****

**Green University of Bangladesh**

**Department of Computer Science and Engineering (CSE)**

**Faculty of Sciences and Engineering**

**Semester: (Summer, Year: 2023), B.Sc. in CSE (Day)**

**CLP-1**

**Course Title: Algorithms Lab**

**Course Code: CSE-202 Section: DC**

**Student Details**

|  |  |  |
| --- | --- | --- |
| **Name** | | **ID** |
|  | Md. Sohanur Rahman | 213902106 |

**Lab Date : 23.02.2023**

**Submission Date : 26.02.2023**

**Course Teacher’s Name : Md. Sultanul Islam Ovi**

**[For Teachers use only: Don’t Write Anything inside this box]**

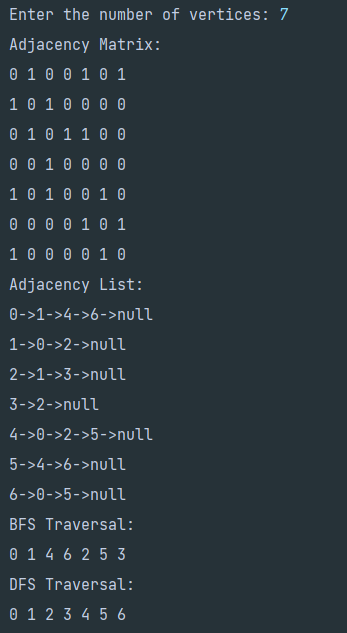
|  |
| --- |
| **Lab Report Status**  **Marks: ………………………………… Signature: .....................**  **Comments: .............................................. Date: ..............................** |

**Problem statement-1:** Implement adjacency matrix, adjacency list, BFS and DFS algorithms on the following graph.

Code:

*package* Algorithms\_Lab;  
  
*import* java.util.\*;  
  
*public class* MergeGraph {  
 *private int*[][] Matrix;  
 *private List*<*List*<Integer>> List;  
 *private int* vertices;  
 *public* MergeGraph(*int* vertices){  
 *this*.vertices=vertices;  
 *this*.Matrix=*new int*[vertices][vertices];  
 *this*.List=*new* ArrayList<>();  
 *for*(*int* i=0;i<vertices;i++){  
 List.add(*new* ArrayList<>());  
 }  
 }  
 *public void* addEdge(*int* src,*int* dest){  
 Matrix[src][dest]=1;  
 Matrix[dest][src]=1;  
 List.get(src).add(dest);  
 }  
 *//For print adjacencyMatrix  
 public void* printMatrix(){  
 System.***out***.println("Adjacency Matrix: ");  
 *for*(*int* i=0;i<vertices;i++){  
 *for*(*int* j=0;j<vertices;j++){  
 System.***out***.print(Matrix[i][j]+" ");  
 }  
 System.***out***.println();  
 }  
 }  
 *//For print adjacency list.  
 public void* printList(){  
 System.***out***.println("Adjacency List:");  
 *for*(*int* i=0;i<vertices;i++){  
 System.***out***.print(i+"->");  
 *for*(*int* j=0;j<List.get(i).size();j++){  
 *int* curr=List.get(i).get(j);  
 System.***out***.print(curr+"->");  
 }  
 System.***out***.print("null");  
 System.***out***.println();  
 }  
 }  
 *//For BFS traversal.  
 public void* bfs(*int* curr){  
 System.***out***.println("BFS Traversal:");  
 *Queue*<Integer> q=*new* LinkedList<>();  
 *boolean*[] visited=*new boolean*[vertices];  
 q.add(curr);  
 visited[curr]=*true*;  
 *while*(!q.isEmpty()){  
 curr=q.remove();  
 System.***out***.print(curr+" ");  
 *for*(*int* i=0;i<List.get(curr).size();i++){  
 *int* n=List.get(curr).get(i);  
 *if*(!visited[n]){  
 *//System.out.print(curr+" ");  
 // q.add(n);* visited[n]=*true*;  
 q.add(n);  
 }  
 }  
 }  
 System.***out***.println();  
 System.***out***.println("DFS Traversal:");  
 }  
 *public void* dfs(*int* curr, *boolean*[] visited){  
 *if*(visited[curr]){  
 *return*;  
 }  
 System.***out***.print(curr+" ");  
 visited[curr]=*true*;  
 *for*(*int* i=0;i<List.get(curr).size();i++){  
 *int* n= List.get(curr).get(i);  
 dfs(n,visited);  
 }  
 }  
  
  
 *public static void* main(String[] args) {  
 Scanner scan=*new* Scanner(System.***in***);  
 System.***out***.print("Enter the number of vertices: ");  
 *int* v=scan.nextInt();  
 MergeGraph obj=*new* MergeGraph(v);  
 obj.addEdge(0,1);  
 obj.addEdge(0,4);  
 obj.addEdge(0,6);  
 obj.addEdge(1,0);  
 obj.addEdge(1,2);  
 obj.addEdge(2,1);  
 obj.addEdge(2,3);  
 obj.addEdge(3,2);  
 obj.addEdge(4,0);  
 obj.addEdge(4,2);  
 obj.addEdge(4,5);  
 obj.addEdge(5,4);  
 obj.addEdge(5,6);  
 obj.addEdge(6,0);  
 obj.addEdge(6,5);  
 obj.printMatrix();  
 obj.printList();  
 obj.bfs(0);  
 obj.dfs(0,*new boolean*[v]);  
 }  
}

Output:

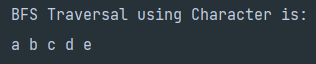


**Problem Statement-2:** Create a graph using Character variables as nodes and implement BFS/DFS algorithm on that graph

BFS Code:

*package* Algorithms\_Lab;  
  
*import* java.util.\*;  
  
*public class* BFS {  
 *Set*<Character> visited;  
 *Queue*<Character> q;  
 HashMap<Character, *List*<Character>> graph =*new* HashMap<>();  
 *public void* createGraph(){  
 graph.put('a',Arrays.*asList*('b','c','d'));  
 graph.put('b',Arrays.*asList*('a','c'));  
 graph.put('c',Arrays.*asList*('a','b','e'));  
 graph.put('d',Arrays.*asList*('a'));  
 graph.put('e',Arrays.*asList*('c'));  
 }  
 *public void* bfs(*char* start){  
 visited=*new* HashSet<>();  
 q=*new* LinkedList<>();  
 q.add(start);  
 visited.add(start);  
 *while*(!q.isEmpty()){  
 *char* curr=q.remove();  
 System.***out***.print(curr+" ");  
 visited.add(curr);  
 *List*<Character> adjNodes=graph.get(curr);  
 *for*(*char* adjnode:adjNodes){  
 *if*(!visited.contains(adjnode)) {  
 q.add(adjnode);  
 visited.add(adjnode);  
 }  
 }  
 }  
 }  
 *public static void* main(String[] args) {  
 BFS obj=*new* BFS();  
 obj.createGraph();  
 System.***out***.println("BFS Traversal using Character is:");  
 obj.bfs('a');  
 }  
}

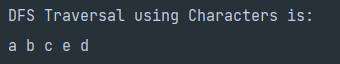
Output:



DFS Code:

*package* Algorithms\_Lab;  
  
*import* java.util.\*;  
  
*public class* DFS {  
 *Set*<Character> visited=*new* HashSet<>();  
 HashMap<Character, *List*<Character>> graph=*new* HashMap<>();  
 *public void* createGraph(){  
 graph.put('a',Arrays.*asList*('b','c','d'));  
 graph.put('b',Arrays.*asList*('a','c'));  
 graph.put('c',Arrays.*asList*('a','b','e'));  
 graph.put('d',Arrays.*asList*('a'));  
 graph.put('e',Arrays.*asList*('c'));  
 }  
 *public void* dfs(*char* start){  
 visited.add(start);  
 System.***out***.print(start+" ");  
 *List*<Character>neighbors=graph.get(start);  
 *if*(neighbors!=*null*){  
 *for*(*char* n:neighbors){  
 *if*(!visited.contains(n)){  
 dfs(n);  
 }  
 }  
 }  
 }  
 *public static void* main(String[] args) {  
 DFS obj=*new* DFS();  
 obj.createGraph();  
 System.***out***.println("DFS Traversal using Characters is:");  
 obj.dfs('a');  
  
 }  
}

Output:

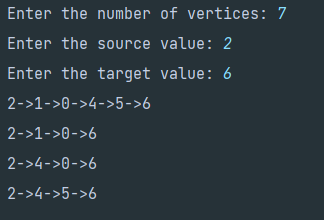


**Problem Statement-3:** Find the path between two nodes (2 and 6) using DFS.

Code:

*package* Algorithms\_Lab;  
  
*import* java.util.ArrayList;  
*import* java.util.Scanner;  
  
*public class* PrintAllPaths {  
 *static class* Edge{  
 *int* src,dest;  
 *public* Edge(*int* src,*int* dest){  
 *this*.src=src;  
 *this*.dest=dest;  
 }  
 }  
 *public static void* createGraph(ArrayList<Edge>[] graph){  
 *for*(*int* i=0;i< graph.length;i++){  
 graph[i]=*new* ArrayList<>();  
 }  
 graph[0].add(*new* Edge(0,1));  
 graph[0].add(*new* Edge(0,4));  
 graph[0].add(*new* Edge(0,6));  
 graph[1].add(*new* Edge(1,2));  
 graph[1].add(*new* Edge(1,0));  
 graph[2].add(*new* Edge(2,1));  
 graph[2].add(*new* Edge(2,4));  
 graph[2].add(*new* Edge(2,3));  
 graph[3].add(*new* Edge(3,2));  
 graph[4].add(*new* Edge(4,0));  
 graph[4].add(*new* Edge(4,2));  
 graph[4].add(*new* Edge(4,5));  
 graph[5].add(*new* Edge(5,4));  
 graph[5].add(*new* Edge(5,6));  
 graph[6].add(*new* Edge(6,0));  
 graph[6].add(*new* Edge(6,5));  
 }  
 *public static void* printAllPath(ArrayList<Edge>[] graph,*int* src,*int* tar,String path,*boolean*[] visited){  
 *if*(src==tar){  
 System.***out***.println(path);  
 *return*;  
 }  
 *for*(*int* i=0;i< graph[src].size();i++){  
 Edge e=graph[src].get(i);  
 *if*(!visited[e.dest]){  
 visited[e.dest]=*true*;  
 *printAllPath*(graph,e.dest,tar,path+"->"+e.dest,visited);  
 visited[e.dest]=*false*;  
 }  
 }  
 }  
 *public static void* main(String[] args) {  
 Scanner scan=*new* Scanner(System.***in***);  
 System.***out***.print("Enter the number of vertices: ");  
 *int* v=scan.nextInt();  
 ArrayList<Edge>[] graph=*new* ArrayList[v];  
 *createGraph*(graph);  
 System.***out***.print("Enter the source value: ");  
 *int* src=scan.nextInt();  
 System.***out***.print("Enter the target value: ");  
 *int* tar=scan.nextInt();  
 *boolean*[] visited=*new boolean*[v];  
 visited[src]=*true*;  
 *printAllPath*(graph,src,tar,""+src,visited);  
 }  
}

Output:



\*\*\*\*\*Thanks You\*\*\*\*\*