

Homework OF 31-03-2025

Subject: CSW2 (CSE 2141)

Name: Arpit Kumar Mohanty

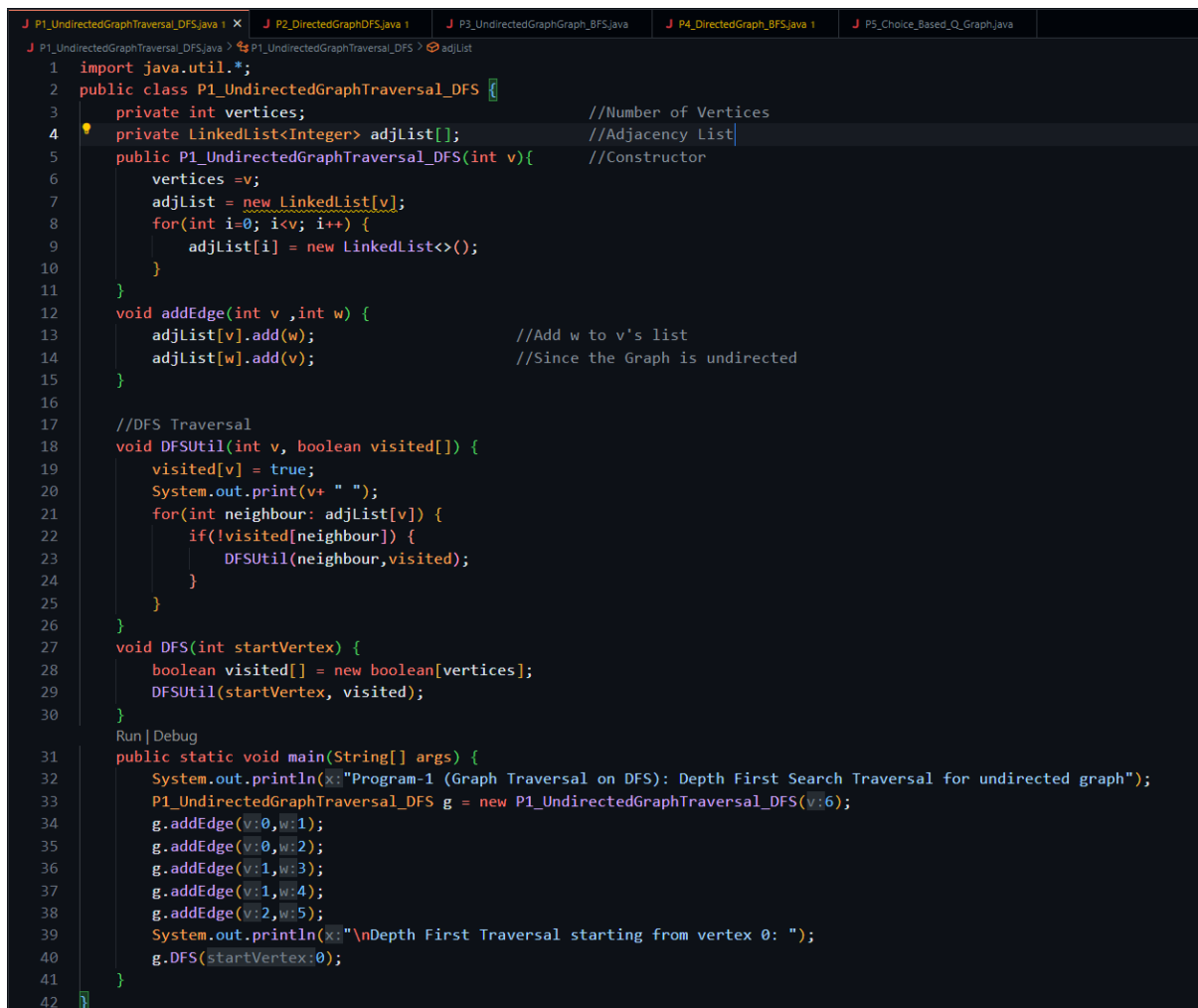
Registration Number: 2341013237

Section: 23412G1

Branch: CSE

Q1. Write Java program on Depth First Search Traversal for undirected graph

Solution:



```
1 import java.util.*;
2 public class P1_UndirectedGraphTraversal_DFS {
3     private int vertices; //Number of Vertices
4     private LinkedList<Integer> adjList[]; //Adjacency List
5     public P1_UndirectedGraphTraversal_DFS(int v) { //Constructor
6         vertices = v;
7         adjList = new LinkedList[v];
8         for(int i=0; i<v; i++) {
9             adjList[i] = new LinkedList<>();
10        }
11    }
12    void addEdge(int v ,int w) {
13        adjList[v].add(w); //Add w to v's list
14        adjList[w].add(v); //Since the Graph is undirected
15    }
16
17    //DFS Traversal
18    void DFSUtil(int v, boolean visited[]) {
19        visited[v] = true;
20        System.out.print(v+ " ");
21        for(int neighbour: adjList[v]) {
22            if(!visited[neighbour]) {
23                DFSUtil(neighbour,visited);
24            }
25        }
26    }
27    void DFS(int startVertex) {
28        boolean visited[] = new boolean[vertices];
29        DFSUtil(startVertex, visited);
30    }
31
32    public static void main(String[] args) {
33        System.out.println("Program-1 (Graph Traversal on DFS): Depth First Search Traversal for undirected graph");
34        P1_UndirectedGraphTraversal_DFS g = new P1_UndirectedGraphTraversal_DFS(6);
35        g.addEdge(0,1);
36        g.addEdge(0,2);
37        g.addEdge(1,3);
38        g.addEdge(1,4);
39        g.addEdge(2,5);
40        System.out.println("Depth First Traversal starting from vertex 0: ");
41        g.DFS(startVertex:0);
42    }
```

Output:

Note: Recompile with -Xlint:unchecked for details.

Program-1 (Graph Traversal on DFS): Depth First Search Traversal for undirected graph

Depth First Traversal starting from vertex 0:

0 1 3 4 2 5

[Done] exited with code=0 in 0.763 seconds

Q2. Write Java program on Depth First Search Traversal for directed graph

Solution:

```
J P1_UndirectedGraphTraversal_DFS.java 1 J P2_DirectedGraphDFS.java 1 X J P3_UndirectedGraphBFS.java J P4_DirectedGraph_BFS.java 1 J P5_Choice_Based_Q_Graph
J P2_DirectedGraphDFS.java > P2_DirectedGraphDFS > addVertex(char)
1 import java.util.*;
2 public class P2_DirectedGraphDFS {
3     private int vertices;
4     private LinkedList<Integer> adjList[];
5     private Map<Character, Integer> vertexIndexMap;
6     private Map<Integer, Character> indexVertexMap;
7     P2_DirectedGraphDFS(int v) {
8         vertices = v;
9         adjList = new LinkedList[v];
10        vertexIndexMap = new HashMap<>();
11        indexVertexMap = new HashMap<>();
12        for (int i = 0; i < v; i++) {
13            adjList[i] = new LinkedList<>();
14        }
15    }
16
17    void addVertex(char vertex) {
18        int index = vertexIndexMap.size();
19        vertexIndexMap.put(vertex, index);
20        indexVertexMap.put(index, vertex);
21    }
22
23    void addEdge(char v, char w) {
24        int fromIndex = vertexIndexMap.get(v);
25        int toIndex = vertexIndexMap.get(w);
26        adjList[fromIndex].add(toIndex);
27    }
28
29    void DFSUtil(int v, boolean visited[]) {
30        visited[v] = true;
31        System.out.print(indexVertexMap.get(v) + " ");
32        for (int neighbour : adjList[v]) {
33            if (!visited[neighbour]) {
34                DFSUtil(neighbour, visited);
35            }
36        }
37    }
38
39    void DFS(char startVertex) {
40        boolean visited[] = new boolean[vertices];
41        int startIndex = vertexIndexMap.get(startVertex);
42        DFSUtil(startIndex, visited);
43    }
}
```

```
J P1_UndirectedGraphTraversal_DFS.java 1 J P2_DirectedGraphDFS.java 1 X J P3_UndirectedGraphBFS.java J P4_DirectedGraph_BFS.java 1 J P5_Choice_Based_Q_Graph.java
J P2_DirectedGraphDFS.java > P2_DirectedGraphDFS > addVertex(char)
2 public class P2_DirectedGraphDFS {
38
39     void DFS(char startVertex) {
40         boolean visited[] = new boolean[vertices];
41         int startIndex = vertexIndexMap.get(startVertex);
42         DFSUtil(startIndex, visited);
43     }
44
45     Run | Debug
46     public static void main(String[] args) {
47         System.out.println(x:"Program-2 (Graph Traversal on DFS) : Depth First Search traversal for ddirected graph");
48         P2_DirectedGraphDFS g = new P2_DirectedGraphDFS(v:5);
49         g.addVertex(vertex:'a');
50         g.addVertex(vertex:'b');
51         g.addVertex(vertex:'c');
52         g.addVertex(vertex:'d');
53         g.addVertex(vertex:'e');
54
55         g.addEdge(v:'a', w:'b');
56         g.addEdge(v:'a', w:'c');
57         g.addEdge(v:'b', w:'d');
58         g.addEdge(v:'b', w:'e');
59         g.addEdge(v:'d', w:'a');
60         g.addEdge(v:'d', w:'e');
61         g.addEdge(v:'d', w:'d');
62
63         System.out.println(x:"\nDepth First Search Traversal starting from vertex a : ");
64         g.DFS(startVertex:'a');
65     }
```

Output:

```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\31-03-2025 [Chap-17]
java P2_DirectedGraphDFS
Note: P2_DirectedGraphDFS.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Program-2 (Graph Traversal on DFS) : Depth First Search traversal for ddirected graph

Depth First Search Traversal starting from vertex a :
a b d e c
[Done] exited with code=0 in 0.74 seconds
```

Q3. Write Java program on Breadth First Search Traversal for undirected graph.

Solution:

```

1  import java.util.*;
2  public class P3_UndirectedGraphGraph_BFS {
3      private int vertices;           // Number of vertices
4      private LinkedList<Integer>[] adjList; // Adjacency List
5
6      // Constructor
7      P3_UndirectedGraphGraph_BFS(int v) {
8          vertices = v;
9          adjList = new LinkedList[v];
10         for (int i = 0; i < v; i++) {
11             adjList[i] = new LinkedList<>(); // Initialize each adjacency list
12         }
13     }
14
15     // Add edge to the graph (Undirected Graph)
16     void addEdge(int v, int w) {
17         adjList[v].add(w);
18         adjList[w].add(v);
19     }
20
21     // BFS Traversal
22     void BFS(int startVertex) {
23         boolean[] visited = new boolean[vertices];
24         Queue<Integer> queue = new LinkedList<>(); // Using Queue for BFS
25
26         visited[startVertex] = true;
27         queue.add(startVertex);
28
29         while (!queue.isEmpty()) {
30             int v = queue.poll();
31             System.out.print(v + " ");
32
33             // Traverse all adjacent vertices of v
34             for (int neighbor : adjList[v]) {
35                 if (!visited[neighbor]) {
36                     visited[neighbor] = true;
37                     queue.add(neighbor);
38                 }
39             }
40         }
41     }
42

```

```
// BFS Traversal
void BFS(int startVertex) {
    boolean[] visited = new boolean[vertices];
    Queue<Integer> queue = new LinkedList<>(); // Using Queue for BFS

    visited[startVertex] = true;
    queue.add(startVertex);

    while (!queue.isEmpty()) {
        int v = queue.poll();
        System.out.print(v + " ");

        // Traverse all adjacent vertices of v
        for (int neighbor : adjList[v]) {
            if (!visited[neighbor]) {
                visited[neighbor] = true;
                queue.add(neighbor);
            }
        }
    }
}

Run | Debug
public static void main(String[] args) {
    System.out.println(x:"Breadth-First Search (BFS) Traversal for an Undirected Graph:");

    P3_UndirectedGraphGraph_BFS g = new P3_UndirectedGraphGraph_BFS(v:6);
    g.addEdge(v:0, w:1);
    g.addEdge(v:0, w:2);
    g.addEdge(v:1, w:3);
    g.addEdge(v:1, w:4);
    g.addEdge(v:2, w:5);

    System.out.println(x:"\nBFS Traversal starting from vertex 0:");
    g.BFS(startVertex:0);
}
}
```

Output:

```
[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\31-03-202
P3_UndirectedGraphGraph_BFS
Note: P3_UndirectedGraphGraph_BFS.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Breadth-First Search (BFS) Traversal for an Undirected Graph:

BFS Traversal starting from vertex 0:
0 1 2 3 4 5
[Done] exited with code=0 in 0.739 seconds
```

Q4. Write Java program on Breadth First Search Traversal for directed graph

Solution:

```

J P1_UndirectedGraphTraversal_DFS.java 1 J P2_DirectedGraphDFS.java 1 J P3_UndirectedGraphGraph_BFS.java 1 J P4_DirectedGraph_BFS.java 1 X J P
J P4_DirectedGraph_BFS.java > P4_DirectedGraph_BFS > DirectedGraphBFS(int)
1 import java.util.*;
2
3 public class P4_DirectedGraph_BFS {
4     private int vertices;
5     private LinkedList<Integer>[] adjList;
6     private Map<Character, Integer> vertexIndexMap; // Maps characters to indices
7     private Map<Integer, Character> indexVertexMap; // Maps indices back to characters
8
9     // Constructor
10    P4_DirectedGraph_BFS(int v) {
11        vertices = v;
12        adjList = new LinkedList[v];
13        vertexIndexMap = new HashMap<>();
14        indexVertexMap = new HashMap<>();
15        for (int i = 0; i < v; i++) {
16            adjList[i] = new LinkedList<>(); // Initialize adjacency lists
17        }
18    }
19
20    // Add a vertex
21    void addVertex(char vertex) {
22        int index = vertexIndexMap.size();
23        vertexIndexMap.put(vertex, index);
24        indexVertexMap.put(index, vertex);
25    }
26
27    // Add a directed edge (from 'v' to 'w')
28    void addEdge(char v, char w) {
29        int fromIndex = vertexIndexMap.get(v);
30        int toIndex = vertexIndexMap.get(w);
31        adjList[fromIndex].add(toIndex);
32    }
33
34    // BFS Traversal
35    void BFS(char startVertex) {
36        boolean[] visited = new boolean[vertices];
37        Queue<Integer> queue = new LinkedList<>();
38        int startIndex = vertexIndexMap.get(startVertex);
39
40        visited[startIndex] = true;
41        queue.add(startIndex);
42

```

```

38     int startIndex = vertexIndexMap.get(startVertex);
39
40     visited[startIndex] = true;
41     queue.add(startIndex);
42
43     while (!queue.isEmpty()) {
44         int v = queue.poll();
45         System.out.print(indexVertexMap.get(v) + " ");
46
47         for (int neighbor : adjList[v]) {
48             if (!visited[neighbor]) {
49                 visited[neighbor] = true;
50                 queue.add(neighbor);
51             }
52         }
53     }
54 }
55
Run | Debug
56 public static void main(String[] args) {
57     System.out.println(x:"Breadth-First Search (BFS) Traversal for a Directed Graph:");
58
59     P4_DirectedGraph_BFS g = new P4_DirectedGraph_BFS(v:5);
60     g.addVertex(vertex:'A');
61     g.addVertex(vertex:'B');
62     g.addVertex(vertex:'C');
63     g.addVertex(vertex:'D');
64     g.addVertex(vertex:'E');
65
66     g.addEdge(v:'A', w:'B');
67     g.addEdge(v:'A', w:'C');
68     g.addEdge(v:'B', w:'D');
69     g.addEdge(v:'B', w:'E');
70     g.addEdge(v:'D', w:'A');
71     g.addEdge(v:'D', w:'E');
72
73     System.out.println(x:"\nBFS Traversal starting from vertex A:");
74     g.BFS(startVertex:'A');
75 }
76 }

```

Output:

```

[Running] cd "a:\Programs\HTML & CSS (from Sems)\4th Semester\CSW-2\31-03-2025 [
Note: P4_DirectedGraph_BFS.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
Breadth-First Search (BFS) Traversal for a Directed Graph:

BFS Traversal starting from vertex A:
A B C D E
[Done] exited with code=0 in 0.774 seconds

```

Q5. Java Program on Graph Traversal using Switch case as: a) addEdge b) BFS c) DFS d) Exit default: invalid choice for the Undirected Graph. Use Scanner Class in the main function for choices in the run time and

adding the edges.

Solution:

```
J P1_UndirectedGraphTraversal_DFS.java 1 J P2_DirectedGraphDFS.java 1 J P3_UndirectedGraphGraph_BFS.java 1 J P4_DirectedGraph_BFS.java 1 J P5_Choice_Based_Q_Graph.java
J P5_Choice_Based_Q_Graph.java > P5_Choice_Based_Q_Graph
1 import java.util.*;
2 public class P5_Choice_Based_Q_Graph {
3     private List<Integer>[] adj;
4
5     @SuppressWarnings("unchecked")
6     P5_Choice_Based_Q_Graph(int v) {
7         adj = new LinkedList[v];
8         Arrays.setAll(adj, i -> new LinkedList<>());
9     }
10
11     void addEdge(int v, int w) {
12         if (v >= 0 && v < adj.length && w >= 0 && w < adj.length) {
13             adj[v].add(w);
14             adj[w].add(v);
15         } else {
16             System.out.println("Invalid edge! Vertices should be between 0 and " + (adj.length - 1));
17         }
18     }
19
20     void BFS(int s) {
21         if (s < 0 || s >= adj.length) {
22             System.out.println(x:"Invalid start vertex for BFS!");
23             return;
24         }
25
26         boolean[] vis = new boolean[adj.length];
27         Queue<Integer> q = new LinkedList<>();
28         vis[s] = true;
29         q.add(s);
30
31         System.out.print(s:"BFS: ");
32         while (!q.isEmpty()) {
33             int v = q.poll();
34             System.out.print(v + " ");
35
36             for (int n : adj[v]) {
37                 if (!vis[n]) {
38                     vis[n] = true;
39                     q.add(n);
40                 }
41             }
42     }
```



```

P5_Choice_Based_Q_Graph.java > P5_Choice_Based_Q_Graph
2   public class P5_Choice_Based_Q_Graph {
20   void BFS(int s) {
31       System.out.print(s:"BFS: ");
32       while (!q.isEmpty()) {
33           int v = q.poll();
34           System.out.print(v + " ");
35
36           for (int n : adj[v]) {
37               if (!vis[n]) {
38                   vis[n] = true;
39                   q.add(n);
40               }
41           }
42       }
43       System.out.println();
44   }
45
46   void DFS(int s) {
47       if (s < 0 || s >= adj.length) {
48           System.out.println(x:"Invalid start vertex for DFS!");
49           return;
50       }
51
52       boolean[] vis = new boolean[adj.length];
53       System.out.print(s:"DFS: ");
54       DFSUtil(s, vis);
55       System.out.println();
56   }
57
58   private void DFSUtil(int v, boolean[] vis) {
59       vis[v] = true;
60       System.out.print(v + " ");
61       for (int n : adj[v]) {
62           if (!vis[n]) {
63               DFSUtil(n, vis);
64           }
65       }
66   }
67

```

```

P5_Choice_Based_Q_Graph.java > P5_Choice_Based_Q_Graph
2 public class P5_Choice_Based_Q_Graph {
68 public static void main(String[] args) {
69     Scanner sc = new Scanner(System.in);
70     System.out.print(s:"Enter number of vertices: ");
71     int vertices = sc.nextInt();
72     P5_Choice_Based_Q_Graph g = new P5_Choice_Based_Q_Graph(vertices);
73
74     while (true) {
75         System.out.println(x:"\n1) Add Edge  2) BFS  3) DFS  4) Exit");
76         System.out.print(s:"Enter choice: ");
77         int choice = sc.nextInt();
78
79         switch (choice) {
80             case 1 -> {
81                 System.out.print(s:"Enter two vertices: ");
82                 int v = sc.nextInt(), w = sc.nextInt();
83                 g.addEdge(v, w);
84             }
85             case 2 -> {
86                 System.out.print(s:"Start vertex for BFS: ");
87                 g.BFS(sc.nextInt());
88             }
89             case 3 -> {
90                 System.out.print(s:"Start vertex for DFS: ");
91                 g.DFS(sc.nextInt());
92             }
93             case 4 -> {
94                 System.out.println(x:"Exiting...");
95                 sc.close();
96                 return;
97             }
98             default -> System.out.println(x:"Invalid choice! Please try again.");
99         }
100     }
101 }
102 }

```

Output:

```
● PS C:\WINDOWS\System32\WindowsPowerShell\v1.0> & 'C:\Program Files\
\Roaming\Code\User\workspaceStorage\9be4fc751454dc38f56ff6747146354a
Enter number of vertices: 4

1) Add Edge  2) BFS  3) DFS  4) Exit
Enter choice: 1
Enter two vertices: 1 2

1) Add Edge  2) BFS  3) DFS  4) Exit
Enter choice: 2
Start vertex for BFS: 2
BFS: 2 1

1) Add Edge  2) BFS  3) DFS  4) Exit
Enter choice: 3
Start vertex for DFS: 2
DFS: 2 1

1) Add Edge  2) BFS  3) DFS  4) Exit
Enter choice: 4
Exiting...
○ PS C:\WINDOWS\System32\WindowsPowerShell\v1.0> █
```