**1.(a)**

**BFS(startVert):**

**input: start Vertex of a graph**

**Output: DF-spanning tree**

**Algorithm:**

1. Create tree using Graph structure, create 2 stacks
2. Choose first vertex for spanning and push it to the first stack
3. While stack is not empty repeat the following:

* currentVert = stack.pop()
* if currentVert is not discovered yet: mark it as discovered, add each adjacent vertex to the second stack and create an adge between currentVert and each adjacent vert
* pop elements from the second stack to the first

**(b)**

**DFS(startVert):**

**input: start Vertex of a graph**

**Output: BF-spanning tree**

**Algorithm:**

1. Create tree using Graph structure, create queue
2. Choose start vertec and enqueue it
3. When queue is not empty repeat the following:

* currentVert = queue.dequeue()
* for each adjacent vertex of currentVert if it’s undiscovered:
  + mark it as discovering
  + add edge between currentVert and adjacent vertex
  + add adjacent vert to the queue
  + mark currentVert as discovered

**2.**

**ConnectedComponents(Graph):**

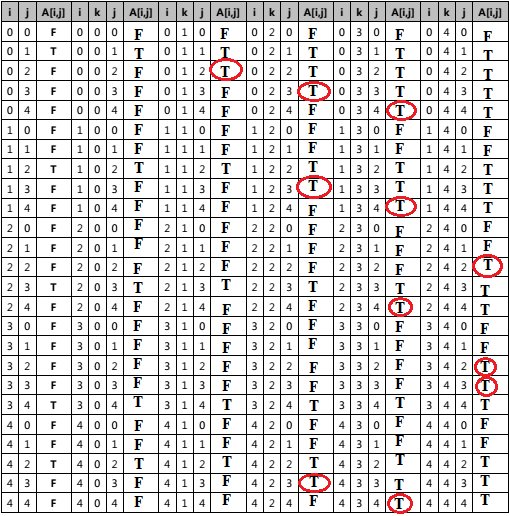
Input: Graph data structure

Output: list which contains sets of connected components

Algorithm:

* Pick any vertex of the graph, mark it as discovered and launch DFS-traversal (marking traversed vertices as discovered) starting from this vertex. Add each traversed vertex to the set
* Add set to the list of connected groups
* Do above steps for each remaining undiscovered vertex

**3.**



**4.** embed, other, refer, class, leaks, every, array

**5.** 15 26 37 50 53 57| 2 11 14 21 22 25

**6*.***

[35, 57, 53, 26, 50, 15, 22, 21, 25, 14, 11, 2] swap

[35, 2, 53, 26, 50, 15, 22, 21, 25, 14, 11, 57] swap

[35, 2, 11, 26, 50, 15, 22, 21, 25, 14, 53, 57]

[35, 2, 11, 26, 50, 15, 22, 21, 25, 14, 53, 57] swap

[35, 2, 11, 26, 14, 15, 22, 21, 25, 50, 53, 57]

[35, 2, 11, 26, 14, 15, 22, 21, 25, 50, 53, 57]

[35, 2, 11, 26, 14, 15, 22, 21, 25, 50, 53, 57]

[35, 2, 11, 26, 14, 15, 22, 21, 25, 50, 53, 57] swap

[25, 2, 11, 26, 14, 15, 22, 21, 35, 50, 53, 57]

**7.** 57

/ \

50 53

/ \ / \

26 35 15 22

/ \ / \ /

21 25 14 11 2

[57, 50, 53, 26, 35, 15, 22, 21, 25, 14, 11, 2]

**8.** 1: [53, 50, 22, 26, 35, 15, 2, 21, 25, 14, 11, 57]

2: [50, 35, 22, 26, 14, 15, 2, 21, 25, 11, 53, 57]

3: [35, 26, 22, 25, 14, 15, 2, 21, 11, 50, 53, 57]

**4: [26, 25, 22, 21, 14, 15, 2, 11, 35, 50, 53, 57]**