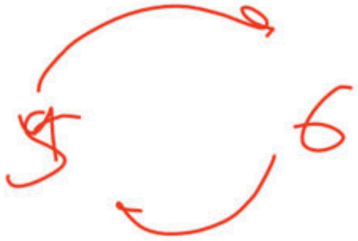
(1) {a, b, e} {c, d, f, g, h}

(2) {a, b, e} {c, d, h} {f, g}

(3) {a, b, e} {f, g} {c, d} {h}

(4) {a, b, c, d} {e, f, g} {h}





Break

Breadth First Traversal (or Search)

- Breadth First Traversal (or Search) for a graph is similar to Breadth First Traversal of a tree. The only catch here is, unlike trees, graphs may contain cycles, so we may come to the same node again.

- To avoid processing a node more than once, we use a Boolean visited array. For simplicity, it is assumed that all vertices are reachable from the starting vertex, i.e. the graph is connected

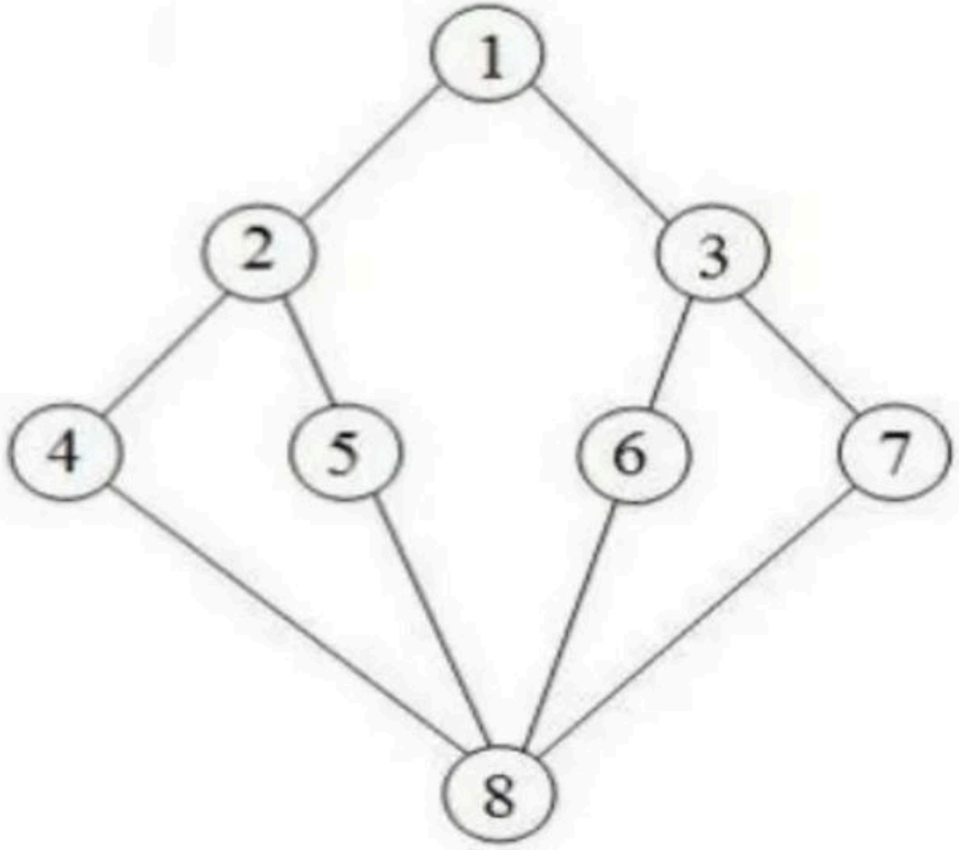
Q Which of the following are valid and invalid BFS traversal sequence

a) 1, 3, 2, 5, 4, 7, 6, 8

b) 1, 3, 2, 7, 6, 4, 5, 8

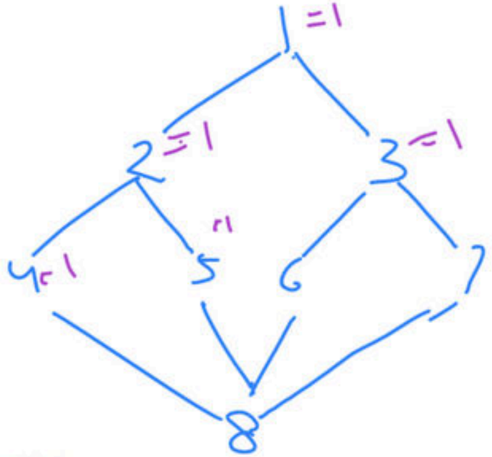
c) 1, 2, 3, 5, 4, 7, 6, 8

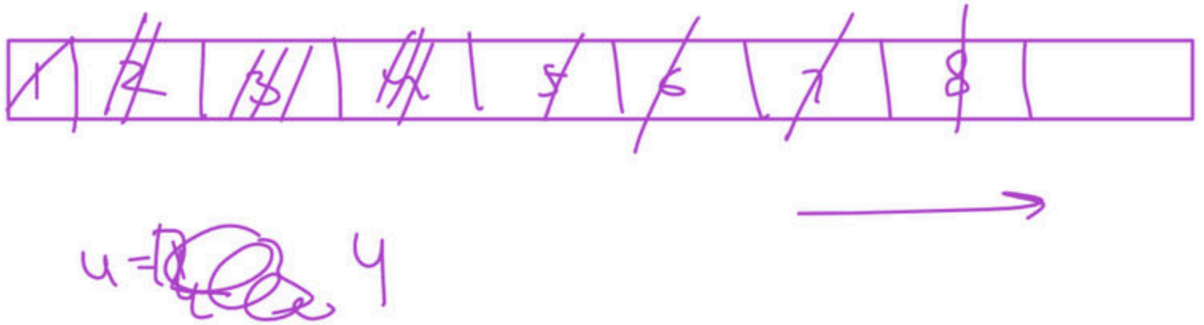
d) 1, 2, 3, 7, 5, 6, 4, 8



1 3 2 7 6 4 5 8

```
BFS(v)
{
        visited(v) = 1
        insert[V,Q]
        While(Q != Phi)
        {
                u = Delete(Q);
                for all x adjacent to u
                {
                        if (x is not visited)
                                {
                                        visited(x) = 1
                                        insert(x,Q)
                                }
                }
        }
}
```





u = 4

```
BFS(G, s)
 1  for each vertex u ∈ G.V − {s}
 2      u.color = WHITE
 3      u.d = ∞
 4      u.π = NIL
 5  s.color = GRAY
 6  s.d = 0
 7  s.π = NIL
 8  Q = ∅
 9  ENQUEUE(Q, s)
10  while Q ≠ ∅
11      u = DEQUEUE(Q)
12      for each v ∈ G.Adj[u]
13          if v.color == WHITE
14              v.color = GRAY
15              v.d = u.d + 1
16              v.π = u
17              ENQUEUE(Q, v)
18      u.color = BLACK
```

Before proving the various properties of breadth-first search, we take on the somewhat easier job of analyzing its running time on an input graph $G = (V, E)$. We use aggregate analysis, as we saw in Section 17.1. After initialization, breadth-first search never whitens a vertex, and thus the test in line 13 ensures that each vertex is enqueued at most once, and hence dequeued at most once. The operations of enqueuing and dequeuing take $O(1)$ time, and so the total time devoted to queue operations is $O(V)$. Because the procedure scans the adjacency list of each vertex only when the vertex is dequeued, it scans each adjacency list at most once. Since the sum of the lengths of all the adjacency lists is $\Theta(E)$, the total time spent in scanning adjacency lists is $O(E)$. The overhead for initialization is $O(V)$, and thus the total running time of the BFS procedure is $O(V + E)$. Thus, breadth-first search runs in time linear in the size of the adjacency-list representation of G.

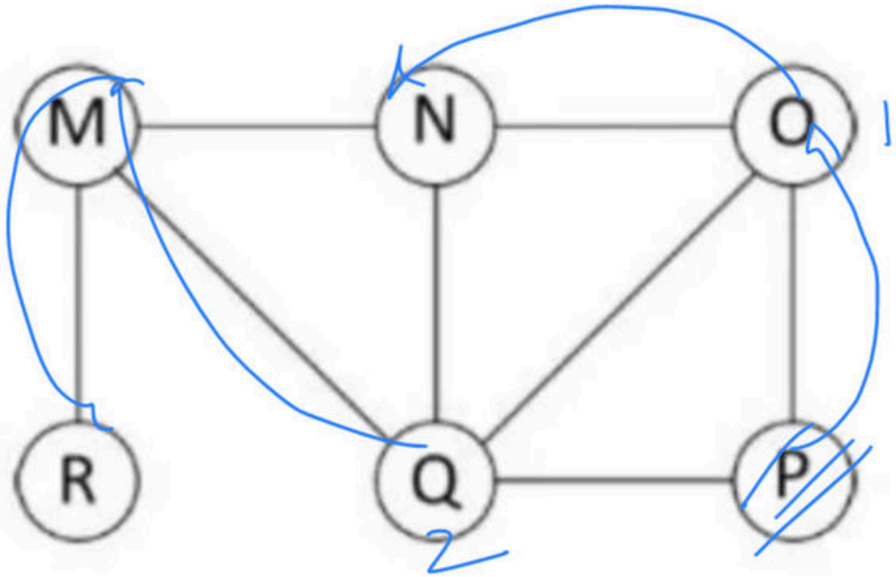
Q Breath First Search (BFS) has been implemented using queue data structure. Which one of the following is a possible order of visiting the nodes in the graph above? **(Gate-2017) (1 Marks)**

a) MNOPQR

b) NQMPOR

c) QMNROP

d) POQNMR



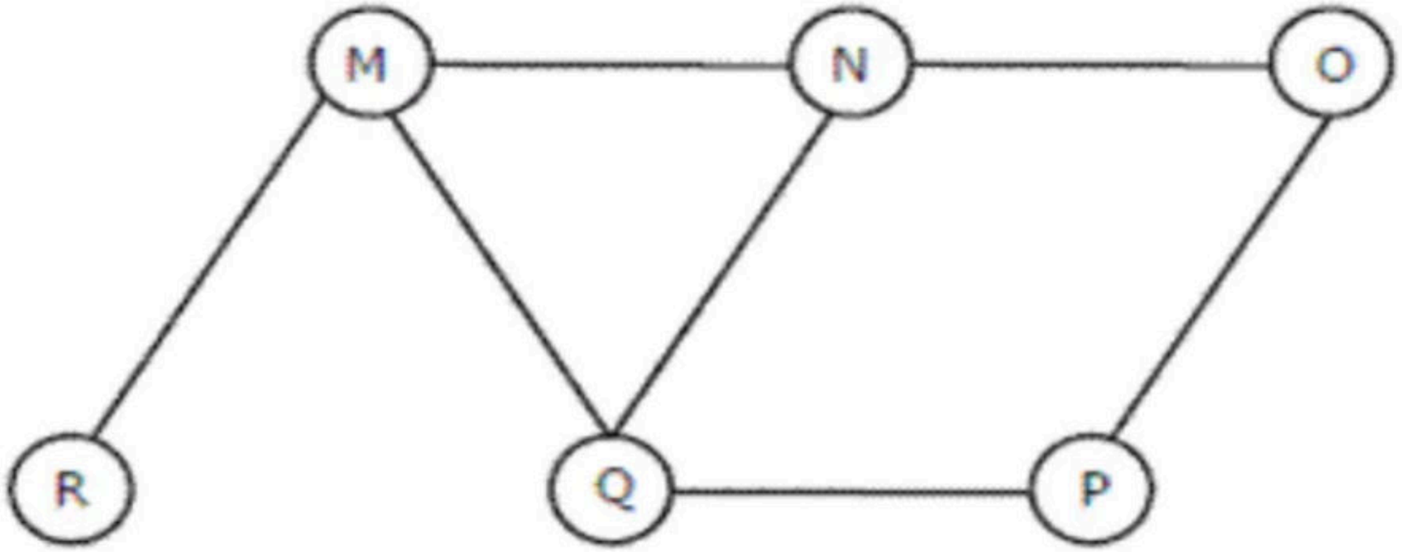
Q The Breadth First Search algorithm has been implemented using the queue data structure. One possible order of visiting the nodes of the following graph is **(Gate-2008) (1 Marks)**

(A) MNOPQR

(B) NQMPOR

(C) QMNPRO

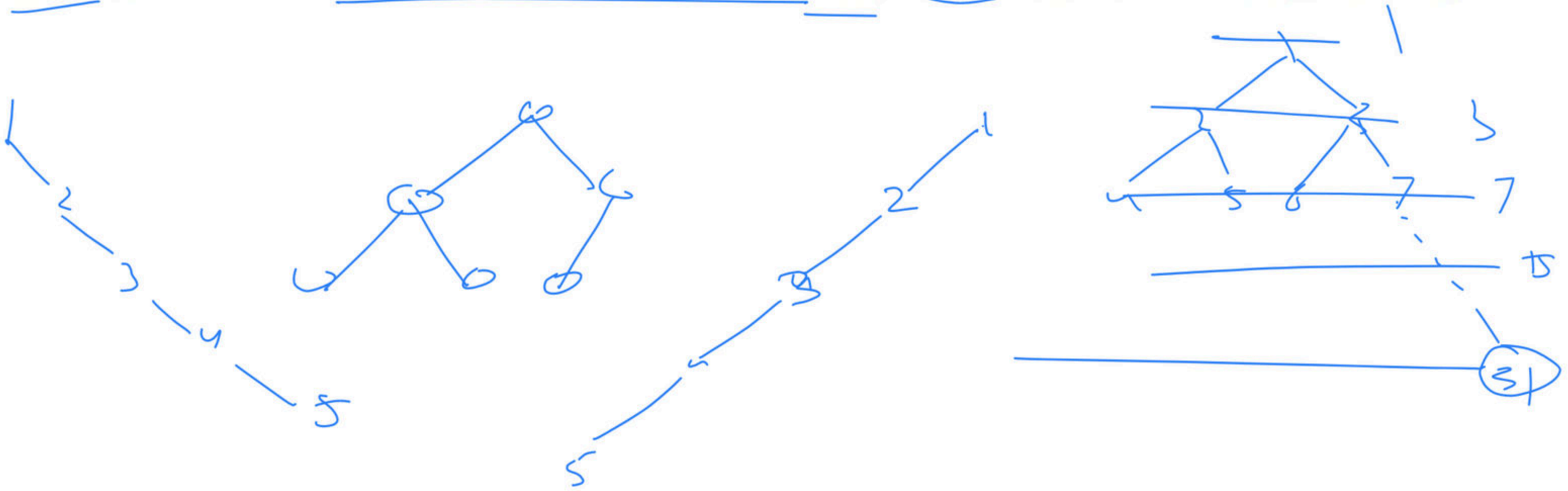
(D) QMNPOR



Break

Q.15 Consider a complete binary tree with 7 nodes. Let A denote the set of first 3 elements obtained by performing Breadth-First Search (BFS) starting from the root. Let B denote the set of first 3 elements obtained by performing Depth-First Search (DFS) starting from the root. The value of $|A - B|$ is _____. **(GATE-2021)**

Q Breadth First Search (BFS) is started on a binary tree beginning from the root vertex. There is a vertex t at a distance four from the root. If t is the n-th vertex in this BFS traversal, then the maximum possible value of n is 31 **(Gate-2016) (2 Marks)**



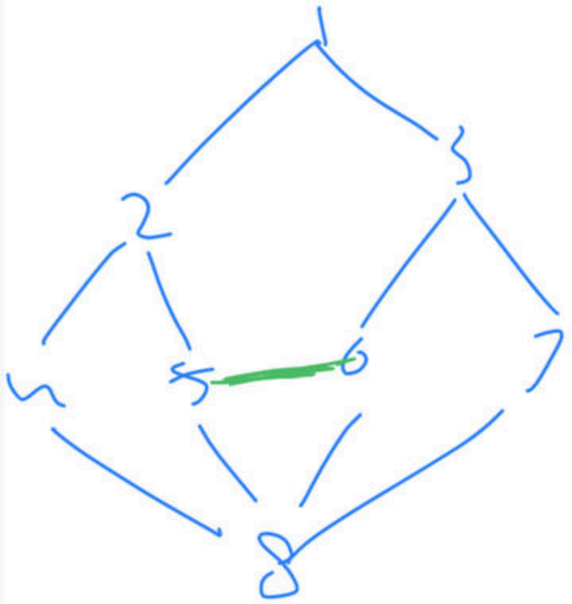
Q Let G = (V, E) be a simple undirected graph, and s be a particular vertex in it called the source. For x ∈ V, let d(x) denote the shortest distance in G from s to x. A breadth first search (BFS) is performed starting at s. Let T be the resultant BFS tree. If (u, v) is an edge of G that is not in T, then which one of the following CANNOT be the value of d(u) – d(v)? **(Gate-2015)(2 Marks)**

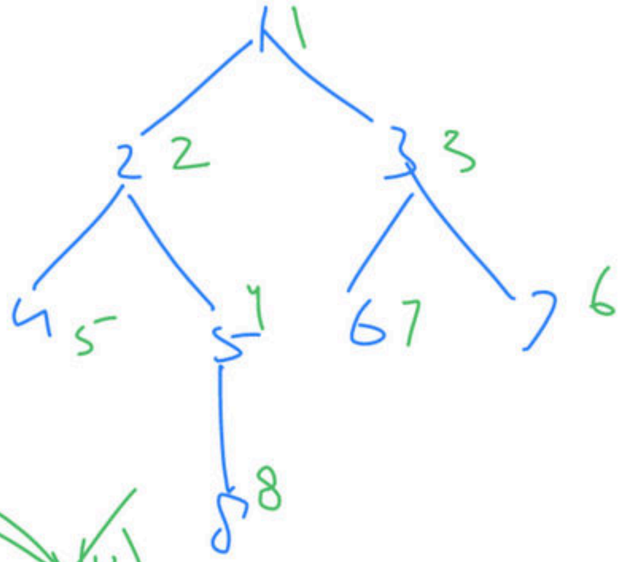
(A) -1

(B) 0

(C) 1

(D) 2





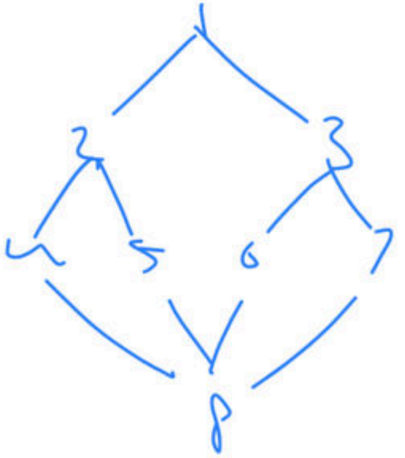
Q Consider the tree arcs of a BFS traversal from a source node W in an unweighted, connected, undirected graph. The tree T formed by the tree arcs is a data structure for computing. **(Gate-2014) (2 Marks)**

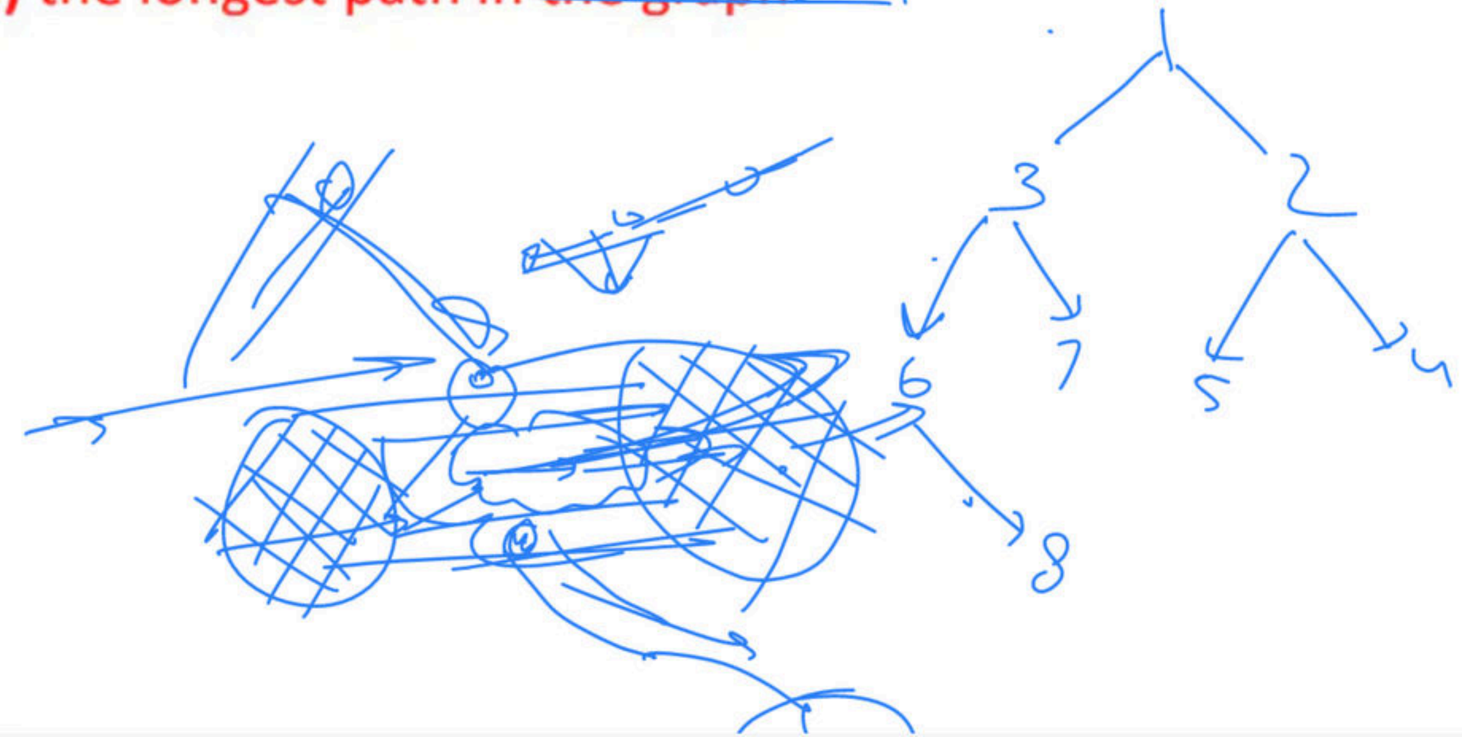
a) the shortest path between every pair of vertices.

b) the shortest path from W to every vertex in the graph.

c) the shortest paths from W to only those nodes that are leaves of T.

d) the longest path in the graph





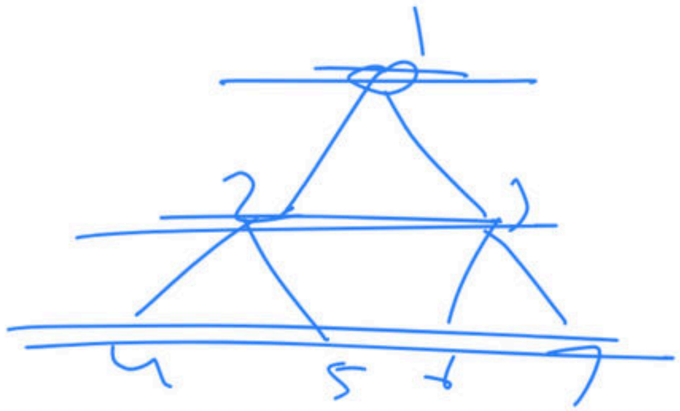
Q Level order traversal of a rooted tree can be done by starting from the root and performing **(Gate-2004) (1 Marks)**

(A) preorder traversal

(B) inorder traversal

(C) depth first search

(D) breadth first search



L.O ⟶ BFS

Q Consider an undirected unweighted graph G. Let a breadth first traversal of G be done starting from a node r. Let d(r, u) and d(r, v) be the lengths of the shortest paths from r to u and v respectively, in G. If u is visited before v during the breadth-first traversal, which of the following statements is correct? **(GATE-2001) (2 Marks)**

(A) d(r, u) < d (r, v) **(B)** d(r, u) > d(r, v) **(C)** d(r, u) <= d (r, v) **(D)** None of the above

Break

Q Let G be a simple undirected graph. Let T_D be a depth first search tree of G. Let T_B be a breadth first search tree of G. Consider the following statements.

(I) No edge of G is a cross edge with respect to T_D. (A cross edge in G is between two nodes neither of which is an ancestor of the other in T_D).

(II) For every edge (u, v) of G, if u is at depth i and v is at depth j in T_B, then $|i - j| = 1$.

Which of the statements above must necessarily be true? **(Gate-2018) (2 Marks)**

a) I only **b)** II only **c)** Both I and II **d)** Neither I nor II

Q The most efficient algorithm for finding the number of connected components in an undirected graph on n vertices and m edges has time complexity. **(Gate-2008)(1 Marks)**

a) O(n) **b)** O(m) **c)** O(m+n) **d)** O(m.n)