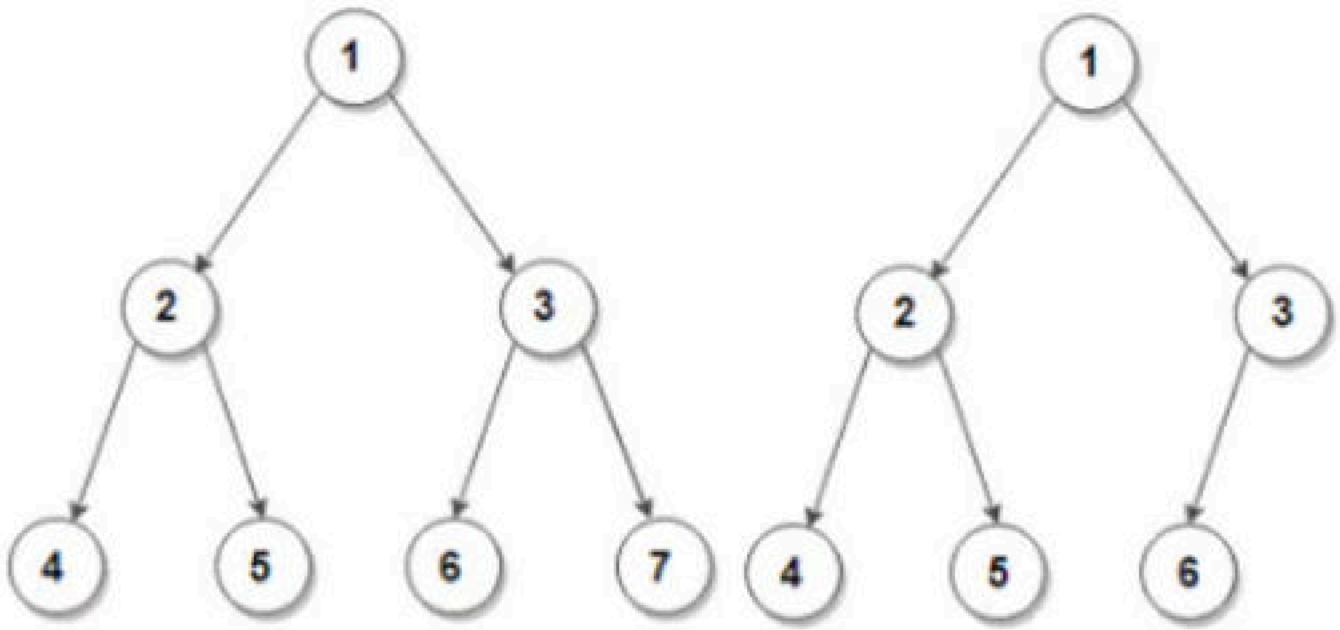
# Decision Control in C Programming

Complete Course on Data Structures for GATE

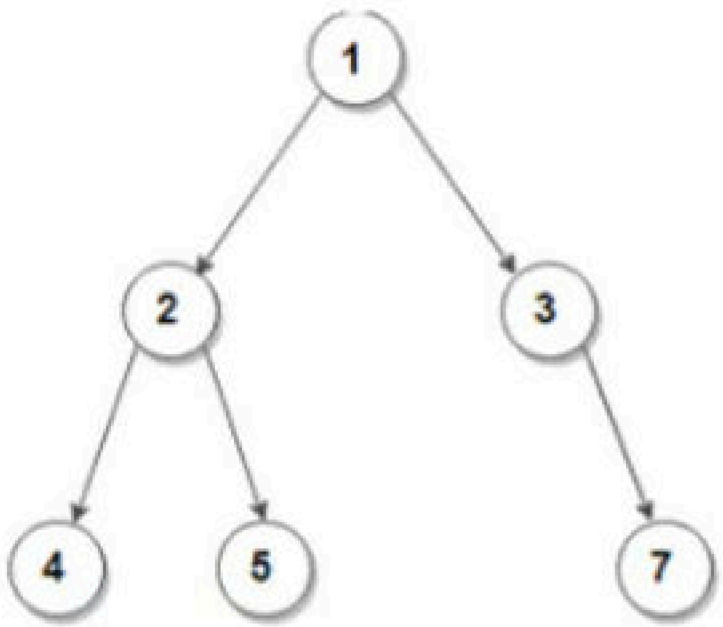
**Sanchit Jain** • Lesson 25 • July 16, 2022

# Complete Binary Tree

- Consider a binary tree T, the maximum number of nodes at height h is $2^h$ nodes.
- The binary tree T is said to be complete binary tree, if all its level except possibly the last, have the maximum number of nodes and if all the nodes at the last level appear as far left as possible.

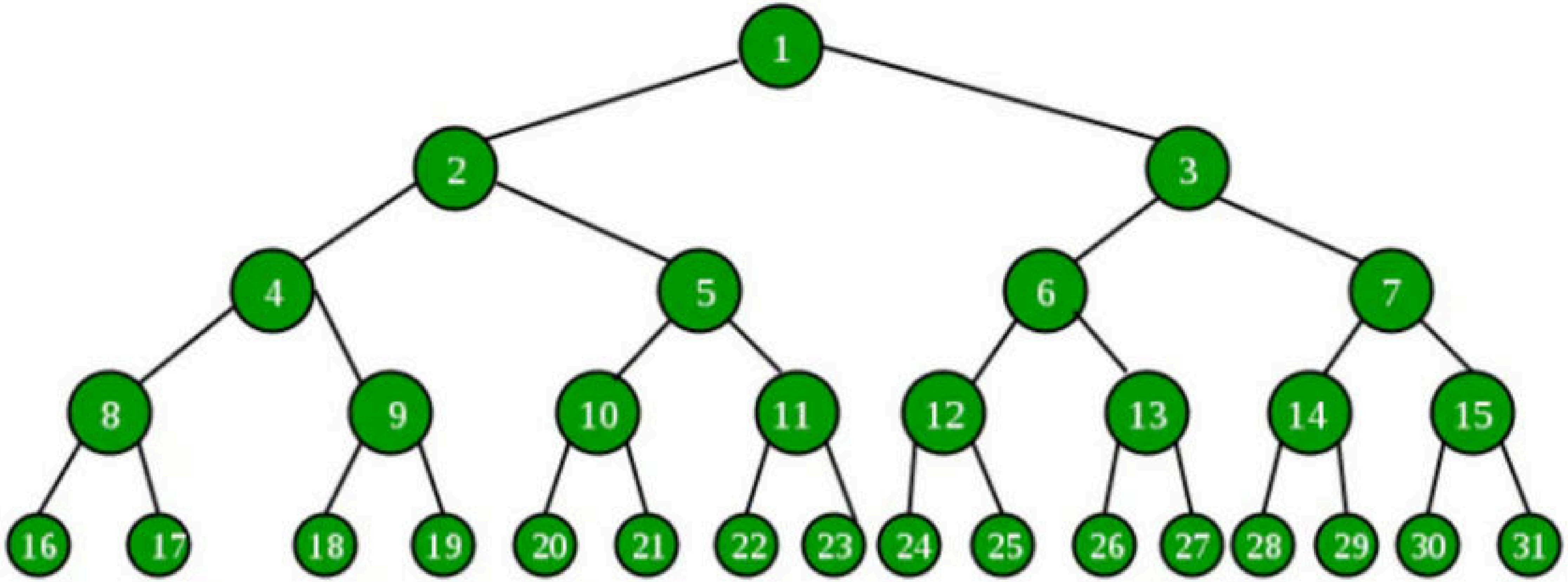



Complete Binary Tree




Not a Complete Binary Tree

- One can easily determine the children and parent of a node k in any complete tree T
- Specially the left and right children of the node K are 2*k, 2*k + 1 and the parent of k is the node lower bound(k/2)



**Q** A scheme for storing binary trees in an array X is as follows. Indexing of X starts at 1 instead of 0. the root is stored at X[1]. For a node stored at X[i], the left child, if any, is stored in X[2i] and the right child, if any, in X[2i+1]. To be able to store any binary tree on n vertices the minimum size of X should be. **(GATE - 2006) (2 Marks)**

**(A)** $\log_2 n$ **(B)** n **(C)** $2n + 1$ **(D)** $2^n - 1$

**Q** Let LASTPOST, LASTIN and LASTPRE denote the last vertex visited in a post order, inorder and preorder traversal, respectively, of a complete binary tree. Which of the following is always true? **(GATE - 2000) (1 Marks)**

**(A)** LASTIN = LASTPOST

**(B)** LASTIN = LASTPRE

**(C)** LASTPRE = LASTPOST

**(D)** None of the above

**Q**. What are the children for node ‘w’ of a complete-binary tree in an array representation? **[Asked in Hexaware 2018]**

**a)** 2w and 2w+1
**b)** 2+w and 2-w
**c)** w+1/2 and w/2
**d)** w-1/2 and w+1/2

Answer: a

Explanation: The left child is generally taken as 2*w whereas the right child will be taken as 2*w+1 because root node is present at index 0 in the array and to access every index position in the array.

**Q** If the tree is not a complete binary tree then what changes can be made for easy access of children of a node in the array? **[Asked in Goldman Sachs ]**

**a)** every node stores data saying which of its children exist in the array
**b)** no need of any changes continue with 2w and 2w+1, if node is at i
**c)** keep a separate table telling children of a node
**d)** use another array parallel to the array with tree

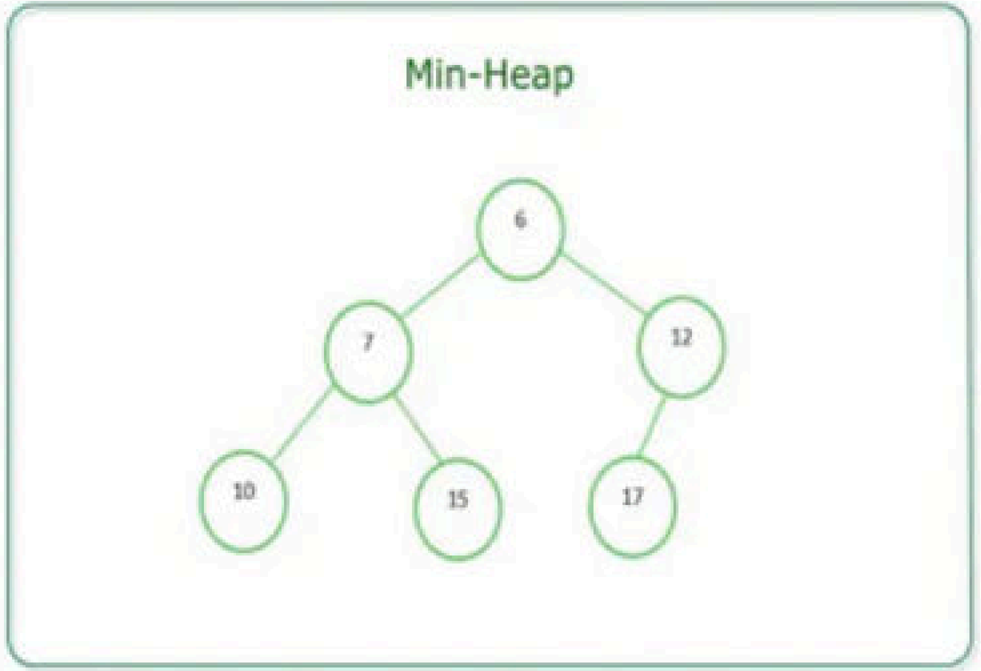
Answer: a

Explanation: Array cannot represent arbitrary shaped trees. It can only be used in case of complete trees. If every node stores data saying that which of its children exists in the array then elements can be accessed easily.

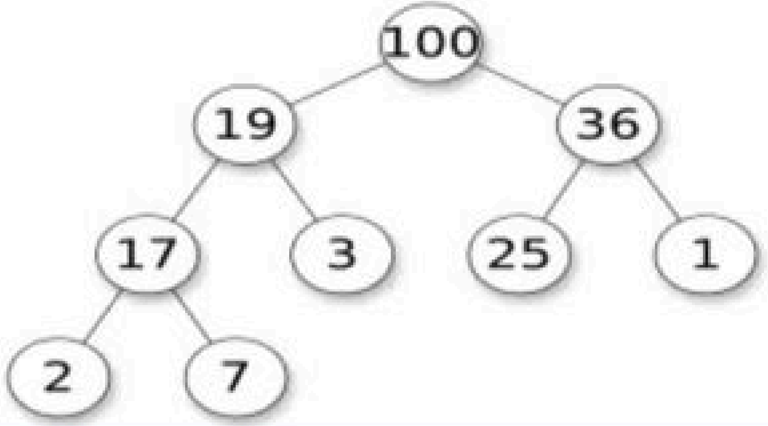
# Break

# Heap

- Suppose H is a complete binary tree with n elements, H is called a Heap, if each node N of H has following properties:
  - The value of N is greater than to the value at each of the children of N then it is called Max heap.
  - A min heap is defined as the value at N is less than the value at any of the children of N.

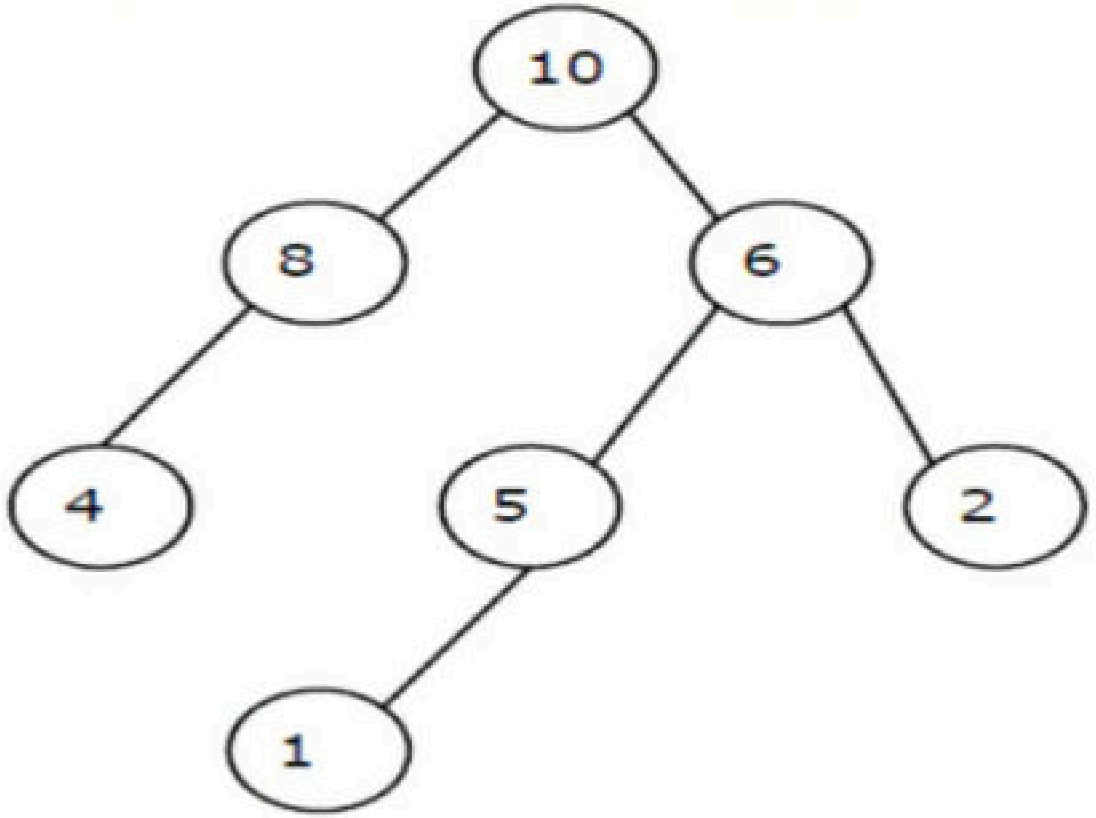



**Tree representation**

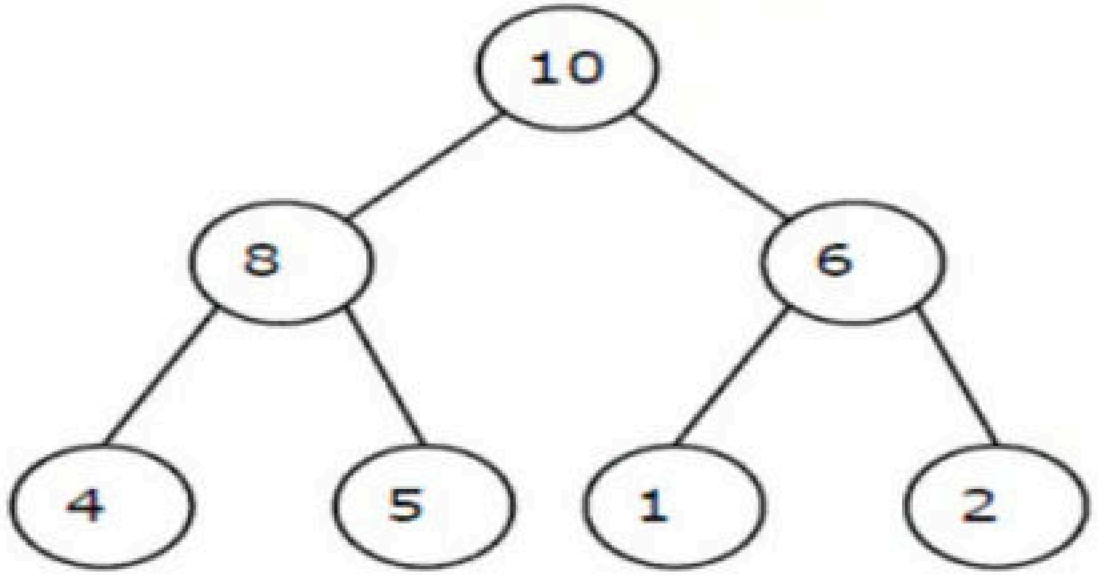


**Q** A max-heap is a heap where the value of each parent is greater than or equal to the values of its children. Which of the following is a max-heap? **(GATE - 2011) (2 Marks)**

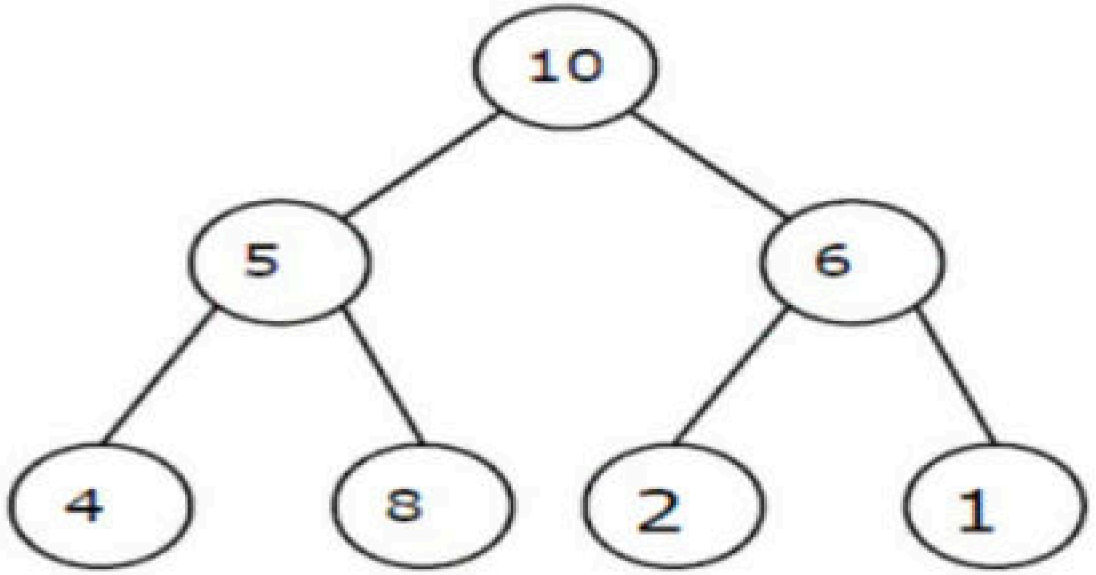
(A)



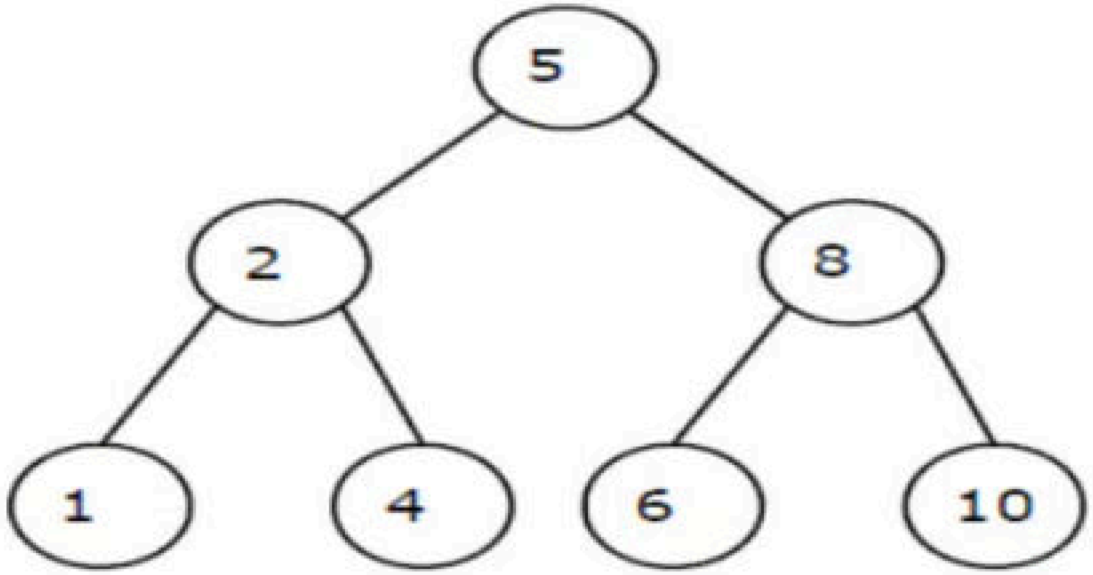

(B)




(C)




(D)



**Q** Consider a binary max-heap implemented using an array. Which one of the following arrays represents a binary max-heap? **(GATE - 2006) (Marks)**

**(A)** 23,17,14,6,13,10,1,12,7,5

**(B)** 23,17,14,6,13,10,1,5,7,12

**(C)** 23,17,14,7,13,10,1,5,6,12

**(D)** 23,17,14,7,13,10,1,12,5,7

**Q** Consider any array representation of an n element binary heap where the elements are stored from index 1 to index n of the array. For the element stored at index i of the array (i <= n), the index of the parent is

**(GATE - 2001) (1 Marks)**

**(A)** i – 1 **(B)** floor(i/2) **(C)** ceiling(i/2) **(D)** (i+1)/2

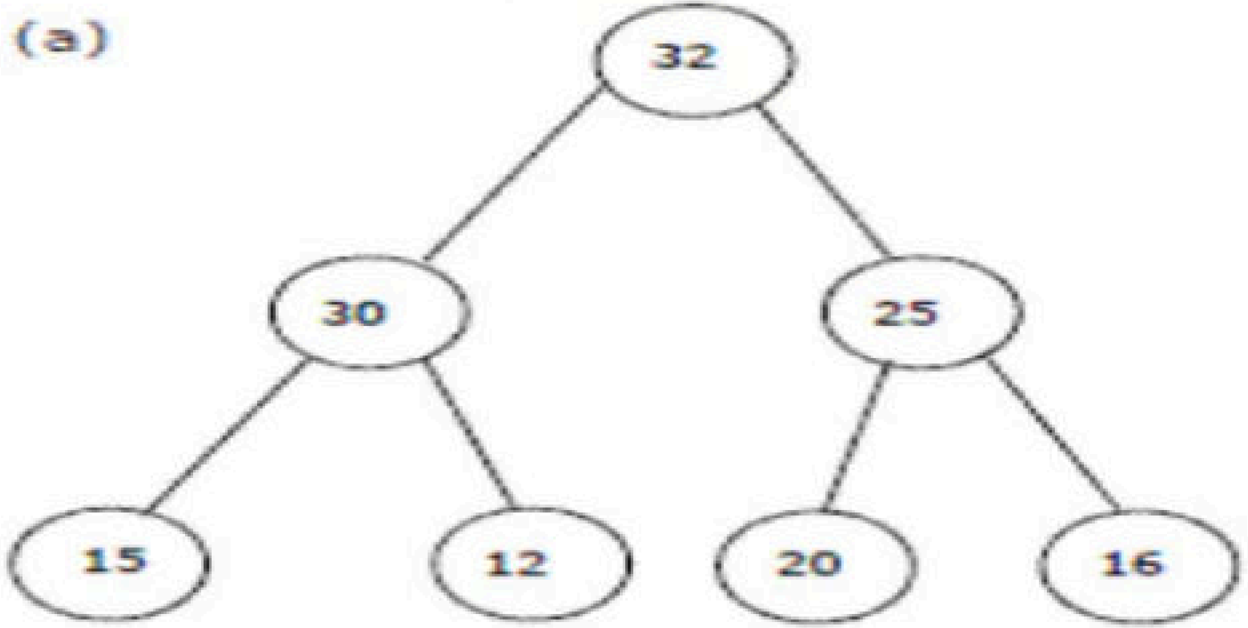
# Break

**Q** The number of possible min-heaps containing each value from {1, 2, 3, 4, 5, 6, 7} exactly once is ________. **(Gate-2018) (1 Marks)**

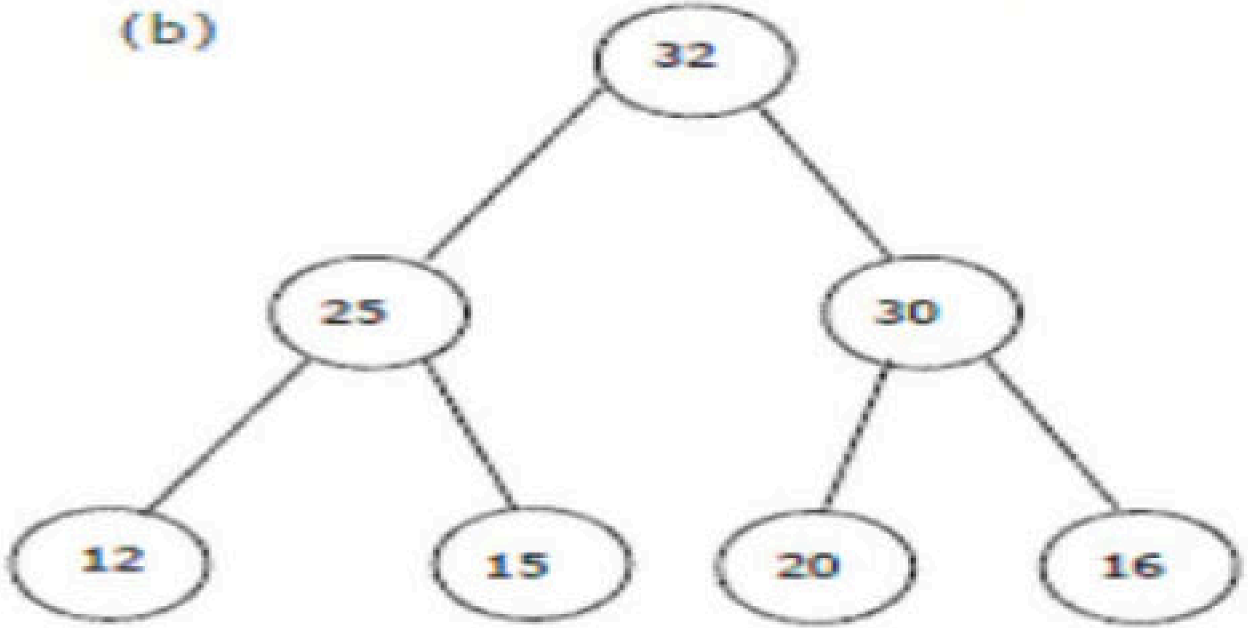
**Q** A complete binary min-heap is made by including each integer in [1, 1023] exactly once. The depth of a node in the heap is the length of the path from the root of the heap to that node. Thus, the root is at depth 0. The maximum depth at which integer 9 can appear is _____________ **(Gate-2016) (1 Marks)**

**Q** The elements 32, 15, 20, 30, 12, 25, 16 are inserted one by one in the given order into a Max Heap. The resultant Max Heap is. **(GATE - 2004) (2 Marks)**

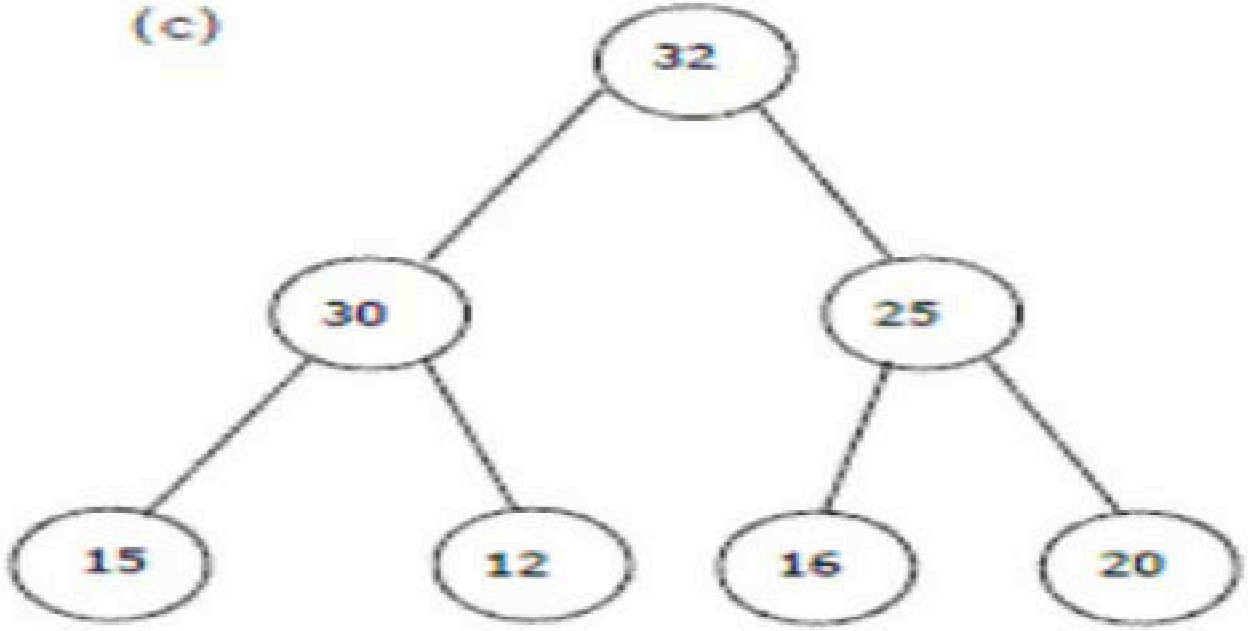
(a)



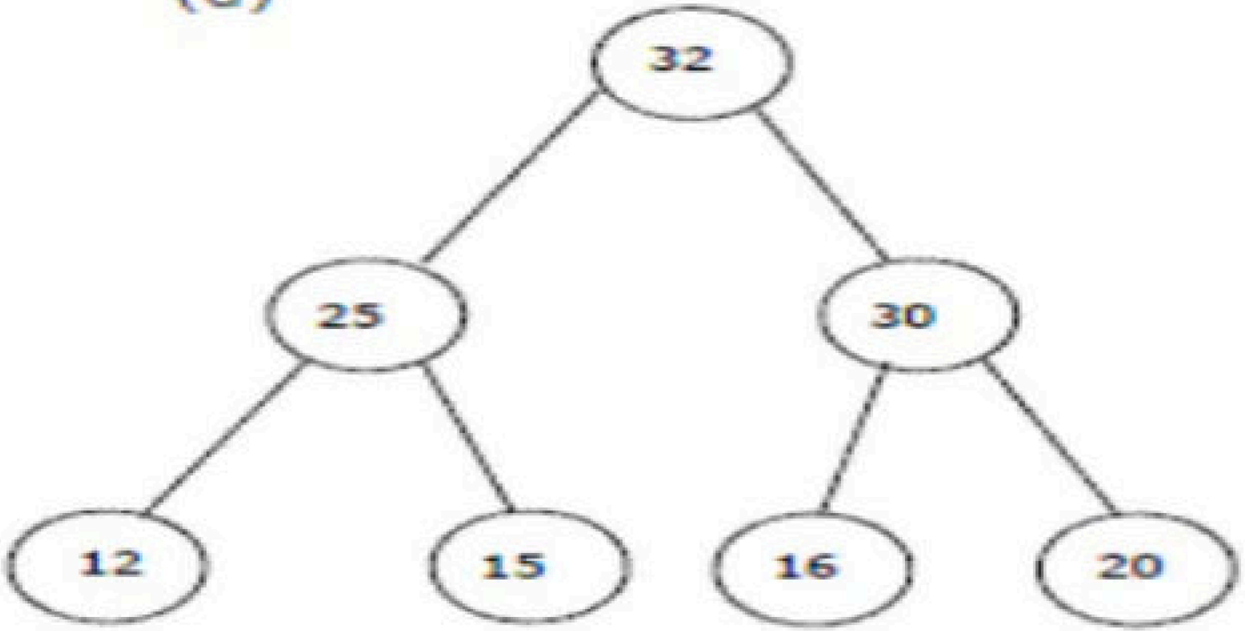

(b)




(c)




(d)



**Q** The elements 32, 15, 20, 30, 12, 25, 16 are inserted one by one in the given order into a Max Heap. The resultant Max Heap is. **(GATE - 2004) (2 Marks)**

# Break

**Q** Consider the following array of elements. ⟨89, 19, 50, 17, 12, 15, 2, 5, 7, 11, 6, 9, 100⟩. The minimum number of interchanges needed to convert it into a max-heap is **(GATE - 2015) (2 Marks)**

**(A)** 4 **(B)** 5 **(C)** 2 **(D)** 3

**Q** Consider a max heap, represented by the array: 40, 30, 20, 10, 15, 16, 17, 8, 4.
Now consider that a value 35 is inserted into this heap. After insertion, the new heap is **(GATE - 2015) (2 Marks)**

| Array index | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| Value | 40 | 30 | 20 | 10 | 15 | 16 | 17 | 8 | 4 |

**a)** 40,30,20,10,15,16,17,8,4,35

**b)** 40,35,20,10,30,16,17,8,4,15

**c)** 40,30,20,10,35,16,17,8,4,15

**d)** 40,35,20,10,15,16,17,8,4,30

**Q** A priority queue is implemented as a Max-Heap. Initially, it has 5 elements. The level-order traversal of the heap is: 10, 8, 5, 3, 2. Two new elements 1 and 7 are inserted into the heap in that order. The level-order traversal of the heap after the insertion of the elements is: **(GATE - 2014) (2 Marks)**

**(A)** 10, 8, 7, 3, 2, 1, 5

**(B)** 10, 8, 7, 2, 3, 1, 5

**(C)** 10, 8, 7, 1, 2, 3, 5

**(D)** 10, 8, 7, 5, 3, 2, 1

**Q** Consider a binary max-heap implemented using an array. Which one of the following arrays represents a binary max-heap? **(GATE - 2009) (2 Marks)**

**(A)** 25,12,16,13,10,8,14

**(B)** 25,12,16,13,10,8,14

**(C)** 25,14,16,13,10,8,12

**(D)** 25,14,12,13,10,8,16

**Q** What is the content of the array after two delete operations on the correct answer to the previous question? **(GATE - 2009) (2 Marks)**

**(A)** 14,13,12,10,8

**(B)** 14,12,13,8,10

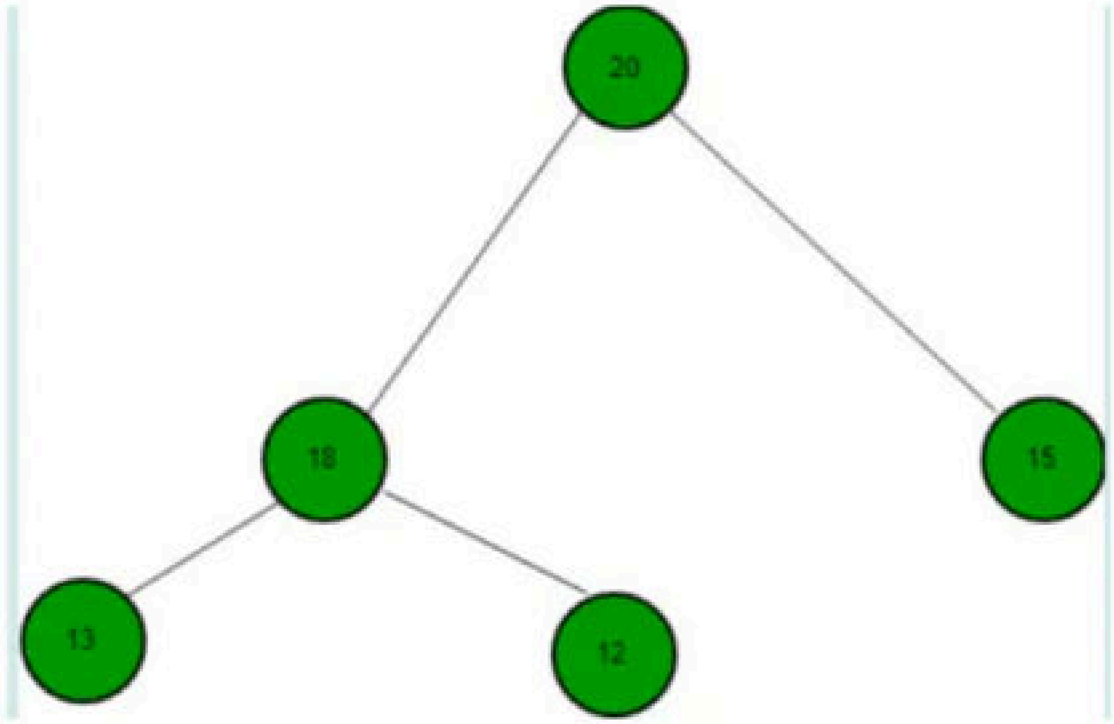
**(C)** 14,13,8,12,10

**(D)** 14,13,12,8,10

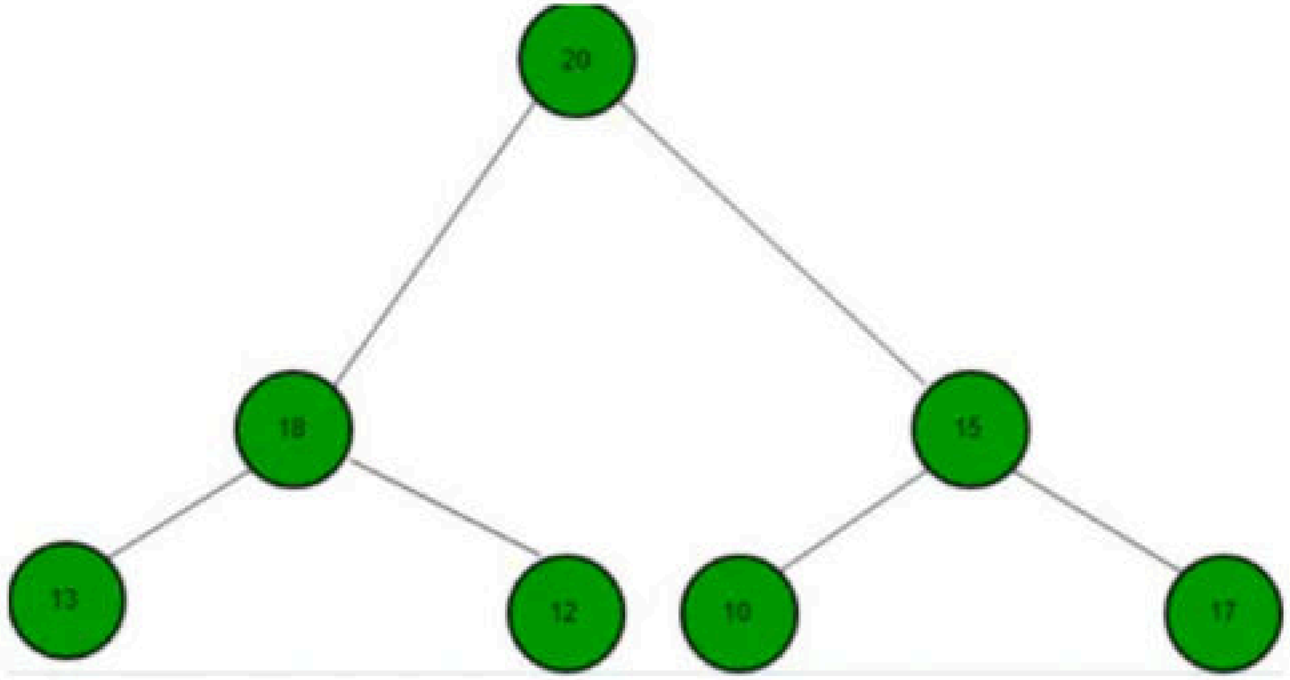
Q. A priority queue is implemented as a max-heap. Initially, it has five elements. The level-order traversal of the heap is as follows: 20, 18, 15, 13, 12 Two new elements '10' and '17' are inserted in the heap in that order. The level-order traversal of the heap after the insertion of the element is: **[Asked in TCS NQT 2019 ]**

a. 20, 18, 17, 15, 13, 12, 10
b. 20, 18, 17, 12, 13, 10, 15
c. 20, 18, 17, 10, 12, 13, 15
d. 20, 18, 17, 13, 12, 10, 15

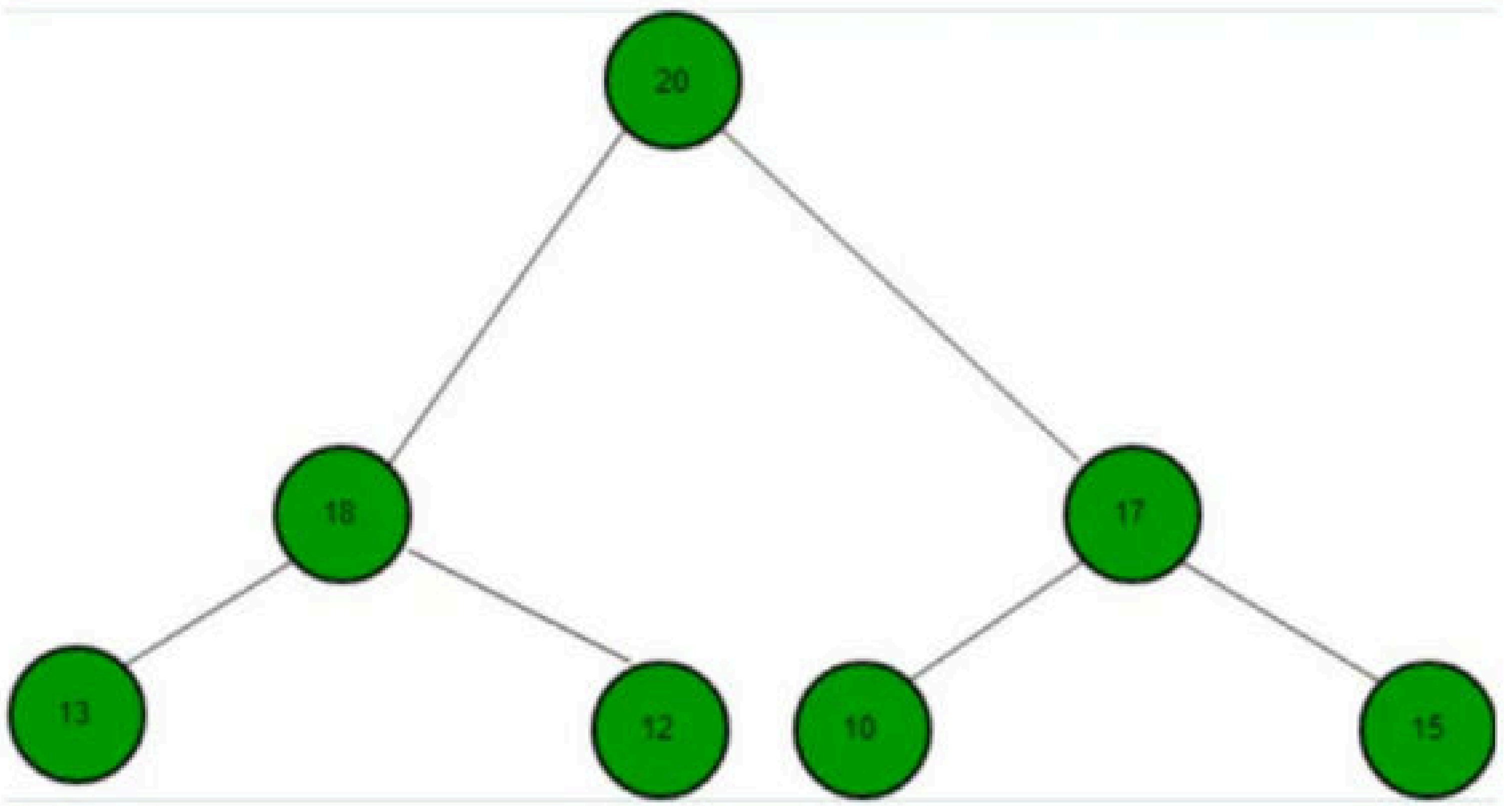
Initially we have:




When we insert 10 and 17:



We have to maintain max-heap, so:




The level-order traversal of the heap after the insertion of the element is 20, 18, 17, 13, 12, 10, 15 So, option (D) is correct.

# Break

**Q** Consider a rooted Binary tree represented using pointers. The best upper bound on the time required to determine the number of subtrees having exactly 4 nodes $O(n^a \operatorname{Log} n^b)$. Then the value of a + 10b is ________ **(GATE-2015) (1 Marks)**

**Q** Let T be a full binary tree with 8 leaves. (A full binary tree has every level full.) Suppose two leaves a and b of T are chosen uniformly and independently at random. The expected value of the distance between a and b in T (ie., the number of edges in the unique path between a and b) is (rounded off to 2 decimal places) _________. **(GATE-2019) (2 Marks)**

**Q** In a heap with n elements with the smallest element at the root, the 7th smallest element can be found in time **(GATE - 2008) (1 Marks)**

**a)** (n log n) **b)** (n) **c)** (log n) **d)** (1)

**Q** In a binary max heap containing n numbers, the smallest element can be found in time **(GATE - 2006) (1 Marks)**

**(A)** 0(n) **(B)** O(logn) **(C)** 0(loglogn) **(D)** 0(1)

**Q** A data structure is required for storing a set of integers such that each of the following operations can be done in O(logn) time, where n is the number of elements in the set.

**I)** Deletion of the smallest element

**II)** Insertion of an element if it is not already present in the set

Which of the following data structures can be used for this purpose? **(GATE - 2003) (2 Marks)**

**a)** A heap can be used but not a balanced binary search tree

**b)** A balanced binary search tree can be used but not a heap

**c)** Both balanced binary search tree and heap can be used

**d)** Neither balanced search tree nor heap can be used

# Break

**Q** We are given a set of n distinct elements and an unlabeled binary tree with n nodes. In how many ways can we populate the tree with the given set so that it becomes a binary search tree? **(GATE - 2011) (2 Marks)**

**(A)** 0 **(B)** 1 **(C)** n! **(D)** (1/(n+1)).2nCn

**Q** The maximum number of binary trees that can be formed with three unlabelled nodes is: **(GATE-2007) (1 Marks)**

**a)** 1 **b)** 5 **c)** 4 **d)** 3

**Q** How many distinct binary search trees can be created out of 4 distinct keys?

**(A)** 4 **(B)** 14 **(C)** 24 **(D)** 42

**Q** how many distinct BST can be constructed with 3 distinct keys?

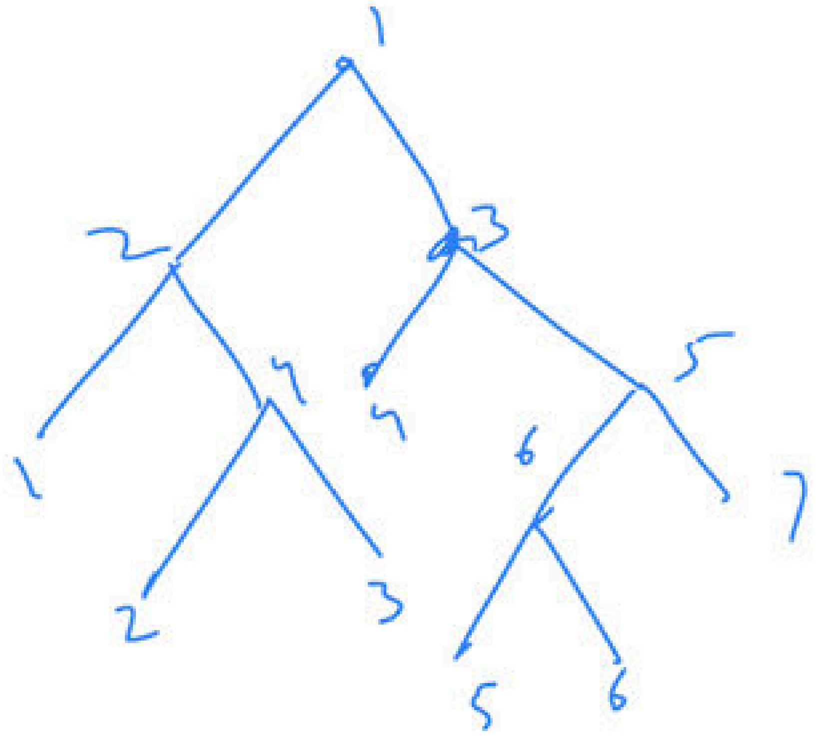
**a)** 4 **b)** 5 **c)** 6 **d)** 9

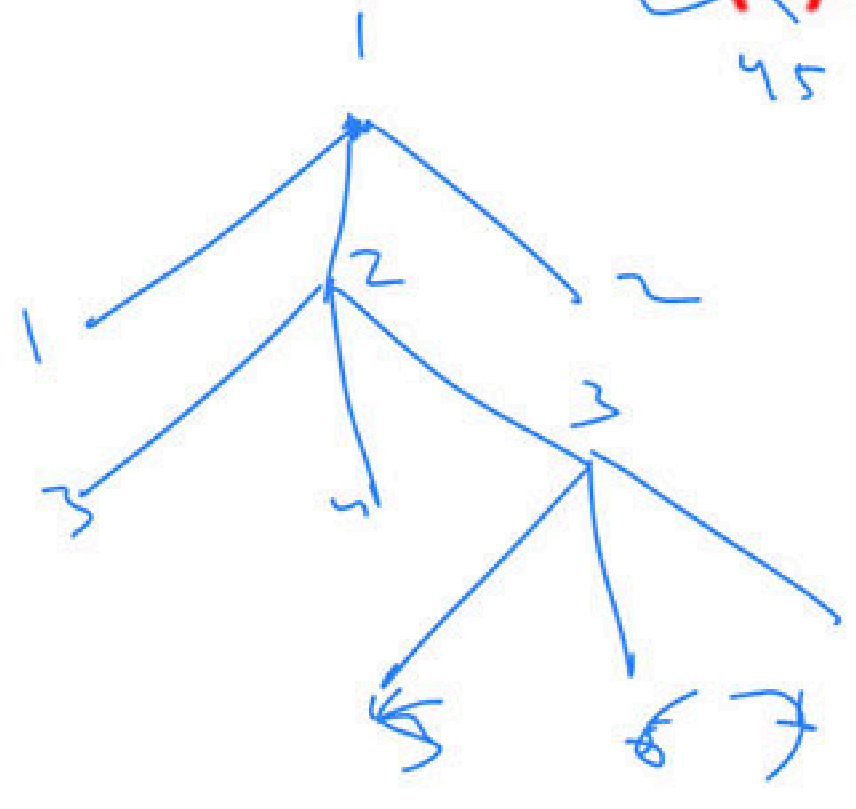
# Break

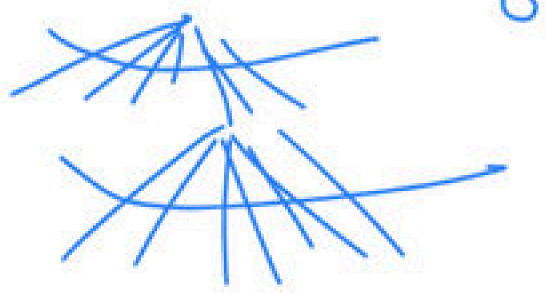
**Q** A complete n-ary tree is a tree in which ~~each node has~~ n children or no children. Let I be the number of internal nodes and L be the number of leaves in a complete n-ary tree. If L = 41, and I = 10, what is the value of n? **(GATE - 2007) (2 Marks)**

**(A)** 3 **(B)** 4 **(C)** 5 **(D)** 6

14 33 45 8







$L = I + 1$

$$\boxed{L = 2I + 1}$$

$$\boxed{L = (n-1)I + 1}$$

$L = (n-1)I + 1$

$\cancel{41} = (n-1)10 + \cancel{1}$

$40 = 4n$

$\cancel{10}$ $n = 5$

**Q** The number of leaf nodes in a rooted tree of n nodes, with each node having 0 or 3 children is: **(GATE - 2002) (2 Marks)**

**a)** n/2 **b)** (n−1)/3 **c)** (n−1)/2 **d)** (2n+1)/3

2 25 12 60

$L = (n-1) I + 1$

$L = (3-1) I + 1$

$L = 2I + 1$

$n = I + L$

$I = n - L$

$L = 2(n-L) + 1$

$L = 2n - 2L + 1$

$L = \frac{2n+1}{3}$

**Q** A complete n-ary tree is one in which every node has 0 or n sons. If x is the number of internal nodes of a complete n-ary tree, the number of leaves in it is given by **(GATE - 1998) (2 Marks)**

**a)** $x(n-1)+1$ **b)** $xn-1$ **c)** $xn+1$ **d)** $x(n+1)$

24 2 1 3

**Q** In a complete k-ary tree, every internal node has exactly k children or no child. The number of leaves in such a tree with an internal node is:

**(A)** n.k

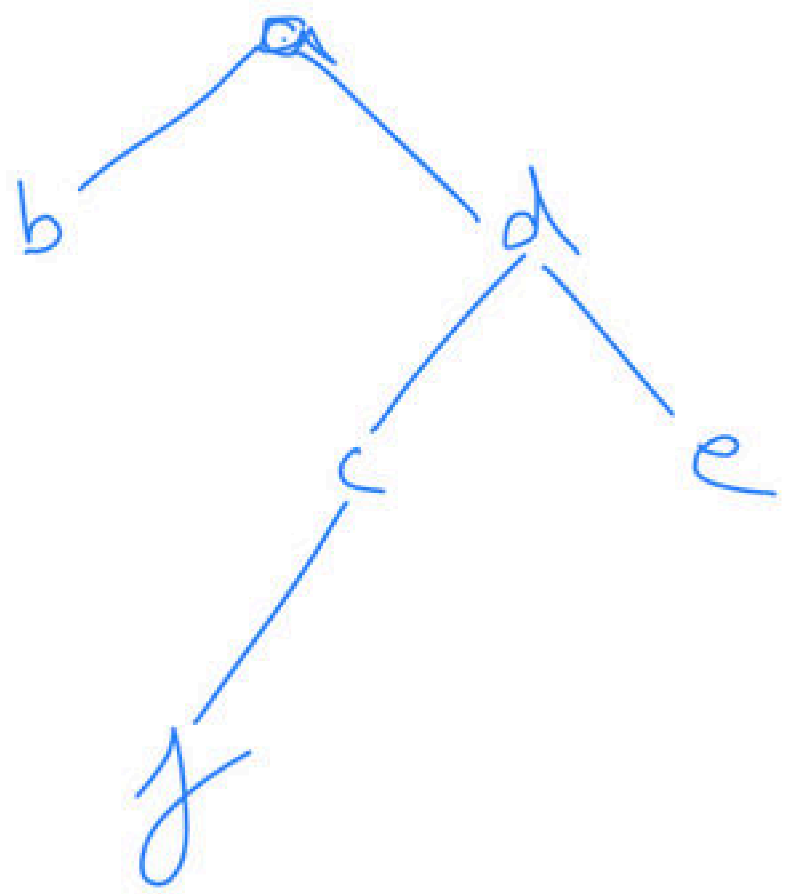
**(B)** (n – 1) k+ 1

**(C)** n (k – 1) + 1

**(D)** n (k – 1)

# Break

**Q** A binary tree T has 20 leaves. The number of nodes in T having two children is 19. **(GATE - 2015) (1 Marks)**



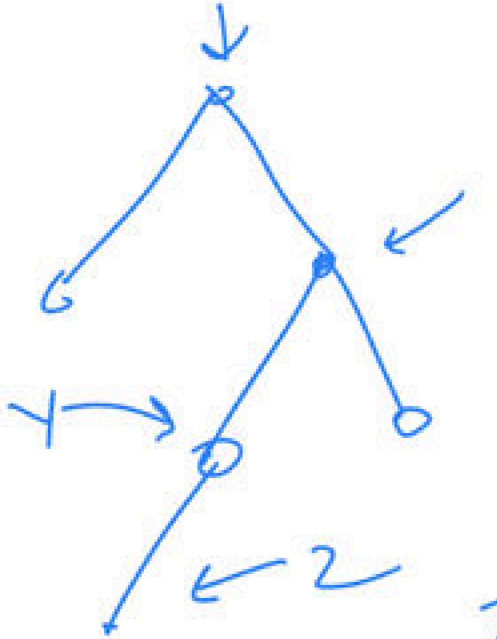
**Q** In a binary tree, the number of internal nodes of degree 1 is 5, and the number of internal nodes of degree 2 is 10. The number of leaf nodes in the binary tree is **(GATE - 2006) (1 Marks)**

$\boxed{n = 5 + 10 + z}$ — 1

**a)** 10 **b)** 11 **c)** 12 **d)** 15

$2n = 39 + z \cdot 2$

$2n = \frac{41+z}{2}$




$= \underset{3}{X} + \underset{2}{Y} + \underset{1}{Z}$

$5 + 10 + z = \frac{41+z}{2}$

$30 + 2z = 41 + z$

$z = 41 - 30$

$\boxed{z = 11}$

$\boxed{2(n-1) = 3 \times 10 - 1 + 5 \times 2 + 1 \times z}$ — 2

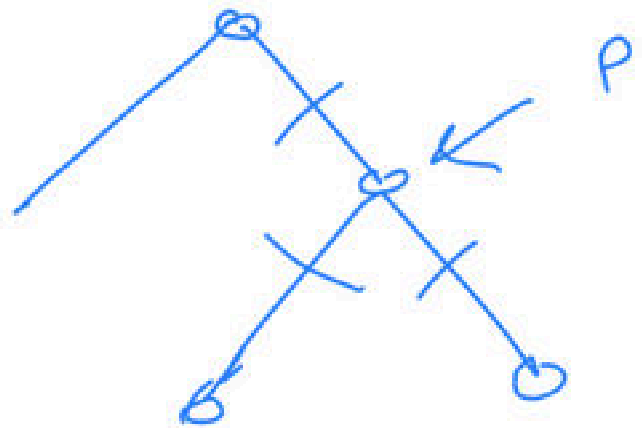
$= 30 - 1 + 10 + z$

$n \to n-1$

$= 2(n-1)$

$2(n-1) = 39 + z$

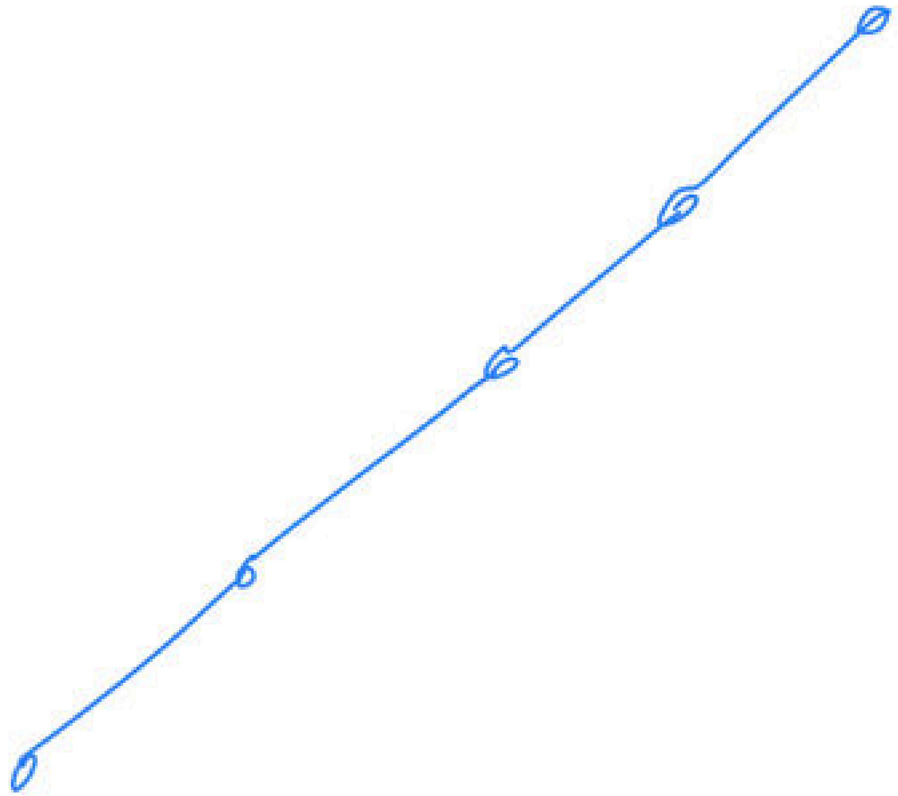
$2n - 2 = 39 + z$





1 4

$L = I + 1$



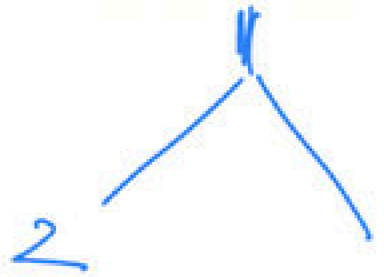
**Q** A binary tree T has n leaf nodes. The number of nodes of degree 2 in T is **(GATE-1995) (1 Marks)**

**a)** $\log_2 n$ **b)** $n-1$ **c)** $n$ **d)** $2^n$

# Break

**Q** Consider the following nested representation of binary trees: (X Y Z) indicates Y and Z are the left and right subtrees, respectively, of node X. Note that Y and Z may be NULL, or further nested. Which of the following represents a valid binary tree? **(GATE - 2000) (1 Marks)**

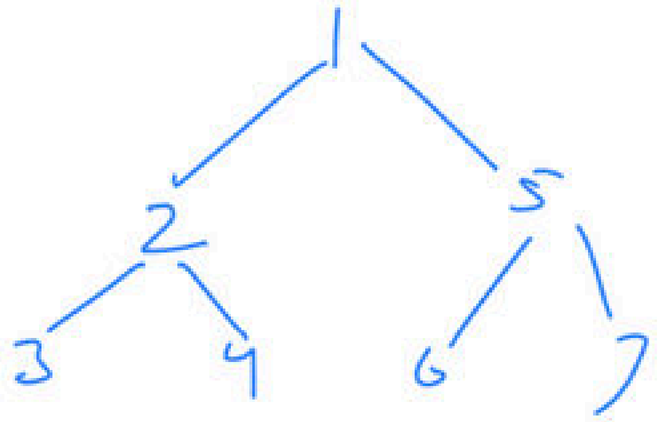
**a)** $\underbrace{1}_{X}\ \underbrace{2}_{Y}\ \underbrace{(4\ 5\ 6\ 7)}_{Z}$ — 9

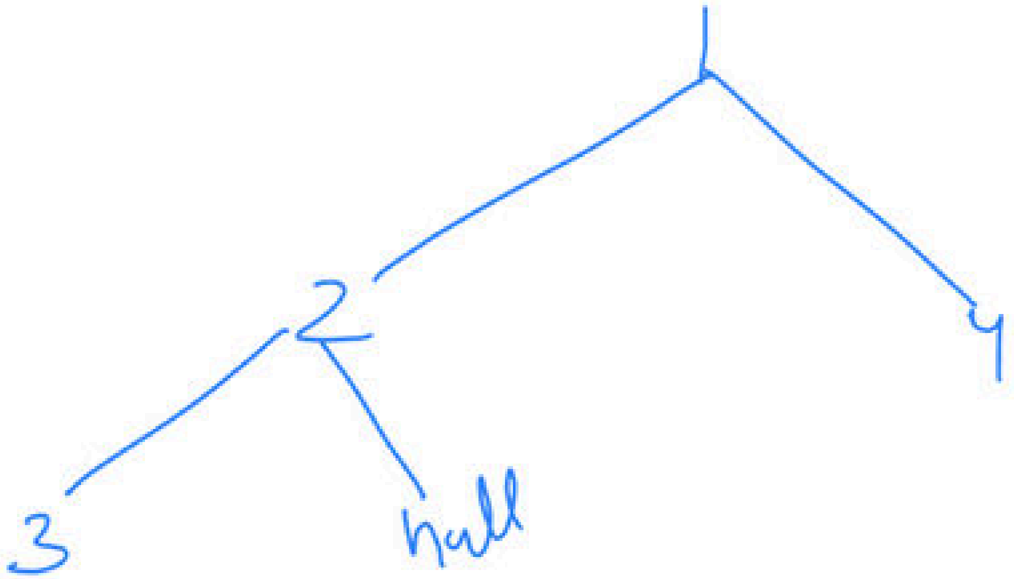


~~**b)** (1 (2 3 4) 5 6) 7)~~ — 7

**c)** $(\underbrace{1}_{X}\ \underbrace{(2\ 3\ 4)}_{Y}\ \underbrace{(5\ 6\ 7)}_{Z})$ — 68

**d)** (1 (2 3 NULL) (4 5)) — 15





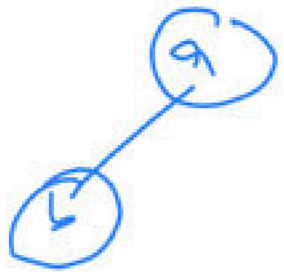
**Q** Which of the following statements is false? **(GATE - 1998) (1 Marks)**

**a)** A tree with a n nodes has (n–1) edges —13 T

**b)** A labeled rooted binary tree can be uniquely constructed given its post order and preorder traversal results. —56 F

**c)** A complete binary tree with n internal nodes has (n+1) leaves. —26 F

**d)** The maximum number of nodes in a binary tree of height h is $2^{h+1}-1$ —5 T



**Q.11** Let $H$ be a binary min-heap consisting of $n$ elements implemented as an array. What is the worst case time complexity of an optimal algorithm to find the maximum element in $H$? **(GATE-2021)**

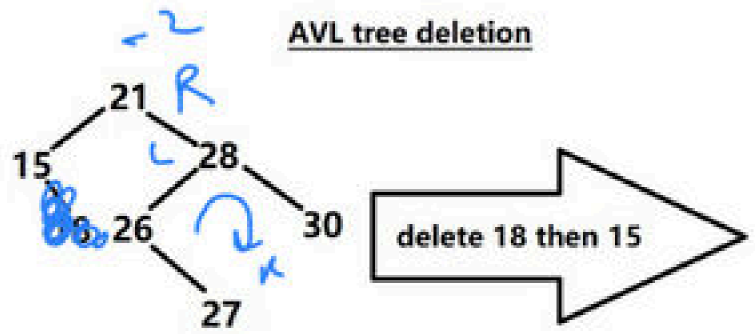
(a) $\Theta(\log n)$ (b) $\Theta(1)$

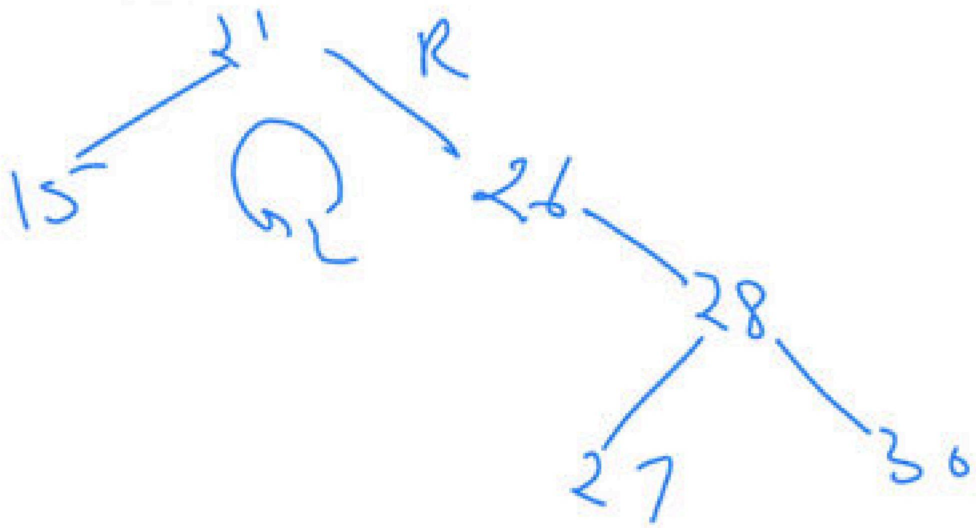
(c) $\Theta(n \log n)$ (d) $\Theta(n)$

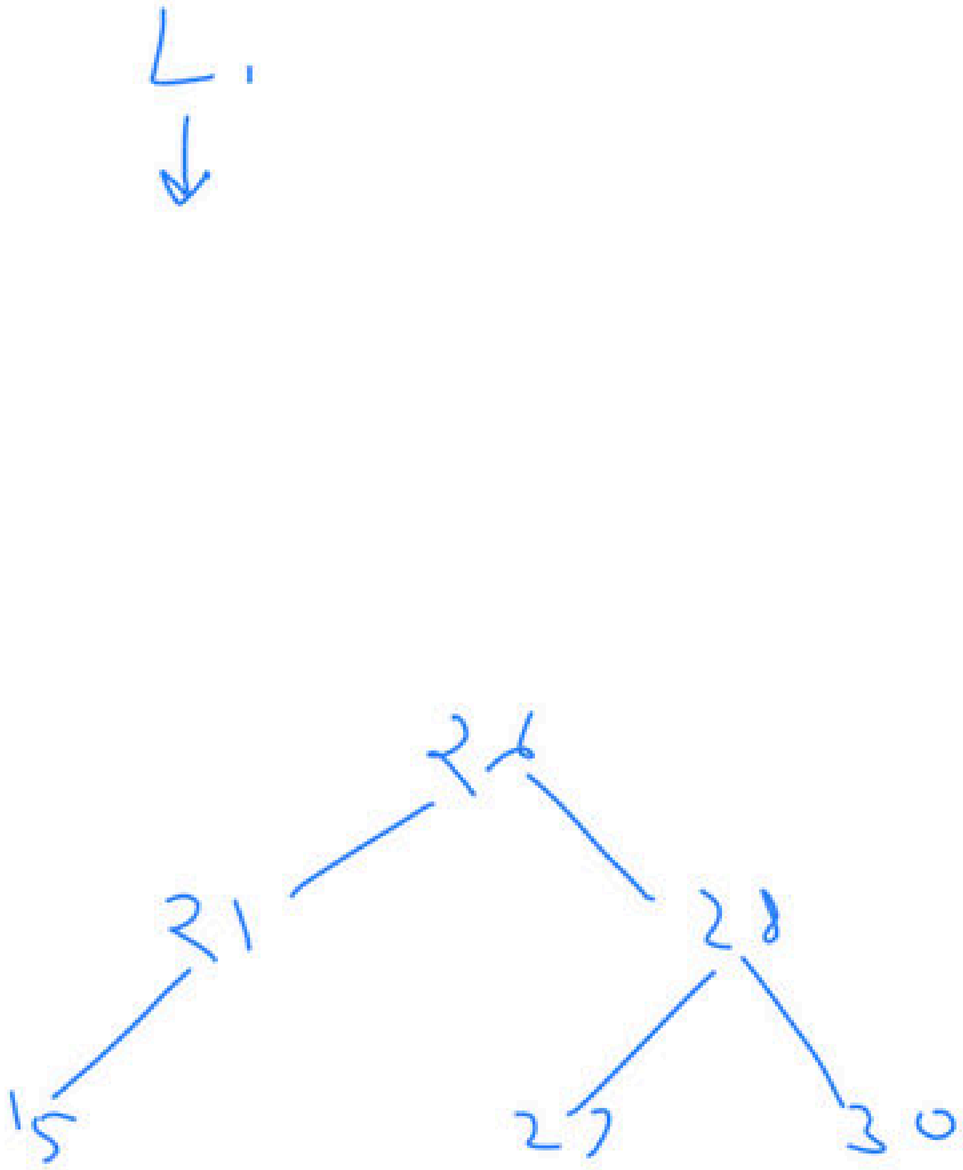
unacademy

▲ 1 • Asked by Arindam

isko ekbar dekh lijiye, atak rha hoon..

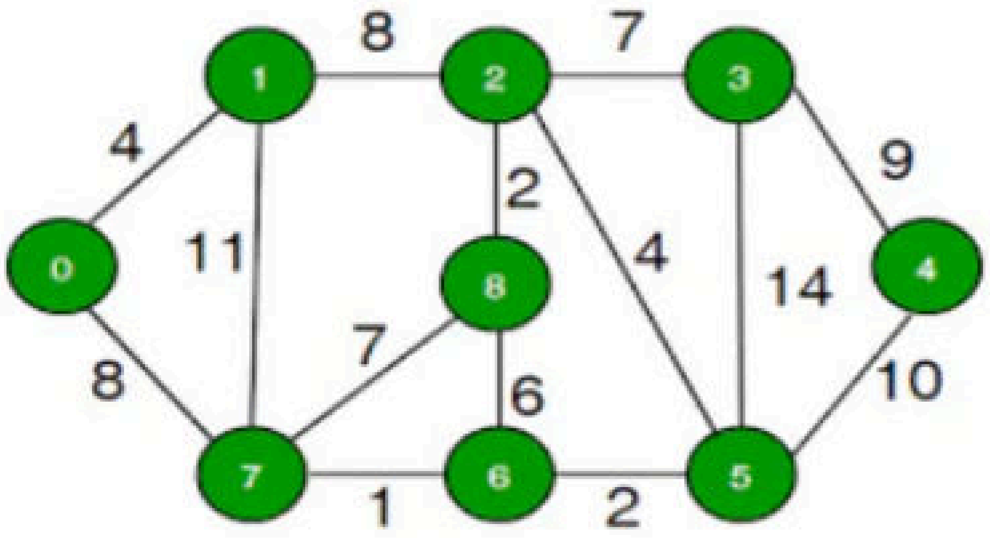


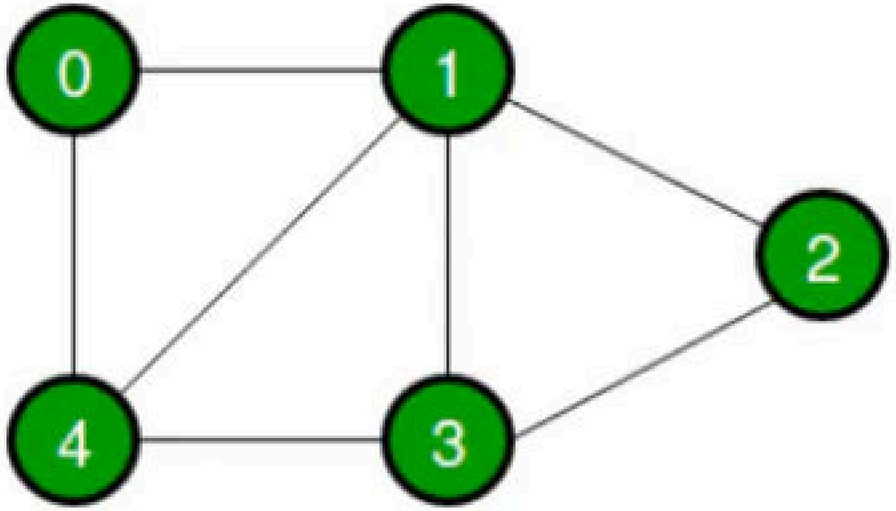



# Graph

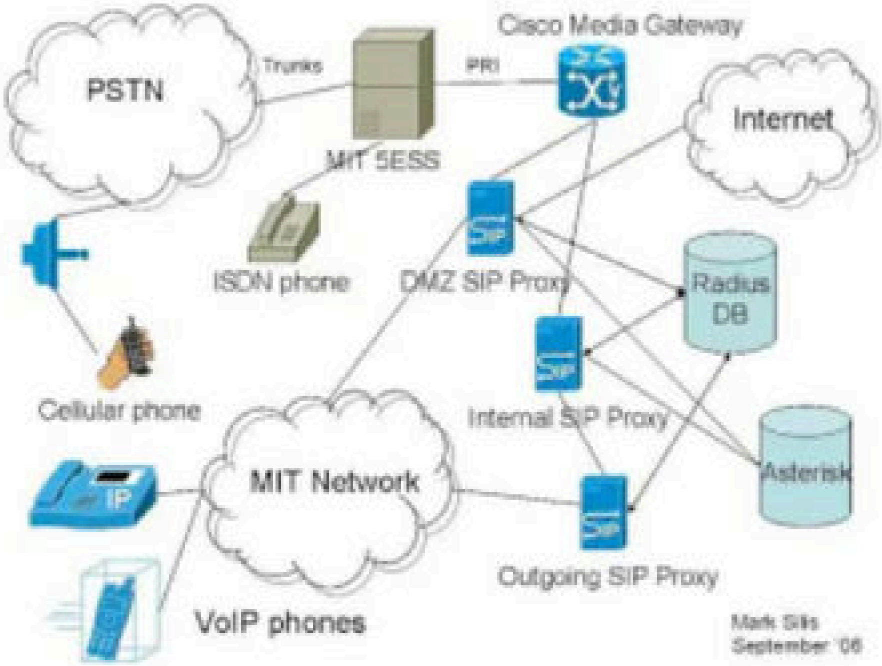
- Graph is a data structure that consists of following two components:
  - A finite set of vertices also called as nodes.
  - A finite set of ordered pair of the form (u, v) called as edge. The pair is ordered because (u, v) is not same as (v, u) in case of a directed graph(di-graph).
  - The pair of 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.

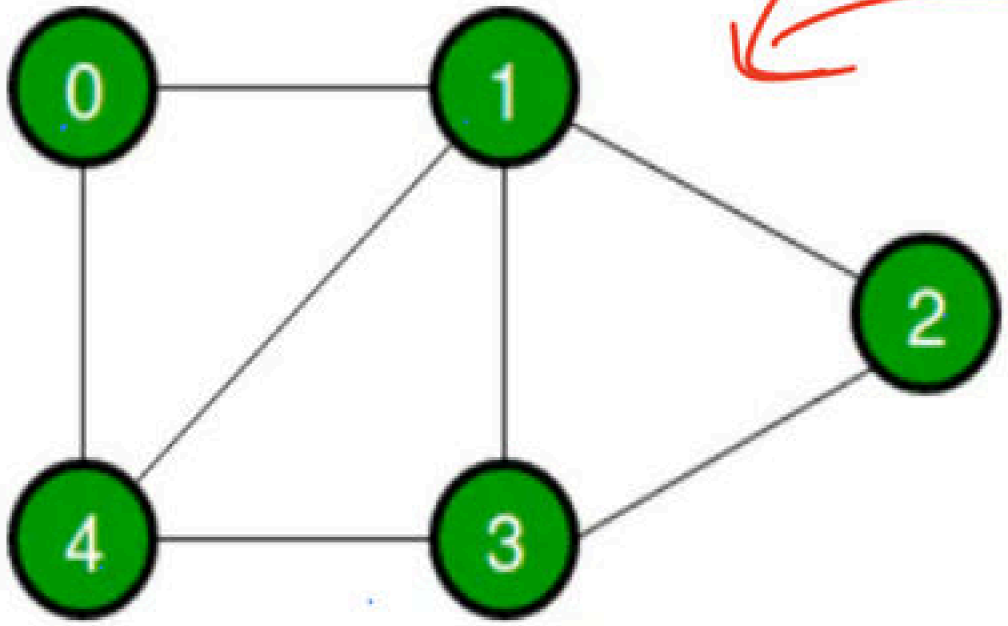
- **Adjacency Matrix:** Adjacency Matrix is a 2D array of size V x V where V is the number of vertices in a graph. Let the 2D array be adj[][], a slot adj[i][j] = 1 indicates that there is an edge from vertex i to vertex j.

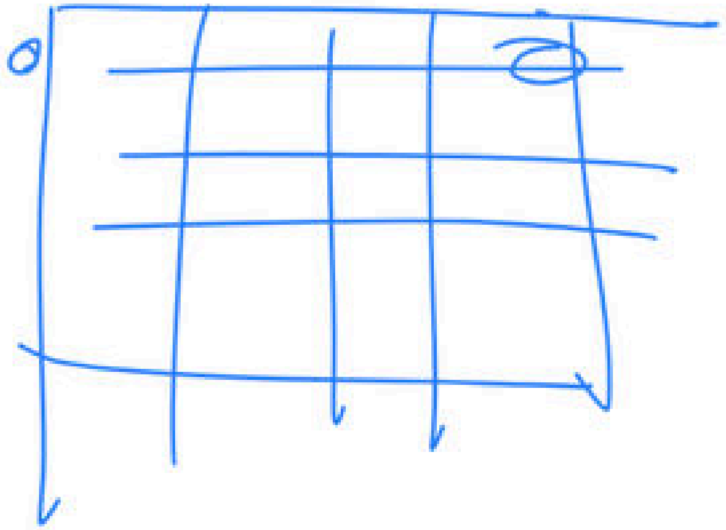
7500 X 7500

- Adjacency matrix for undirected graph is always symmetric.

5, 62, 5700

- 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.





| | 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 1 |
| 2 | 0 | 1 | 0 | 1 | 0 |
| 3 | 0 | 1 | 1 | 0 | 1 |
| 4 | 1 | 1 | 0 | 1 | 0 |

n

n x n

- **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.

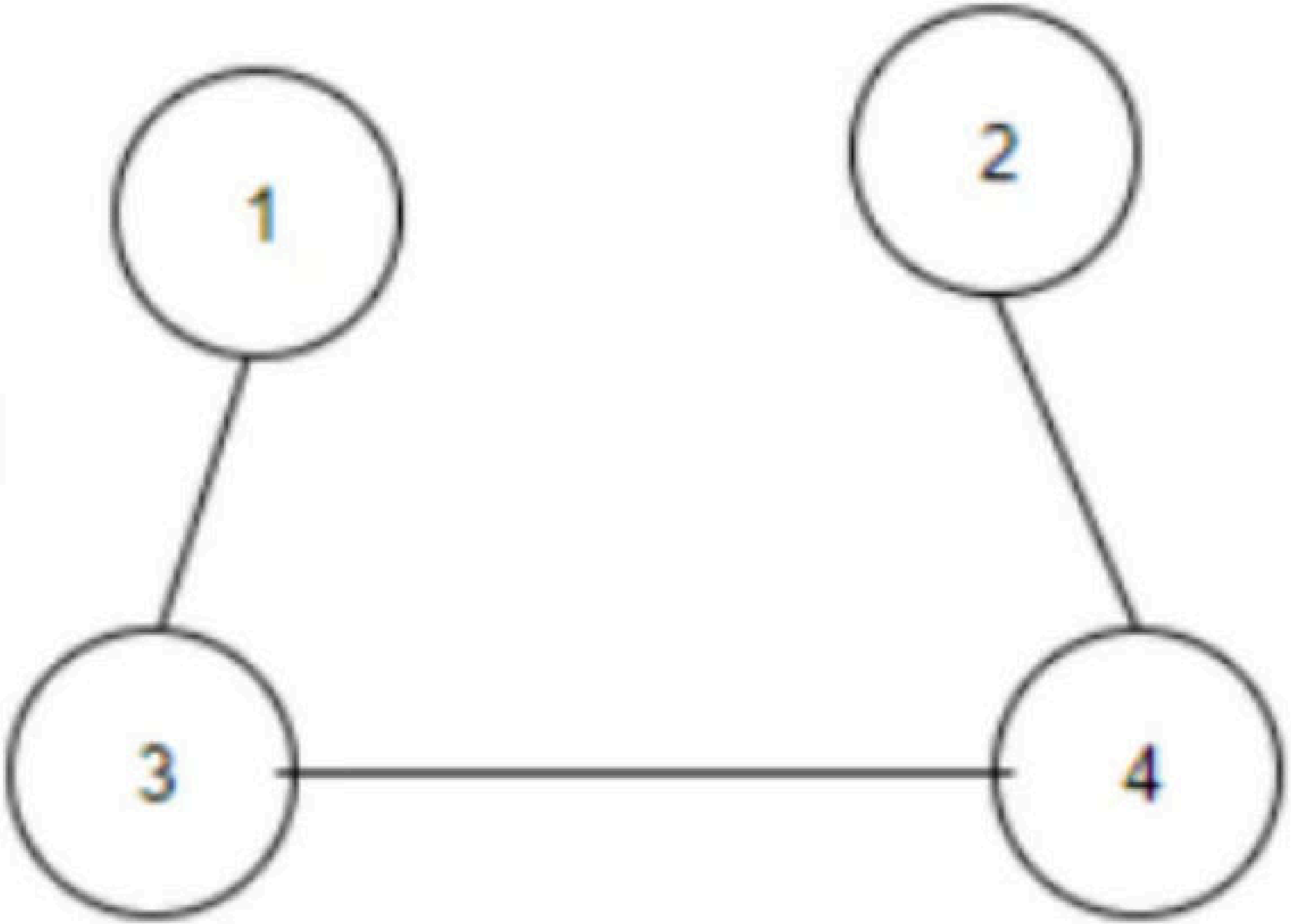
Q. What would be the number of zeros in the adjacency matrix of the given graph? **[Asked in AMCAT 2016 ]**

~~a. 10~~ ← 12

b. 6 ← 13

c. 16 ← 7

d. 0 ← 13

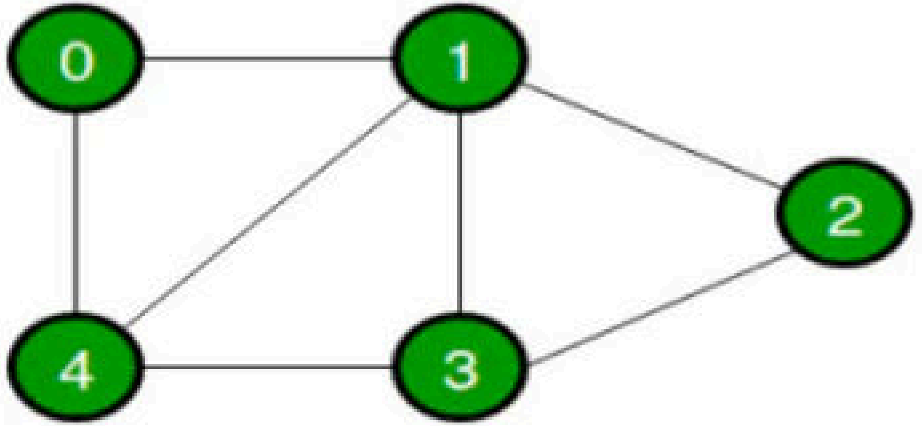


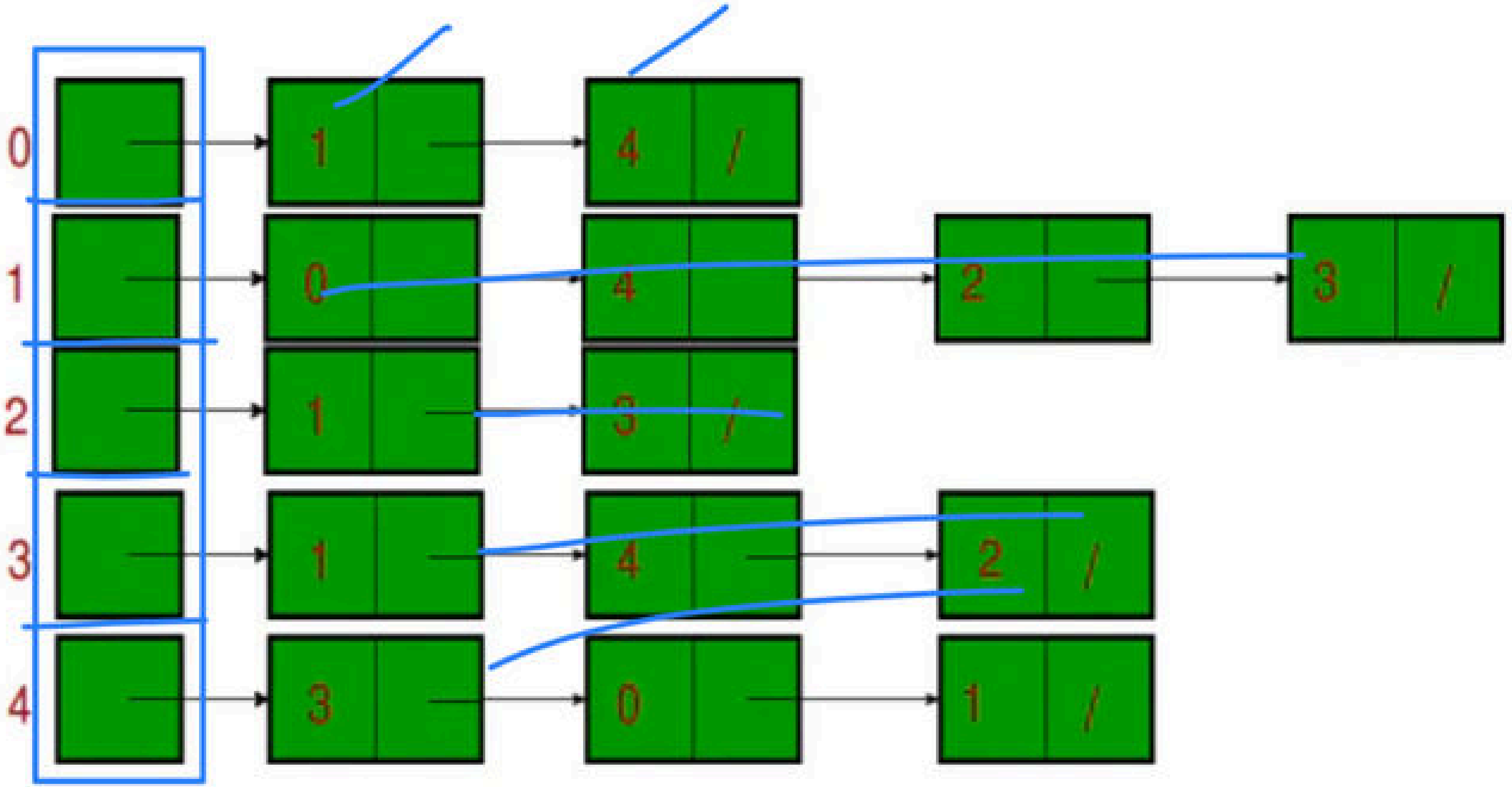

| | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | · | | X | · |
| 2 | · | · | · | X |
| 3 | X | · | · | X |
| 4 | · | X | X | |

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

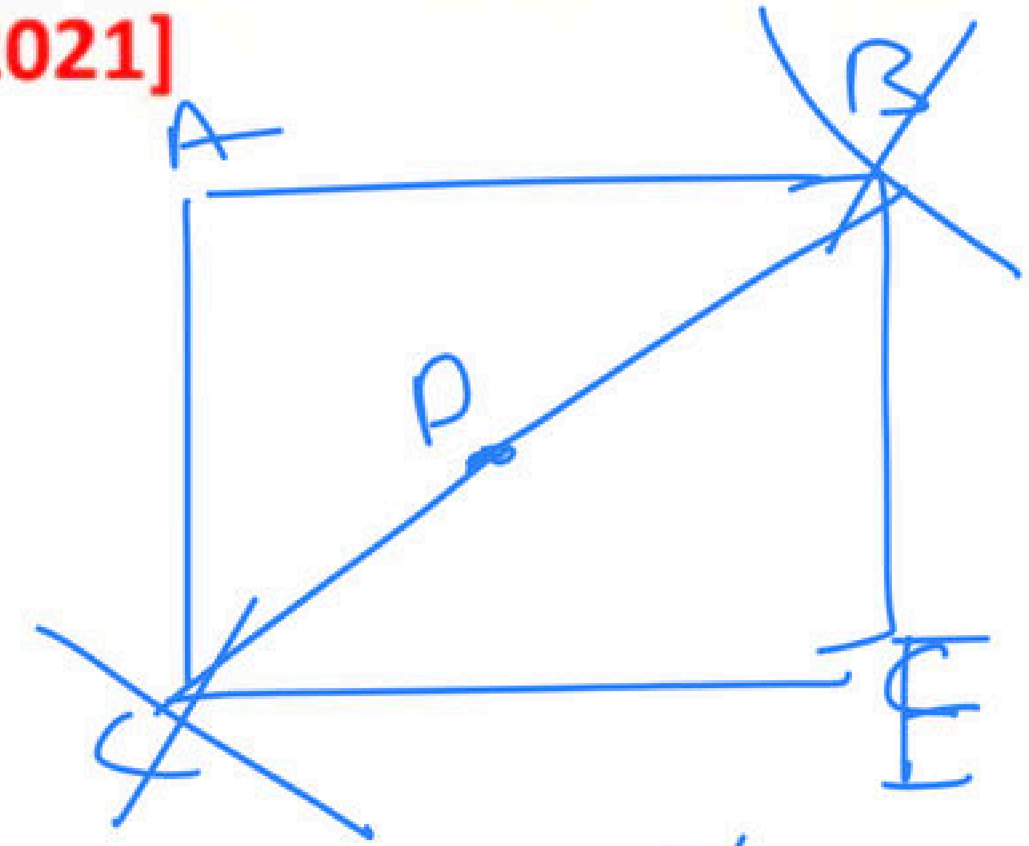
Q. Which of the following statements for a simple graph is correct? **[Asked in Capgemini ]**

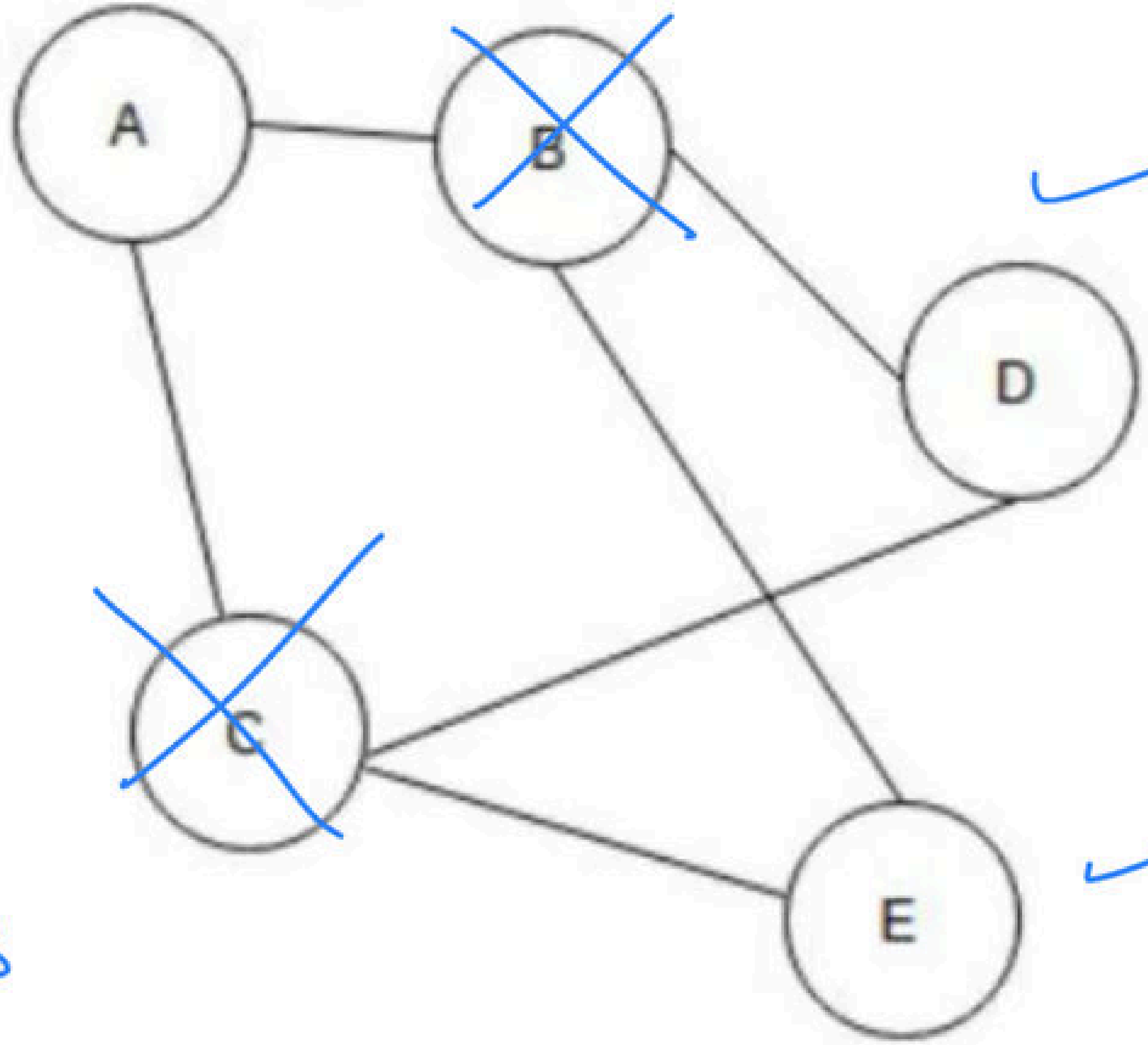
a) Every path is a trail
b) Every trail is a path
c) Every trail is a path as well as every path is a trail
d) Path and trail have no relation

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~~ 21

~~b) G is not a connected graph~~ 19

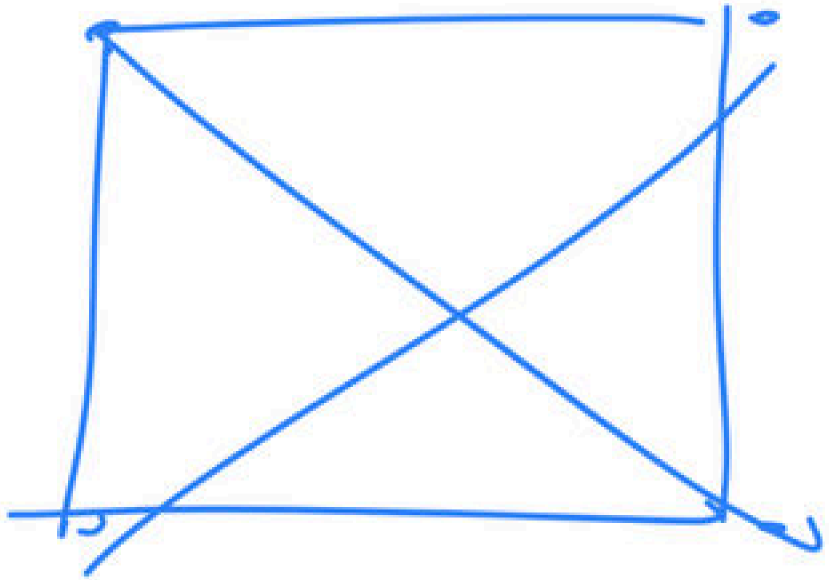
c) The vertex connectivity of the graph is 2 43

~~d) The edge connectivity of the graph is 1~~ 12

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 — 27

b) (n*(n-1))/2 — 67

c) n — 8

d) Information given is insufficient — 1

$$\frac{n(n-1)}{2}$$

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 ]**

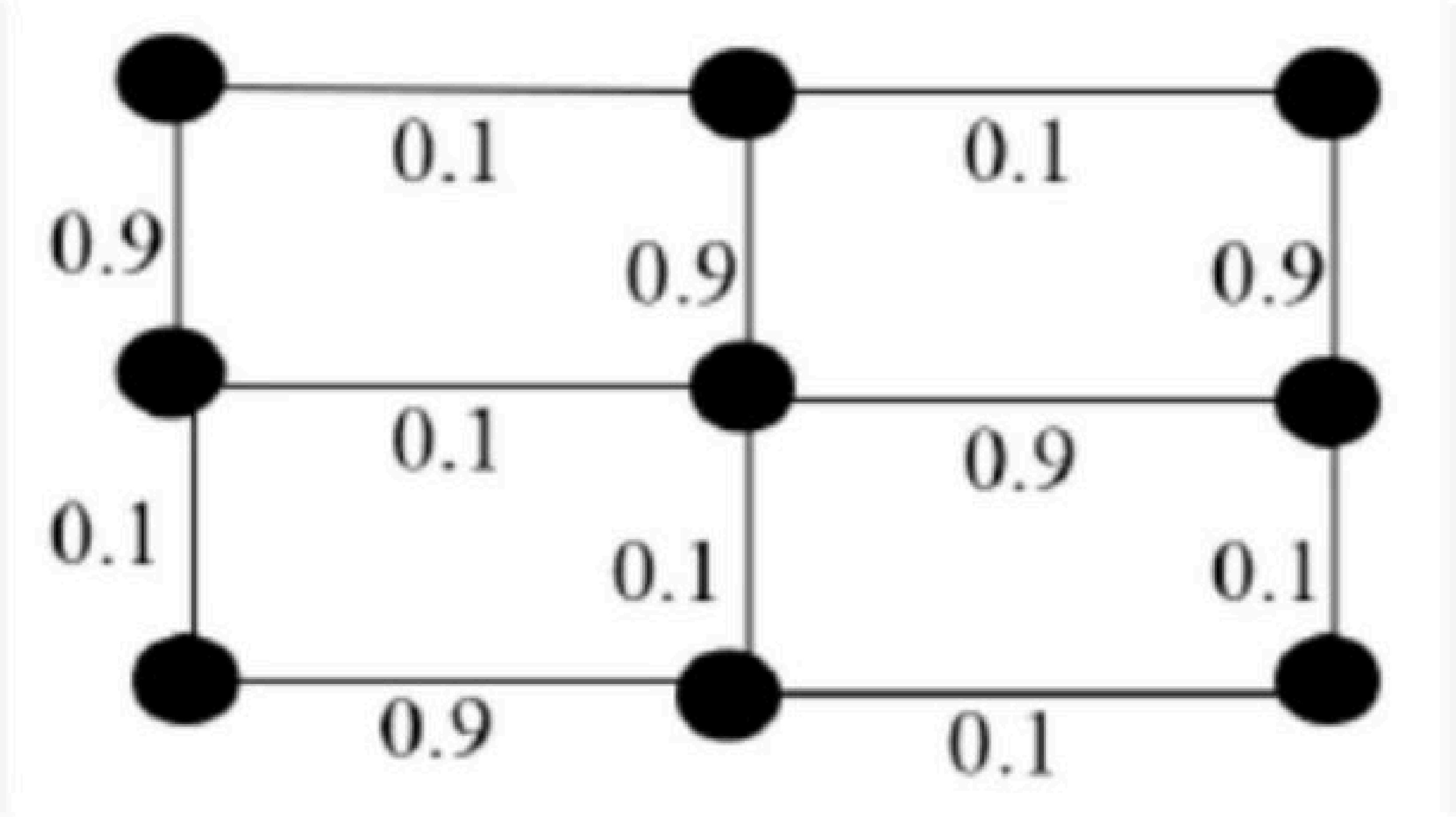
a. P Only
b. Q Only
c. Both P and Q
d. Neither P nor Q

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.

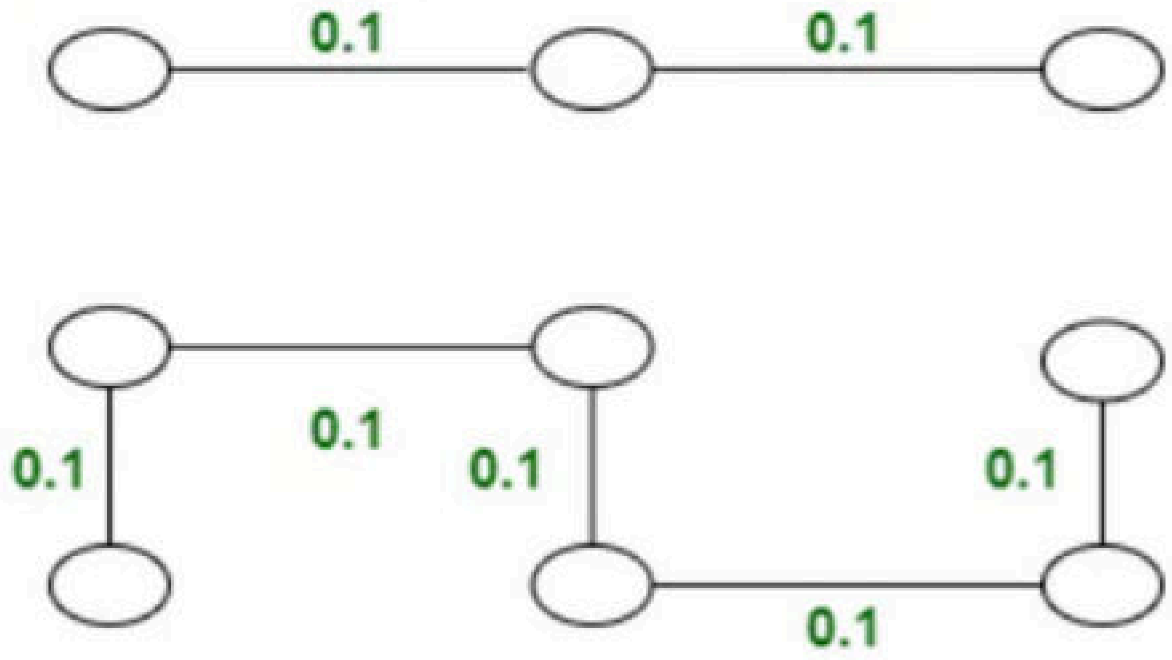
Q. Consider the following un-directed graph with edge weights as shown: The number of minimum-weight spanning trees of the graph is ___________. **[Asked in Cognizant 2021]**



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.

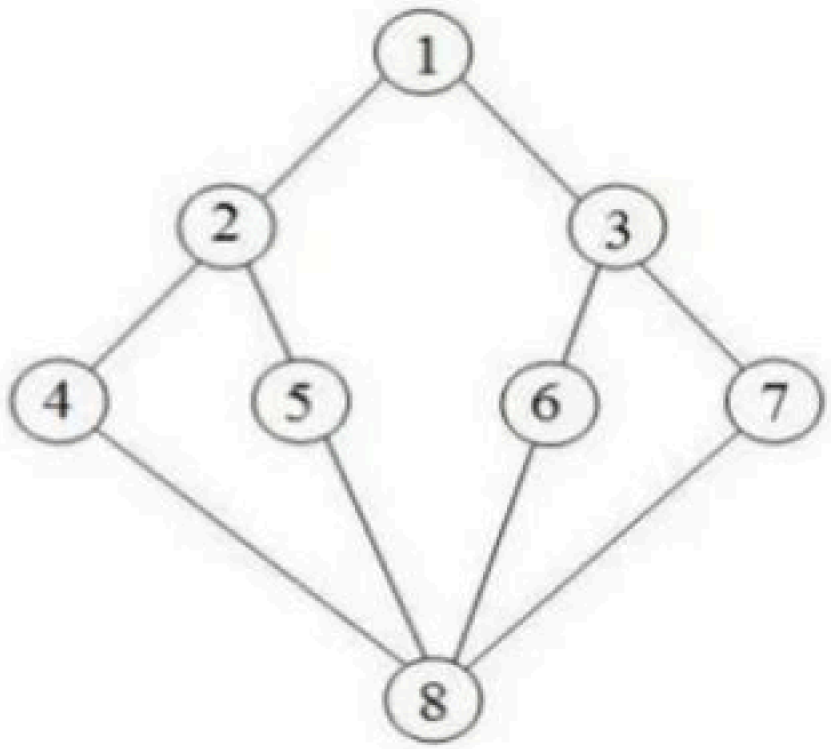
Q. Given an undirected graph G with V vertices and E edges, the sum of the degrees of all vertices is **[Asked in L&T Infotech (LTI) 2019]**

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.



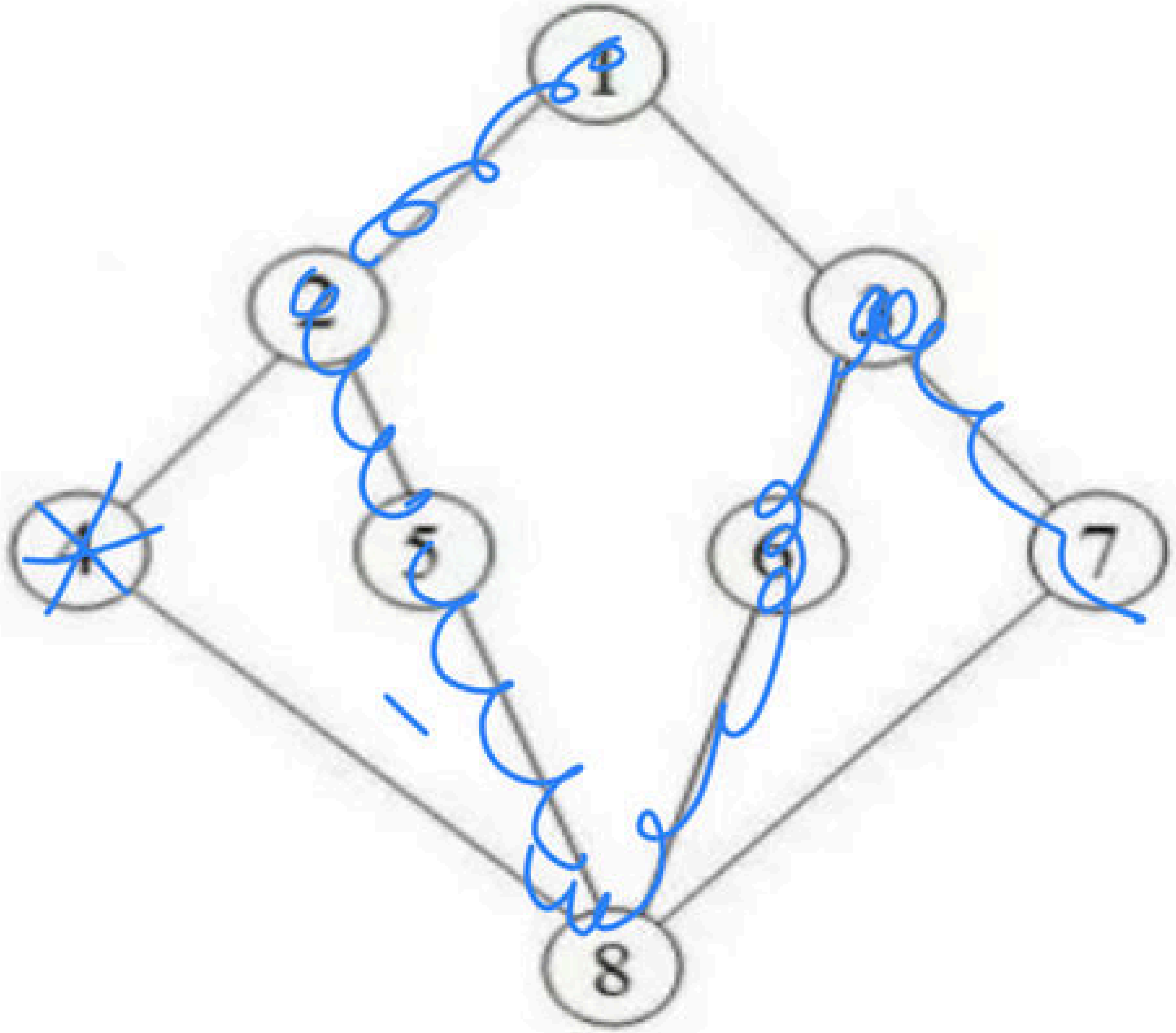
**Q** Which of the following are valid and invalid DFS traversal sequence

**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

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

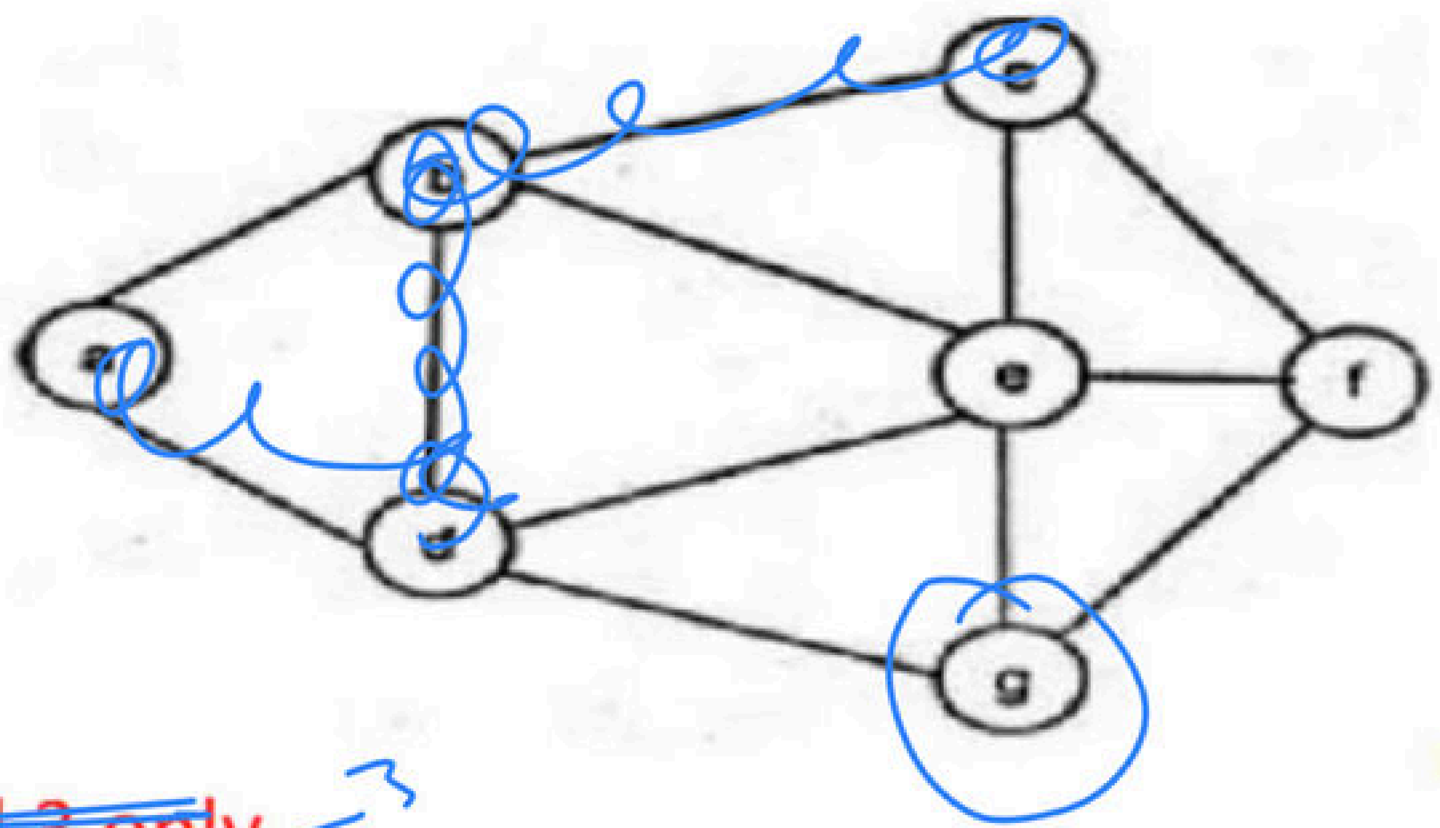
the total cost of executing lines 4–7 of DFS-VISIT is $\Theta(E)$. The running time of DFS is therefore $\Theta(V + E)$.

# 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 b e f c d g~~

~~**(A)** 1 and 3 only~~ —3

**(B)** 2 and 3 only —71

~~**(C)** 2, 3 and 4 only~~ —15

~~**(D)** 1, 2, and~~ 3 —11

**Q** Consider the following graph

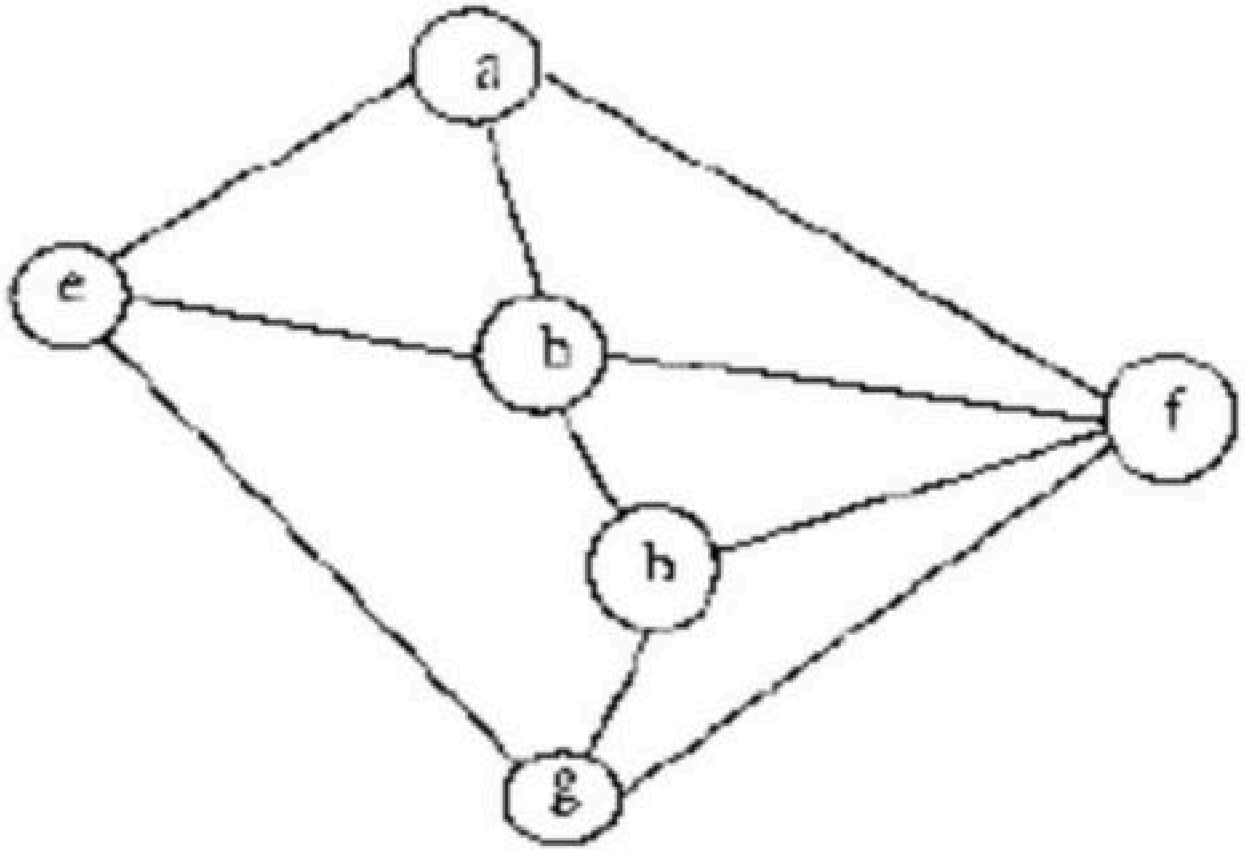
Among the following sequences

**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. **(GATE-2021)**

Let $T$ be a DFS tree obtained by doing DFS in a connected undirected graph $G$.

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 $u$.

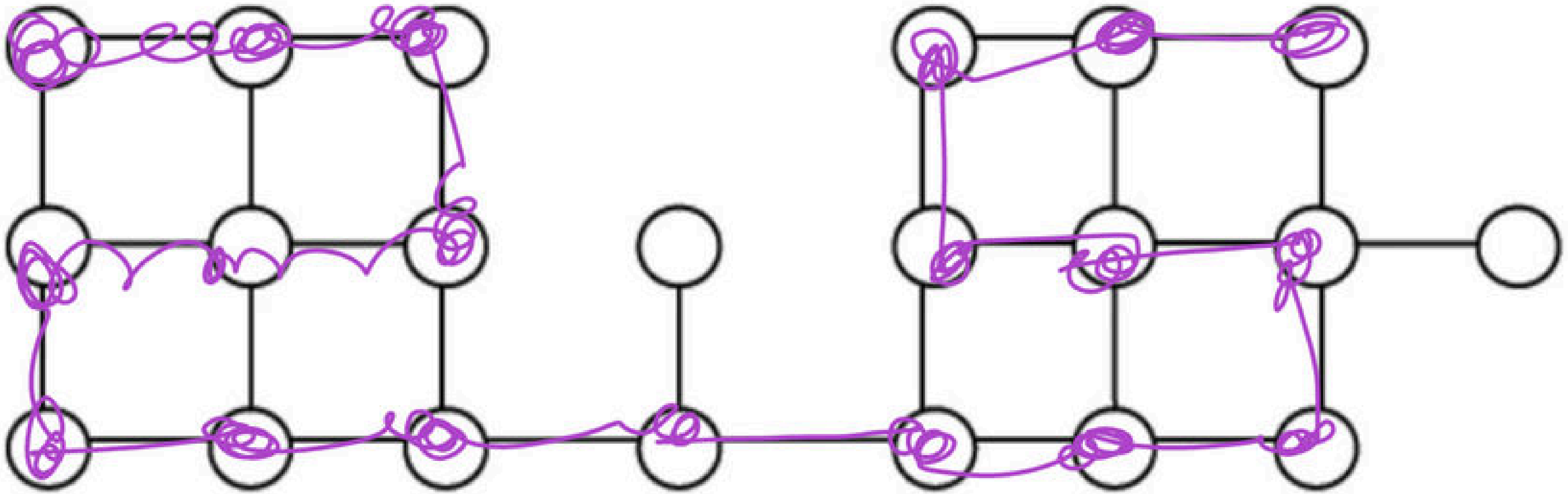
**Q** Suppose depth first search is executed on the graph below starting at some unknown vertex. Assume that a recursive call to visit a vertex is made only after first checking that the vertex has not been visited earlier. Then the maximum possible recursion depth (including the initial call) is __________.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

13 → a) 17

26 → b) 18

34 → ~~c) 19~~

28 → d) 20



**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)**

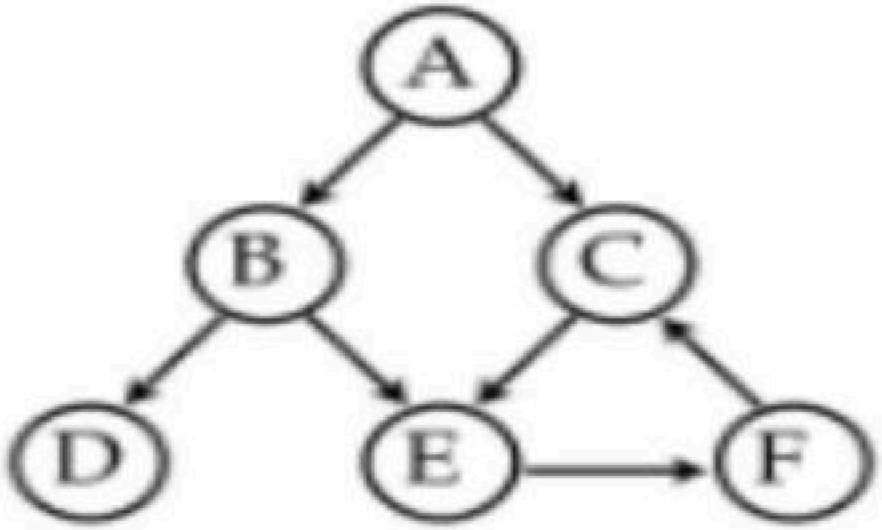
**(A)** There must exist a vertex w adjacent to both u and n in G

**(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)**

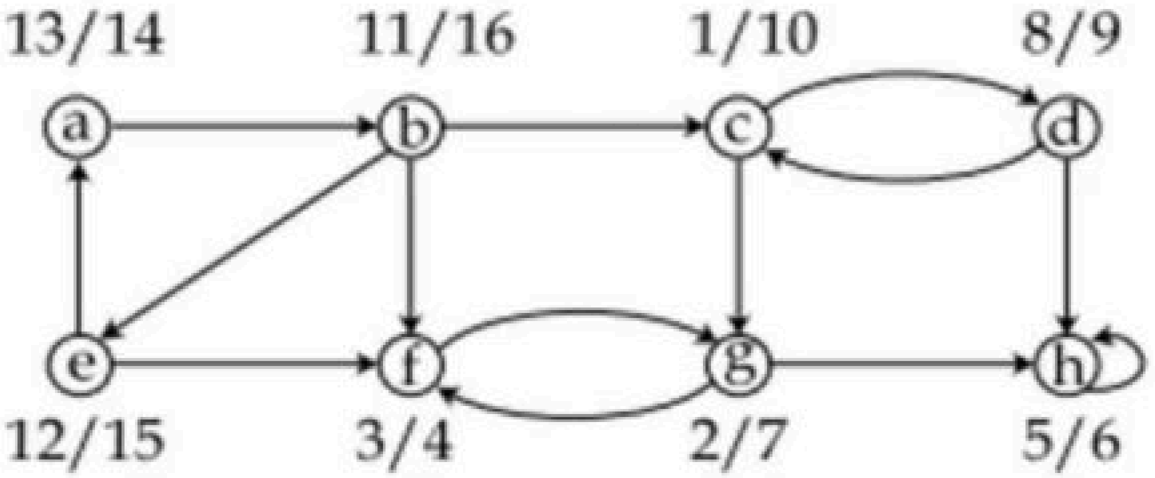
**(A)** {u,v} must be an edge in G, and u is a descendant of v in T

**(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

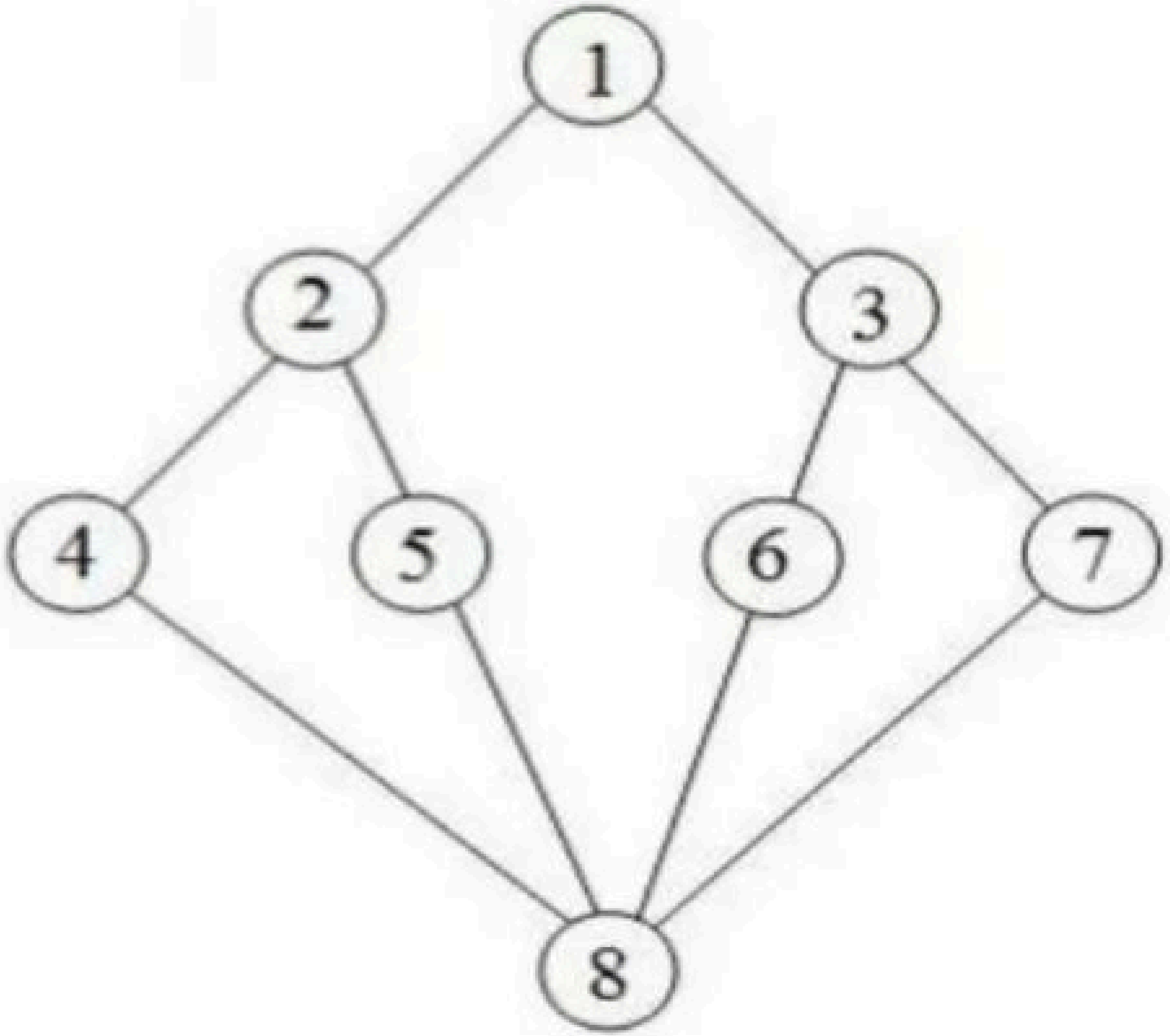
**Q** Which of the following are valid and invalid BFS traversal sequence

**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



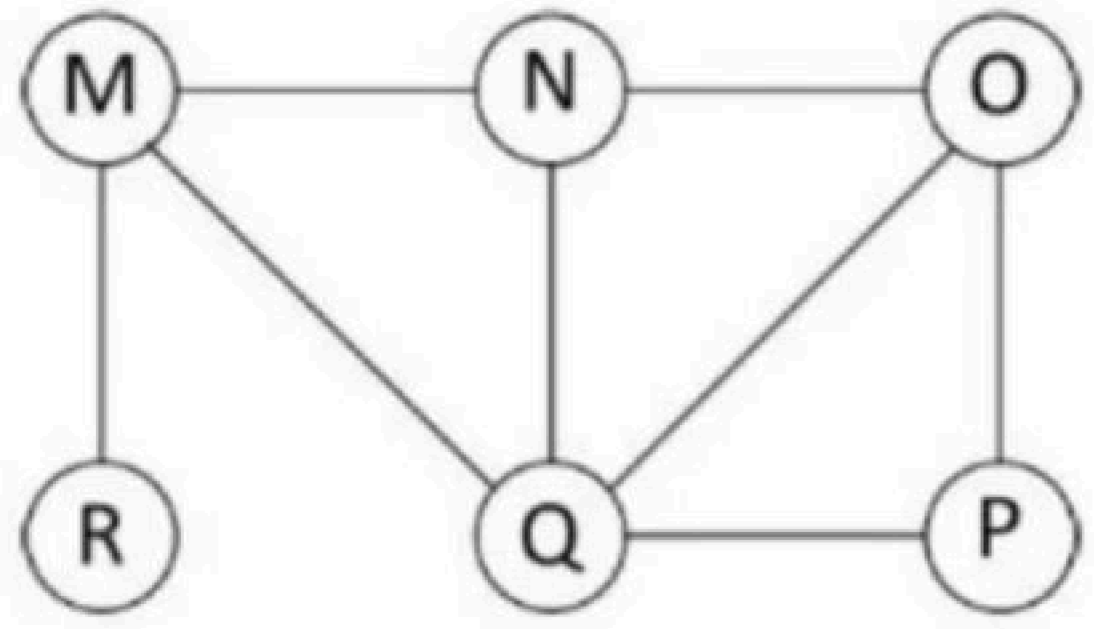
```
BFS(v)
{
        visited(v) = 1
        insert[V,Q]
        While(Q != Phi)
        {
                u = Delete(Q);
                for all x adjacent to u
                {
                        if (x is not visited)
                                {
                                        visited(x)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

```
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 $G$.

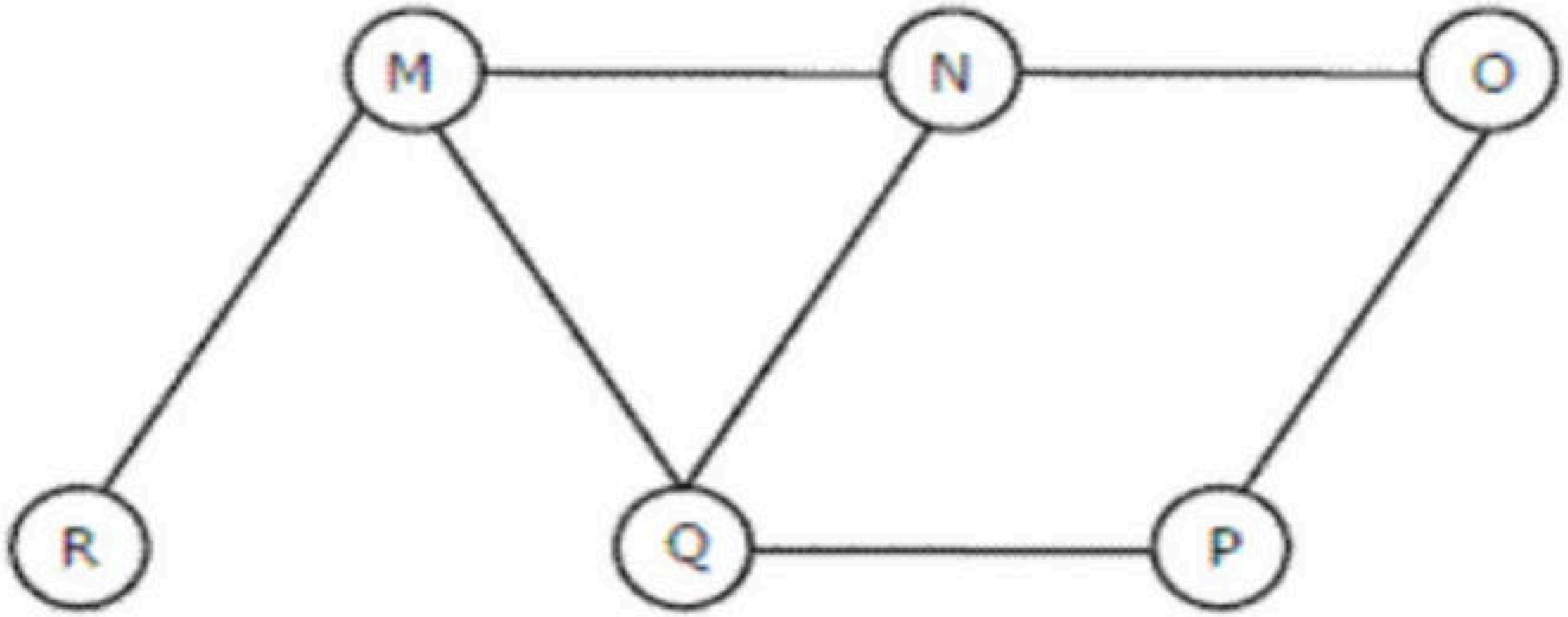
**Q** Breath First Search (BFS) has been implemented using queue data structure. Which one of the following is a possible order of visiting the nodes in the graph above? **(Gate-2017) (1 Marks)**

**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 ________ **(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 $x \in V$, 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

**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)