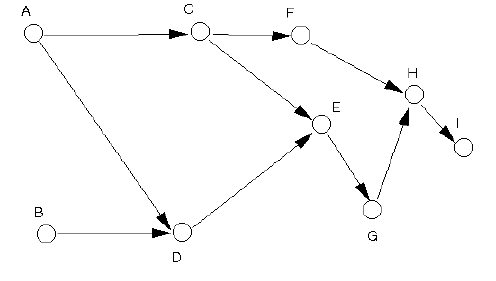
**WIA1002 TUTORIAL 8**

**GRAPH**

1. Write an adjacency matrix and an adjacency list for the following graph.



ANSWER:

Adjacency Matrix:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | A | B | C | D | E | F | G | H | I |
| A | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| B | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| C | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| D | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| E | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| F | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| G | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| H | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| I | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

Adjacency List:

|  |  |  |  |
| --- | --- | --- | --- |
| A-> | C-> | D | / |
| B-> | D-> | / |  |
| C-> | E-> | F | / |
| D-> | E | / |  |
| E-> | G | / |  |
| F-> | H | / |  |
| G-> | H | / |  |
| H-> | I | / |  |
| I | / |  |  |

1. Represent the graph in question 1 using a 2 dimensional array. You use the adjacency matrix or the adjacency list for this purpose?

ANSWER:

Adjacency Matrix, it is a 2D array where the rows and columns represent the vertices in the graph, and the values in the matrix represent the edges between them.

int[][] adjacencyMatrix = {

{ 0, 0, 1, 1, 0, 0, 0, 0, 0 },

{ 0, 0, 0, 1, 0, 0, 0, 0, 0 },

{ 1, 0, 0, 0, 1, 1, 0, 0, 0 },

{ 1, 1, 0, 0, 1, 0, 0, 0, 0 },

{ 0, 0, 1, 1, 0, 0, 1, 0, 0 },

{ 0, 0, 1, 0, 0, 0, 0, 1, 0 },

{ 0, 0, 0, 0, 1, 0, 0, 1, 0 },

{ 0, 0, 0, 0, 0, 1, 1, 0, 1 },

{ 0, 0, 0, 0, 0, 0, 0, 1, 0 }

};

1. Write code to create the graph using linked-list representation. You use the adjacency matrix or the adjacency list for this purpose?

ANSWER:

Adjacency list.

String[][] vert = {“A”,”B”,”C”,”D”,”E”,”F”,”G”,”H”,”I”};

for (String str : vert)

myGraph.addVertex(str);

myGraph.addEdge(“A”, “C”, 1);

myGraph.addEdge(“A”, “D”, 1);

myGraph.addEdge(“B”, “D”, 1);

myGraph.addEdge(“C”, “E”, 1);

myGraph.addEdge(“C”, “F”, 1);

myGraph.addEdge(“D”, “E”, 1);

myGraph.addEdge(“E”, “G”, 1);

myGraph.addEdge(“F”, “H”, 1);

myGraph.addEdge(“G”, “H”, 1);

myGraph.addEdge(“H”, “I”, 1);

LinkedList<Vertex> listA =new LinkedList<>(); listA.addLast(A); listA.addLast(C); listA.addLast(D);

LinkedList<Vertex> listB = new LinkedList<>(); listB.addLast(B); listB.addLast(D);

LinkedList<Vertex> listC = new LinkedList<>(); listC.addLast(C); listC.addLast(E); listC.addLast(F);

LinkedList<Vertex> listD = new LinkedList<>(); listD.addLast(D); listD.addLast(E);

LinkedList<Vertex> listE = new LinkedList<>(); listE.addLast(E); listE.addLast(G);

LinkedList<Vertex> listF = new LinkedList<>(); listF.addLast(F); listF.addLast(H);

LinkedList<Vertex> listG = new LinkedList<>(); listG.addLast(G); listG.addLast(H);

LinkedList<Vertex> listH = new LinkedList<>(); listH.addLast(H); listH.addLast(I);

LinkedList<Vertex> listI = new LinkedList<>(); listI.addLast(I);