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

Example list:

- 1: [2, 3]
- 2: [4]
- 3: [4]
- 4: [5]
- 5: []

Goal Matrix:

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	0
3	0	0	0	1	0
4	0	0	0	0	1
5					

Explanation:

With this list, we first must create the 2 x 2 matrix based on the max value of the nodes, which would be a matrix from 0-5 indices. Then we must iterate through every list of every node. Starting at the first list, set the row # as a node's adjacency list, so it would be 1, then in the iteration, we use the vals in each adjList to set the col val as 1. Repeat this step until we have created a matrix.

This method would most likely have 2 for loops indicating an O(n^2) runtime.

Pseudocode:

Assume adjacency list is given in List<List<Integer>>

Public int [][] convertAdjMatrix(List<List<Integer>> adjList)

Int max = adjList.size() //Gives us the size of our matrix

```
Int[][] matrix = new [max][max] //Create our matrix

For (int i = 0; i < max; i++) //Loop through all lists

{

For (Integer val: adjList.get(i)) //Retrieve the integer values from the sublists

{

Matrix [i][val] = 1;

}

}

Return matrix;
```

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

Use previous example

	1	2	3	4	5
1	0	1	1	0	0
2	0	0	0	1	0
3	0	0	0	1	0
4	0	0	0	0	1
5	0	0	0	0	0

The approach is similar to the previous question, but in reverse. We are going to loop through the entire matrix and create sublists based on the matrix traversal. We will generate sublists on every row, add sublists to the adjList.

```
Public List<List<Integer>> adjList(int [][] adjMatrix)
ArrayList<ArrayList<Integer>> adjList = new …//Initalize list

For (int i = 0; i < adjMatrix[0].length; i ++)
{
ArrayList<Integer> temp = …
For (int j = 0; j < adjMatrix[0].length; j++
{
If adjMatrix[i][j] == 1;
temp.add(j);
}
adjList.add(temp)
}
Return adjList;
```

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

A graph can be represented by adjList. Let us assume that the directed graph is represented through an adjList.

- 1: [2, 3]
- 2: [4]
- 3: [4]
- 4: [5]
- 5: []

Explanation:

We want to create an adjList that looks like

- 1: []
- 2: [1]
- 3: [3]
- 4: [2,3]
- 5: [4]

We want to create a new list, alike to the given graph implementation. So we will be using the data structure List<List<Integer>>. We will initialize an empty version of the given graph, it will have the same amount of nodes. We will do a traversal on the given graph, but every value of the sublist we go to, we will add the values in reverse. Example: enter sublist of node 1, iterate at value 2, we will add the value of 1 to the 2nd sublist of the new graph.

```
Public List<List<Integer>> reverseGraph(List<List<Integer>> adjList)
```

```
{
List<List<Integer>> newList = .. //Initalize new graph based on size of adjList

For (int i = 0; i < adjList.size(); i++) //fill the new graph
{
newList.add(new ArrayList)
}

For (int i = 0; i < adjList.size(); i++) //Iterate through the sublists
{
        For (int j = 0; j < adjList.get(i).size(); i++) //iterate through val in sublist
        {
                Int val = adjList.get(j).get(i) //retrieve the value in adjList
                newList.get(val).add(i);
        }
}
Return newList;
```