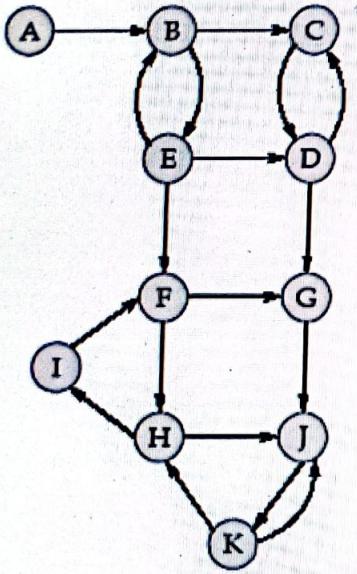
Question No : 1 / 15

Multi Choice Type Question

Consider the following graph:



The number of strongly connected components of the graph are

Answer here

- 4
- 5
- 3
- 2

Question No : 2 / 15

Multi Choice Type Question

What will be the output of the following code?

```
public class PrintEdges {
    public static void main(String[] args) {
        int[][] edges = {{0,1},{1,2},{2,0}};
        for(int i=0;i<edges.length;i++) {
            System.out.println(edges[i][0] + " -> " +
        }
    }
}
```

Answer here

- 1->0, 2->1, 0->2
- **0->1, 1->2, 2->0**
- Error
- 0->2, 2->1, 1->0

Marks : 1 Negative Marks : 0

Clear

Multi Choice Type Question

Suppose we are given a digraph and asked to determine if it is strongly connected. However, instead of using an algorithm for strongly connected components, we decide to check for the existence of a path from every vertex to every other vertex. What is the time complexity of this approach?

Answer here

- O(n + m)
- O(n^2 + m)
- **O(n * (n + m))**
- O(n^3)

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

Question No : 4 / 15

Multi Choice Type Question

In a digraph, a strongly connected component is a subgraph where there is a path from every vertex to every other vertex. Which of the following algorithms is most efficient for finding strongly connected components in a digraph with a large number of vertices and edges?

Answer here

- (●) Tarjan's algorithm
- ( ) Kosaraju's algorithm
- ( ) Dijkstra's algorithm
- ( ) Bellman-Ford algorithm

Answered 15/15
Bookmarked 0/15
Skipped 0/15
Not Viewed 0/15
ved in Server 15/15

Clear

Question No : 5 / 15

Multi Choice Type Question

In the context of strongly connected components, what is the relationship between a vertex and its low-link value in Tarjan's algorithm?

Answer here

- The low-link value is the finishing time of the vertex
- The low-link value is the largest index reachable from the vertex
- The low-link value is the discovery time of the vertex
- **The low-link value is the smallest index reachable from the vertex**

Marks : 1 Negative Marks : 0

Clear

Question No : 6 / 15

Multi Choice Type Question

Which of the following statements is true about strongly connected components in a digraph?

Answer here

- (●) The strongly connected components of a digraph are disjoint
- ( ) A digraph with no cycles cannot have strongly connected components
- ( ) A digraph with a single strongly connected component is a tree
- ( ) Every strongly connected component has exactly one vertex

Multi Choice Type Question

What will be the output of the following code?

```
class StronglyConnectedGraph {
    public static void main(String[] args) {
       int[][] graph = {
    {0, 1, 0, 0, 0},
    {0, 0, 1, 0, 0},
    {0, 0, 0, 1, 0},
    {0, 0, 0, 0, 1},
    {0, 0, 1, 0, 0}
};

        if (isStronglyConnected(graph)) {
            System.out.println("The graph IS strongl
        } else {
            System.out.println("The graph is NOT str
        }
    }

    public static boolean isStronglyConnected(int[][
        int n = graph.length;
        for (int start = 0; start < n; start++) {
            boolean[] visited = new boolean[n];
            dfs(graph, start, visited);
            for (boolean v : visited) {
                if (!v) return false;
            }
        }
        return true;
    }

```

- ○ Runtime error
- ○ Compilation error
- ● The graph is NOT strongly connected.
- ○ The graph IS strongly connected.

Marks : 1 Negative Marks : 0

Clear

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

nswered 15/15
okmarked 0/15
kipped 0/15
t Viewed 0/15
d in Server 15/15
w More

Question No : 8 / 15

Multi Choice Type Question

Consider a digraph with n vertices, what is the time complexity of Tarjan's algorithm for finding strongly connected components?

Answer here

- O(n^3)
- O(n!)
- O(n^2)
- O(n + m)

Marks : 1 Negative Marks : 0

Clear

Question No : 9 / 15

Multi Choice Type Question

A digraph is said to be strongly connected if it has exactly one strongly connected component. Consider a digraph with vertices {1, 2, 3, 4} and edges {(1, 2), (2, 3), (3, 1), (3, 4)}. Is this digraph strongly connected?

Answer here

- Maybe
- Cannot be determined
- Yes
- **No** (selected)

Marks : 1 Negative Marks : 0

Clear

Question No : 10 / 15

Multi Choice Type Question

What is the output of the strongly connected components in the given digraph with multiple components?

```
1  import java.util.ArrayList;
2  import java.util.List;
3  class StronglyConnectedComponents {
4    public static void main(String[] args) {
5      int[][] graph = {
6        {0, 1, 0, 0, 0, 0},
7        {0, 0, 1, 0, 0, 0},
8        {0, 0, 0, 1, 0, 0},
9        {1, 0, 0, 0, 0, 0},
10       {0, 0, 0, 0, 0, 1},
11       {0, 0, 0, 0, 1, 0}
12     };
13     List<List<Integer>> scc = findStronglyConnectedC
14     System.out.println(scc);
15   }
16   public static List<List<Integer>> findStronglyConr
17     int n = graph.length;
18     boolean[] visited = new boolean[n];
19     List<List<Integer>> scc = new ArrayList<>();
20     for (int i = 0; i < n; i++) {
21       if (!visited[i]) {
22         List<Integer> component = new ArrayList<>();
23         dfs(graph, i, visited, component);
24         scc.add(component);
25       }
26     }
27     return scc;
28   }
```

Answer here

- [[0, 1, 2, 3], [4, 5]]
- [[0, 1, 2, 3], [4], [5]]
- [[0, 1, 2, 3], [4], [5]]
- [[0], [1], [2], [3], [4, 5]]

Marks : 1 Negative Marks : 0

Clear

Question No : 11 / 15

Multi Choice Type Question

What is the output of the strongly connected components in the given digraph?

```
import java.util.ArrayList;
import java.util.List;
public class StronglyConnectedComponents {
  public static void main(String[] args) {
    int[][] graph = {
      {0, 1, 0, 0},
      {0, 0, 1, 0},
      {0, 0, 0, 1},
      {1, 0, 0, 0}
    };
    List<List<Integer>> scc = findStronglyConnectedC
    System.out.println(scc);
  }
  public static List<List<Integer>> findStronglyConr
    int n = graph.length;
    boolean[] visited = new boolean[n];
    List<List<Integer>> scc = new ArrayList<>();
    for (int i = 0; i < n; i++) {
      if (!visited[i]) {
        List<Integer> component = new ArrayList<>();
        dfs(graph, i, visited, component);
        scc.add(component);
      }
    }
    return scc;
  }
  public static void dfs(int[][] graph, int vertex,
    visited[vertex] = true;
```

Answer here

- [[0, 1, 2], [3]]
- [[0], [1], [2], [3]]
- [[0, 1, 2, 3]]
- [[0], [1, 2, 3]]

Marks : 1 Negative Marks : 0

Clear

Question No : 12 / 15

Answer here

Multi Choice Type Question

A graph having an edge from each vertex to every other vertex is called a _____

- Weakly Connected
- Loosely Connected
- Strongly Connected
- Tightly Connected

Marks : 1 Negative Marks : 0

Clear

Question No : 13 / 15

Multi Choice Type Question

Let G = (V, E} be a directed graph where V Ts is the set of vertices and E the set of edges. Then which one of the following graphs has the same strongly connected component

Answer here

- (●) G2 (V, E2)t here Et = t(u, v}I(v, u) e E}
- ( ) Gt = (V4. E) where Vt is the set of vertices in G which are not isolated
- ( ) G3 - (V, Et) where E3 - t(•, v)I there is a path of length 2 from u to v in
- ( ) where E = ((u, v)I(u, v) e E}

Marks : 1 Negative Marks : 0

Clear

Question No : 14 / 15

Multi Choice Type Question

What will be the output of the following code?

```
class StronglyConnectedGraph {
    public static void main(String[] args) {
        int[][] graph = {
            {0, 1, 0, 0},
            {0, 0, 1, 0},
            {0, 0, 0, 1},
            {1, 0, 0, 0}
        };

        if (isStronglyConnected(graph)) {
            System.out.println("The graph IS strongl
        } else {
            System.out.println("The graph is NOT str
        }
    }

    public static boolean isStronglyConnected(int[][
        int n = graph.length;
        for (int start = 0; start < n; start++) {
            boolean[] visited = new boolean[n];
            dfs(graph, start, visited);
            for (boolean v : visited) {
                if (!v) return false;
            }
        }
        return true;
    }

    public static void dfs(int[][] graph, int verte
```

Answer here

- ○ The graph is NOT strongly connected.
- ○ Runtime error
- ○ Compilation error
- ● The graph IS strongly connected.

Marks : 1 Negative Marks : 0

Clear

Question No : 15 / 15

Multi Choice Type Question

In a digraph, if we have two strongly connected components, A and B, and there is an edge from a vertex in A to a vertex in B, what can we conclude about the strongly connected components?

Answer here

- ○ There will be a path from every vertex in B to every vertex in A
- ○ A and B will always merge into a single strongly connected component
- ● There will be no path from any vertex in B to any vertex in A
- ○ A and B will remain separate strongly connected components

Marks : 1 Negative Marks : 0

Clear