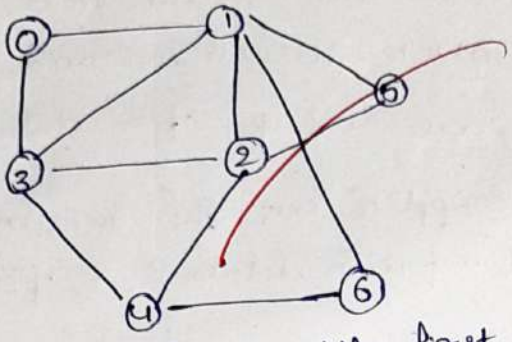
LAB-06

2100030059
B. Pavithi
S-12

Pre-lab:-

1. List down the list of traversal vertices starting from '0' by BFS and DFS methods what data structure's are used in BFS and DFS for traversal?



BFS: BFS stands for Breadth first search It is also known as level order traversal the Queue data structure is used for BFS. when we use BFS algorithm for traversal in a graph we can consider any node as a root node. For, the given graph the traversing starts from 0. once 0 is visited marked that in visited node

① 0

⑤ 0 visited.

1 Queue.

③ 0 1 visited.

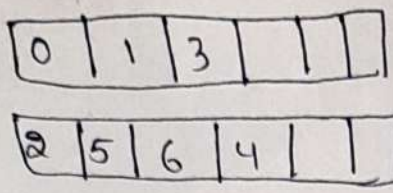
3 2 5 6 1 Queue.

0 1 3 visited.

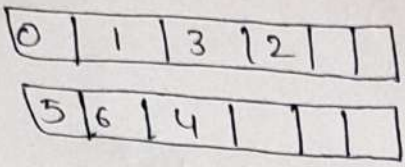
2 5 6 Queue.

2100030059
B. Revathi
S-12

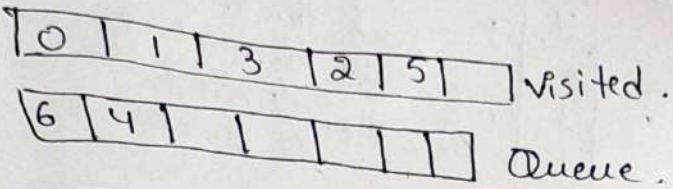
(5)



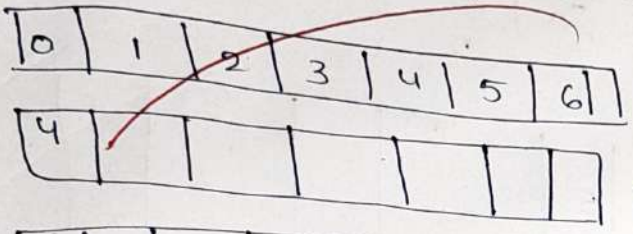
(6)



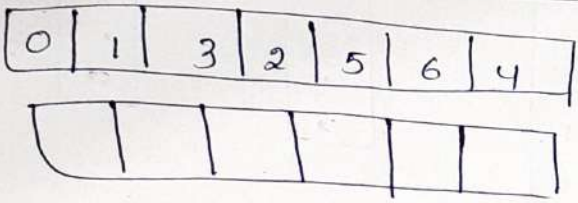
(7)



(8)



(9)

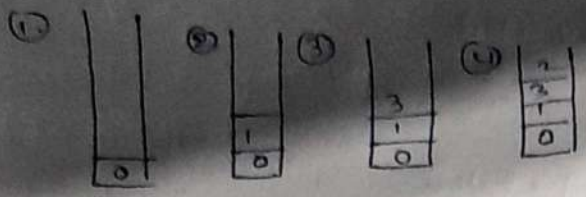


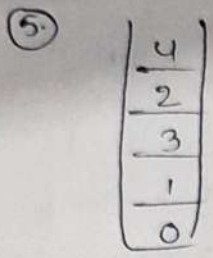
BFS of a give graph

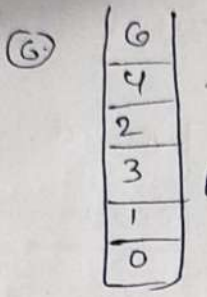
0 - 1 - 3 - 2 - 5 - 6 - 4.

DFS:- DFS stands for Depth First search It is also known as traversal in the stack data structure is used, which works on the HFD principle in DFS. In DFS traversing can be started from any node can be considered as a root node until the root node is not mentioned in the problem. Here the given root node is '0'.

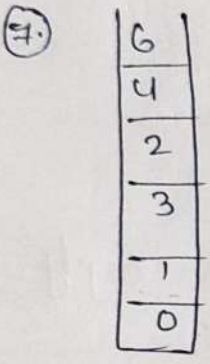
2100030056
B. Revathi
S-12

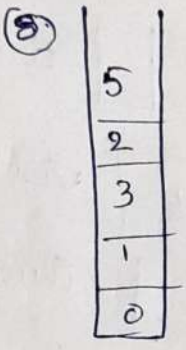


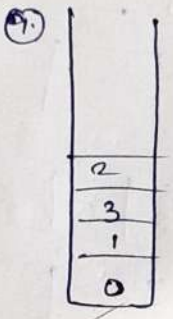


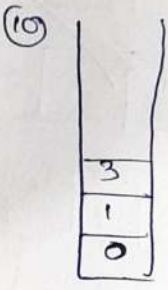


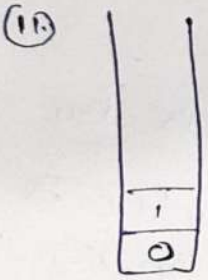
back tracking top most element is popped out from the stack.

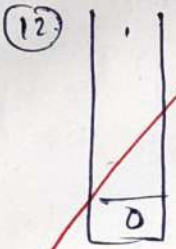




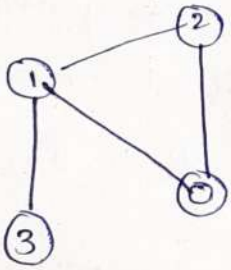








stack is empty.



I/p :- n = 4,
connections = [(0,1), (1,2), [2,0], [1,3]]
O/p :- [[1,3]].

2100030059
B. Revathi
S-12

```
Class Solution {
Public Static void main (String[] args) {
Vector <vector <int >> Critical connections (int n, Vector <
    Vector <int >> k connections)
{
    int  = 1;
    Vector <vector <int >> res;
    vector <int > time(n), low(n);
    unordered-map (int, vector <int >> g;
    for (auto conn : connections) {
    g [conn [0]] . push - back [conn[1]];
    g [conn [1]] . push - back [conn[0]];
    }
     helper [g, 0-1, int, time, low, res);
      return res;
    }
    void helpers (unordered - map <int, vector <int >> &
        g, int cur, int pre, int &cnt, vector <int > &ti
        vector <int > &low
    vector <vector <int >> &res) {
    time [cur] = low [cur] = cur++;
    for (int next: g[cur]) {
    if (time [next] == 0) {
    helper (g, next, cur, cnt, time, low, res)
      low [curr] = min (low[cur], time [next]);
      }
      if (low [next] > time (cur]) {
      res push - back (cur, next });
      }
    }
  }
};
```

2100030039
B. Jyothi
5-12.

In-lab:-

① Write a java programm to implement BFS method.

```
import java.util.ArrayDeque;
import java.util.Queue;
class Node {
  int x, y, dist;
  Node (int x, int y, int dist) {
    this.x = x;
    this.y = y;
    this.dist = dist;
  }
}
class Main {
  Private static final int[] row = {-1, 0, 0, 1};
  Private static final int[] col = {0, 1, 1, 0};
  Private static boolean is safe (int[][]
      field, boolean visited[][], int x,
                    int y)
  {
    return (field[x][y] == 1 && !visited
                    [x][y]);
  }
  Private static boolean is valid (int x, int y,
                    int m, int n) {
  return (x < m && y < N && x >= 0 &&
                    y >= 0);
  }
Private static int BFS (int[][] field)
  { int m = field.length;
    int m = field[0].length;
    boolean[][] visited = new ArrayDeque<>();
```

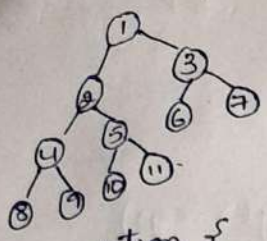
2100030059
B. Revathi
S-12

```
for(int r=0; r<m; r++){
if [field[r][0]==1){
  q.add(new.Node(1,0,0));
  visited[r][0]=true;
}}
while(!q is empty()){
int i = q.peek().x;
int j = q.peek().y;
int dist = q.peek().dist;
  q.poll()
  int (j==N-1){
  return dist;
for (int k=0, k<row length; k++){
if [is valid (i+row[k], j+col[k], m, N) <2
;s safe (field, visited, i+row[k], j+col[k]))
{
visited[i+row[k]][j+col[k]]=true;
q.add (new Node [i+row[k], j+col[k], dist+1]);
}}
  return Integer.MAX_Value;
}
Public static int final shortest Distance (int[][]
    if (mat==null || mat.length==0){
      return 0;
int  m=mat length;
int N = mat[0].length;
int[]r = {-1,-1,-1,0,0,1,1,1};
int[] c = {0,-1,0,1,-1,1,-1,0,1};
for [int i=0; i<m; i++){
for [int j=0; j<N; j++){
```

```
for(int k=0; k<r.length; k++) {
  if (mat[i][j]==0 && isValid(i+r[k], j+c[k],
                  m,N) && mat[i+r[k][j+c[k]]==1)
    mat[i+r[k][j+c[k]]= Integer.max_value;
  }
 }
}
}
for(int i=0; i<m; i++){
 for(int j=0; j<N; j++){
  if (mat[i][j]==Integer.max_value){
   mat[i][j]=0;
  }
 }
}
return BFS(mat);
}
Public static void main (String[] args)
{
  int[][] field = {
    {0,1,1,1,0,1,1,1,1,1},
    {1,1,1,1,1,1,1,1,1,1},
    {1,1,1,1,1,1,0,1,1,1},
    {1,1,1,1,0,1,1,1,1,1}
  };
int dist= find shortest Distance (field);
if (dist!=Integer.max_value){
  System.out.println(dist);
}
else{
    System.out.println("No route is Safe to
          reach destination.");
  }
}
```

210003005 9
B. Revathi
S-12.

(2)




```
class Solution {
    TreeNode xNode = null;
    int leftside, rightside;
    Public boolean btreeGameWinning(TreeNode root,
                                    int n, int x) {

        dFS(root, x);
        return Math.max(n-leftsize-rightsize-1,
        Math.max(leftsize, rightsize)) > n/2;
    }
    Private int dFS(TreeNode root, int x) {
        if (root == null) {
            return 0;
        }
        int left = dfs(root.left, x);
        int right = dfs(root.right, x);
            if(root.val == x) {
            leftSize = left;
            rightSize = right;
        }
            return left + right + 1;
        }
    }
```