Dr. D. Y. Patil Pratishthan’s

**D. Y. Patil Institute of Master of Computer Applications & Management**



Sector No.29, Behind Akurdi Railway Station, Pradhikaran, Nigdi, Pune – 411044

Tel No: 020-27640998, 202737393, Fax no: 27653054,

Website: www.dypimca.ac.in, Email: director@dypimca.ac.in

**(Approved by AICTE, Recognized by DTE, Mah.; Affiliated to SPPU)**

**DSA Part B: Preliminary Term Journal Submission**

**INDEX**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Sr.No** | **Name of the Program** | **Date** | **Page**  **No** | **Sign** |
| 1 | Write a program to create a graph and perform the following operations: a) Add Vertex b) Add edge c) Display | 4/10/2023 | 2 |  |
| 2 | Write a program to create a graph and perform the following operations: a) Delete edge b) Has edge | 4/10/2023 | 3 |  |
| 3 | Write a program to create a graph and perform DFS. | 9/10/2023 | 6 |  |
| 4 | Write a program to create a graph and perform BFS. | 14/10/2023 | 7 |  |
| 5 | Write a program to display adjacency List and Matrix | 17/10/2023 | 9 |  |

**Name**-Sakshi Jagtap

**Roll no**-184

**Q1] Write a program to create a graph and perform the following operations:**

**a) Add Vertex b) Add edge c) Display**

**Code-**

class graph

{

constructor()

{

this.adList={};

}

adVertex(vertex)

{

if(!this.adList[vertex])

{

this.adList[vertex]=new Set();

}

}

adEdge(vertex1,vertex2)

{

if(!this.adList[vertex1])

{

this.adVertex(vertex2);

}

this.adList[vertex1].add(vertex2);

this.adList[vertex2].add(vertex1);

}

display()

{

for(let vertex in this.adList)

{

console.log(vertex+"->"+[...this.adList[vertex]]);

} } }

const gr = new graph(); console.log("adding vertex :"); gr.adVertex("a");

gr.adVertex("b");

gr.adVertex("c");

gr.display();

console.log("adding edge :"); gr.adEdge("a","b"); gr.adEdge("a","c"); gr.adEdge("b","c");

gr.display();

**Output-**

adding vertex :

a-> b-> c-> adding edge : a->b,c b->a,c

**Name**-Sakshi Jagtap

**Roll no**-184

**Q2] Write a program to create a graph and perform the following operations: a) Delete edge b) Has edge**

**Code-**

class graph

{

constructor()

{

this.adList={};

}

adVertex(vertex)

{

if(!this.adList[vertex])

{

this.adList[vertex]=new Set();

}

}

adEdge(vertex1,vertex2)

{

if(!this.adList[vertex1])

{

this.adVertex(vertex2);

}

this.adList[vertex1].add(vertex2);

this.adList[vertex2].add(vertex1);

}

remEdge(vertex1,vertex2)

{

this.adList[vertex1].delete(vertex2);

this.adList[vertex2].delete(vertex1);

}

hasEdge(vertex1,vertex2)

{

Return

{

this.adList[vertex1].has(vertex2)&&this.adList[vertex2].has(vertex1) };

}

display()

{

for(let vertex in this.adList)

{

console.log(vertex+"->"+[...this.adList[vertex]]);

}

} }

const gr = new graph(); console.log("adding vertex :"); gr.adVertex("a");

gr.adVertex("b");

gr.adVertex("c");

gr.display();

console.log("adding edge :"); gr.adEdge("a","b"); gr.adEdge("a","c"); gr.adEdge("b","c");

gr.display();

console.log("Removing edge: ") gr.remEdge("a","b");

gr.display();

console.log("Edge between a and b ");

gr.hasEdge("a" , "b")

gr.display();

**Output-**

adding vertex :

a-> b-> c-> adding edge :

a->b,c b->a,c c->a,b Removing edge: a->c b>c c->a,b Edge between a and b a->c b->c c->a,b

**Name**-Sakshi Jagtap

**Roll no**-184

**Q3] Write a program to create a graph and perform DFS.**

**Code-**

class Graph {

constructor(v) {

this.V = v;

this.adj = new Array(v);

for (let i = 0; i < v; i++) {

this.adj[i] = [];

}

}

addEdge(v, w)

{

this.adj[v].push(w);

}

DFSUtil(v, visited) {

visited[v] = true;

console.log(v + " ");

for (let i of this.adj[v].values()) {

let n = i;

if (!visited[n]) this.DFSUtil(n, visited);

}

}

DFS(v) {

let visited = new Array(this.V);

for (let i = 0; i < this.V; i++)

{

visited[i] = false;

}

this.DFSUtil(v, visited);

}

}

// Driver Code

g = new Graph(4);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 0);

g.addEdge(2, 3);

g.addEdge(3, 3);

console.log("Following is Depth First Traversal " + "(starting from vertex 2)");

g.DFS(2);

**Output-**

Following is Depth First Traversal (starting from vertex 2)

2

0

1

3

**Name**-Sakshi Jagtap

**Roll no**-184

**Q4]. Write a program to create a graph and perform BFS.**

**Code-**

class Graph { // Constructor constructor(v) {

this.V = v;

this.adj = new Array(v);

for (let i = 0; i < v; i++) this.adj[i] = [];

}

// Function to add an edge into the graph addEdge(v, w) { // Add w to v's list. this.adj[v].push(w);

}

// prints BFS traversal from a given source s

BFS(s) {

let visited = new Array(this.V);

for (let i = 0; i < this.V; i++) {

visited[i] = false;

}

// Create a queue for BFS

let queue = [];

// Mark the current node as visited and enqueue it

visited[s] = true;

queue.push(s);

while (queue.length > 0) {

// Dequeue a vertex from queue and print it

s = queue[0];

console.log(s + " ");

queue.shift();

this.adj[s].forEach((adjacent, i) => {

if (!visited[adjacent]) {

visited[adjacent] = true;

queue.push(adjacent);

}

});

} } }

g = new Graph(4);

g.addEdge(0, 1);

g.addEdge(0, 2);

g.addEdge(1, 2);

g.addEdge(2, 0);

g.addEdge(2, 3);

g.addEdge(3, 3);

console.log(

"Following is Breadth First Traversal " + "(starting from vertex 2)");

g.BFS(2);

**Output-**

Following is Breadth First Traversal (starting from vertex 2)

2

0

3

1

**Name**-Sakshi Jagtap

**Roll no**-184

**Q5] Write a program to display adjacency List**

class Graph {

constructor() {

this.graph = new Map();}

addVertex(vertex) {

if (!this.graph.has(vertex)) {

this.graph.set(vertex, []);

}}

addEdge(source, destination) {

if (this.graph.has(source) && this.graph.has(destination)) { this.graph.get(source).push(destination);

this.graph.get(destination).push(source); // For undirected graphs

}}

displayAdjacencyList() {

for (const [vertex, neighbors] of this.graph) {

console.log(`${vertex} -> ${neighbors.join(', ')}`);

} } }

const graph = new Graph(); graph.addVertex(0);

graph.addVertex(1);

graph.addVertex(2);

graph.addVertex(3);

graph.addEdge(0, 1);

graph.addEdge(0, 2);

graph.addEdge(1, 2);

graph.addEdge(2, 3); console.log("Adjacency List:"); graph.displayAdjacencyList();

**Output-**

Adjacency List:

0-> 1, 2

1-> 0, 2

2-> 0, 1, 3

3 -> 2