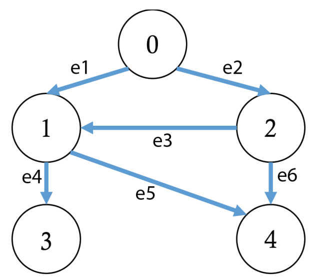
Implementation of Graph

Here we will discuss the very simple implementation of any graph. We will assume that

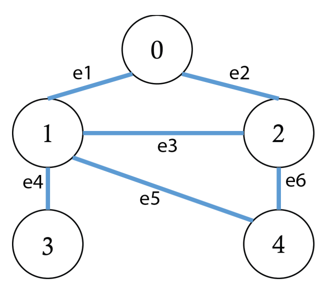
1. Number of vertices in the graph is **n**
2. The vertices are numbered from **0** to **n-1** uniquely
3. Number of edges in the graph is **e**
4. Then **e** number of edges will represent the structure of the graph. Each edge description contains two numbers: **i** and **j**, It means
   - <u>For a directed graph:</u> Vertex-**i** is connected to Vertex-**j**
   - <u>For a non directed graph:</u> Vertex-**i** is connected to Vertex-**j** and Vertex-**j** is also connected to Vertex-**i**

Let's see an example



GRAPH A

Graph **A** is a directed graph because each edge has a direction. The graph has a total **5** vertices, so **n=5** here. So you can see that, the vertices are numbered from **1** to **4**. Number of edges, **e**=**6** for this graph. Edges are:
e1 = 0 1; [As the graph is directed so we can't write **e1** as **1 0**, it will denote that, the edge is directed from **1** to **0**]
e2 = 0 2 e3 = 2 1 e4 = 1 3 e5 = 1 4 e6 = 2 4



GRAPH B

Graph **B** is a non directed graph because edges have no direction here. Graph **B** is similar as Graph **A**, just the edges have no direction. So, **n=5** and **e=6**. Edges are:
e1 = 0 1; [As the graph is non-directed so we can write **e1** as **1 0** also, because **e1** represents that **0** is connected to **1** and **1** is also connected to **0**]
e2 = 0 2 e3 = 2 1 e4 = 1 3 e5 = 1 4 e6 = 2 4

- **Representation of a graph in C++**

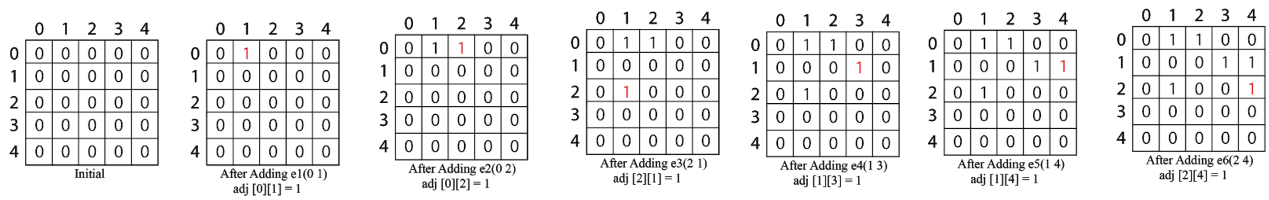
  There are 2 common ways to represent a graph in C++
  1. Adjacency Matrix (2D Array)
  2. Adjacency List (Array of Vector)

  We will discuss the concept of adjacency matrix only.

Adjacency matrix is basically a 2D array of **n** x **n** dimension. Previously we discussed that n is the number of vertices in the given graph. Let, the name of the matrix is **adj.** Initially all the elements of **adj** contain **0**. **0** represents No-connection. Initially we assume that there is no edge/connection in the graph. Gradually we develop the adjacency matrix when the edges are given as input one by one. The technique for developing an adjacency matrix is that:

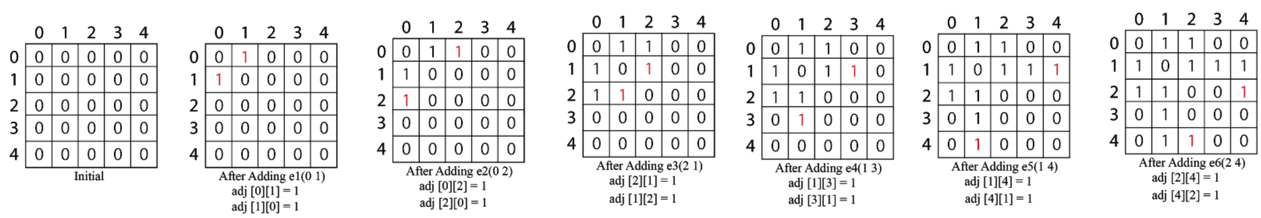
1. For a Directed Graph: If an edge is (**i j**), means **i** is connected to **j** but **j** is not connected to **i**, then we put adj[i][j] = 1. It represents that there is an edge from **i** to **j**. Source vertex (**i**) is represented as the first index and the destination vertex (**j**) is represented as the second index.
2. For a Non directed graph: If an edge is (**i j**), means **i** is connected to **j** and **j** is also connected to **i**, then we put adj[i][j] = 1 and also adj[j][i] = 1. I hope it needs no explanation.

Let's see example:



Adjacency Matrix of Graph A(Directed) step by step

.



Adjacency Matrix of Graph B(Non-Directed) step by step

Now lets do some coding for developing the adjacency matrix of a graph.

Input:

First input is **n**, number of vertices. Second input is **e**, number of edges. Third input is **d**, d=1 if the graph is directed or d=0 if the graph is non-directed. Next there will be **e** number of input lines each containing two integers representing the two vertices of an edge. For example if we consider Graph A, then the input section will be:

5 6 1
0 1
0 2
2 1
1 3
1 4
2 4

[Here n=5, e=6, d=1 as Graph A is directed. Then next **6** lines represent the edges from **e1** to **e6**]

Output:

Print the adjacency matrix of the given graph. For example if we print the adjacency matrix of Graph A it will be like:

0 1 1 0 0
0 0 0 1 1
0 1 0 0 1
0 0 0 0 0
0 0 0 0 0

Try it by yourself first. Congratulations if you have done it yourself. Otherwise, have a look [here]

- **Task 1:**

  First take the necessary inputs for a graph as described in the previous section. Now print the list of connectivity for every vertex of the graph. List of connectivity for a vertex-**i** represents a list of vertices that are connected with vertex-**i**. For example if we consider the Graph **B**, then the list of connectivity for each vertex will be:

  Vertex 0 is connected to: 1 2
  Vertex 1 is connected to: 0 2 3 4
  Vertex 2 is connected to: 0 1 4
  Vertex 3 is connected to: 1
  Vertex 4 is connected to: 1 2

  Again, try it by yourself first. Congratulations if you have done it yourself. Otherwise have a look [here]