Given an adjacency list, how can you convert it to an adjacency matrix?

Explanation:

1. Get the length of the Adjacency List
2. Using the length from step 1, initialize an adjacency matrix of that size
3. Iterate through each vertex in the Adjacency List and update the edge values on the Matrix when an edge exists

Pseudocode:

```
convertAdjacencyListToMatrix (adjList):
    numVertices = adjList.length

    // in this case I will use a 2d array to represent the Adjacency Matrix
    adjMatrix = new int[numVertices][numVertices]

    For each vertice in adjList:
        For each edge in adjList[vertice]:
            adjMatrix[vertice][edge] = 1 //or true

Return adjMatrix
```

Given an adjacency matrix, how can you convert it to an adjacency list?

Explanation:

1. Get the number of vertices based on the size of the Adjacency Matrix
2. Initialize an Adjacency List based on the number of vertices
3. For each row, iterate through the columns of the Matrix
4. Add the values to the List based on the corresponding Matrix values

Pseudocode:

```
convertAdjMatrixToList(adjMatrix):
    numVertices = adjmatrix.size()
    adjList = new List[numVertices]

    For each vertex in adjMatrix:
        For each neighbor in adjMatrix[vertex]:
            if(adjMatrixt[vertex][neighbor] != 0)
                adjList.add(neighbor) @vertex,neighbor

    Return adjList
```

Given a directed graph, how can you reverse the direction of each edge?

Explanation:

1. Iterate through each edge in the graph
2. For each edge from “u” to “v” remove the edge
3. Add and edge from v to u
4. Repeat steps 2 and 3 for each edge in the graph

Pseudocode:

```
reverseEdges(graph):
    For each edge u to v in graph:
        removeEdge(u to v)
        addEdge(v to u)
Return graph
```