**Data Structure and Algorithm Practicals**

9. Graph implementation and graph traversals

class Graph {

constructor() {

this.AdjList = new Map();

console.log('Di-graph');

}

addVertex(vertex) {

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

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

} else {

throw 'Already Exist!!!';

}

}

addEdge(vertex, node) {

if (this.AdjList.has(vertex)) {

if (this.AdjList.has(node)){

let arr = this.AdjList.get(vertex);

if(!arr.includes(node)){

arr.push(node);

}else{

throw `Can't add '${node}', it already exists`;

}

}else {

throw `Can't add non-existing vertex ->'${node}'`;

}

} else {

throw `You should add '${vertex}' first`;

}

}

print() {

console.log(this.AdjList);

for (let [key, value] of this.AdjList) {

console.log(key, value);

}

}

createVisitedObject(){

let arr = {};

for(let key of this.AdjList.keys()){

arr[key] = false;

}

return arr;

}

bfs(startingNode){

console.log('\nBFS')

let visited = this.createVisitedObject();

let q = [];

visited[startingNode] = true;

q.push(startingNode);

while(q.length){

let current = q.pop()

console.log(current);

let arr = this.AdjList.get(current);

for(let elem of arr){

if(!visited[elem]){

visited[elem] = true;

q.unshift(elem)

}

}

}

}

dfs(startingNode){

console.log('\nDFS')

let visited = this.createVisitedObject();

this.dfsHelper(startingNode, visited);

}

dfsHelper(startingNode, visited){

visited[startingNode] = true;

console.log(startingNode);

let arr = this.AdjList.get(startingNode);

for(let elem of arr){

if(!visited[elem]){

this.dfsHelper(elem, visited);

}

}

}

doesPathExist(firstNode, secondNode){

// we will approach this BFS way

let path = [];

let visited = this.createVisitedObject();

let q = [];

visited[firstNode] = true;

q.push(firstNode);

while(q.length){

let node = q.pop();

path.push(node);

let elements = this.AdjList.get(node);

if(elements.includes(secondNode)){

console.log(path.join('->'))

return true;

}else{

for(let elem of elements){

if(!visited[elem]){

visited[elem] = true;

q.unshift(elem);

}

}

}

}

return false;

}

}

function test (arg1, arg2){

if(arg1 === arg2){

console.log(`${arg1} = ${arg2} \t-> passing`)

}else{

throw 'Not passing';

}

}

// IMPLEMENTATION

let g = new Graph();

let arr = ['A', 'B', 'C', 'D', 'E', 'F'];

for (let i = 0; i < arr.length; i++) {

g.addVertex(arr[i]);

}

g.addEdge('A', 'B');

g.addEdge('A', 'D');

g.addEdge('A', 'E');

g.addEdge('B', 'C');

g.addEdge('D', 'E');

g.addEdge('E', 'F');

g.addEdge('E', 'C');

g.addEdge('C', 'F');

g.print();

g.bfs('A');

g.dfs('A');