CS29003 ALGORITHMS LABORATORY
ASSIGNMENT 4
Date: 8th Aug, 2019

Graph Representation

Adjacency List

A graph is a data structure that consists mainly of the following two sets:
1. A set V of vertices also called nodes (for our purposes this set is finite).
2. A subset E of ordered pair of vertices of the form (i, j) where i and j are nodes. If there is a direction involved then we call the graph a directed graph or a digraph.

Part-1

Construct an adjacency list from the user input of the graph. Input should be read from the file. **Do not use input redirection**.
Input format: First line of input consists of $|V|$ where $|V|$ is the number of vertices and each vertex is numbered from 1 to $|V|$. Second line of input consists of $|E|$ where $|E|$ is the number of edges in the graph. Next each of the $|E|$ lines consist of a pair of indices i j which implies there exists an edge from vertex i to vertex j. Structures for adjacency list are given below. Strictly use the structure given below.

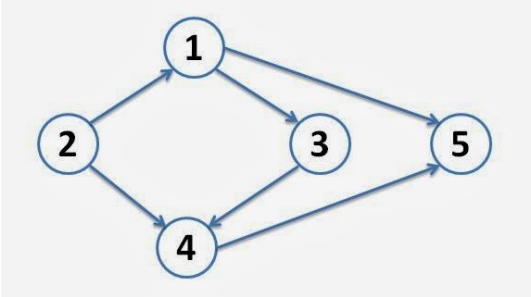
```
struct node
{
     int vertex;
     struct node* next;
};


struct Graph
{
     int numVertices;
     struct node** adjLists;
};
```

Part-2

Kivid is a young data structure enthusiast. After learning adjacency list, he comes up with another data structure for storing graphs. Since he has been out of practice from coding for a long time, he needs your help. Your marks will depend on how happy Kivid is with your implementation. More details are given below.
Since the number of edges are previously known, the edges can be stored in an array of Edge (structure given below, again strictly use this structure).

```
typedef
struct {
    int from;   // id of a source vertex
    int to;     // id of a destination vertex
    int weight; // weight of an edge
} Edge;
```

Weight of an edge will be the sum of the in-degree of both the source and destination vertex. In the graph given below, consider the edge from node 1 to node 5. In-degree of 1 is 1 and in-degree of 5 is 2. Therefore the weight of this edge is $1 + 2 = 3$.



After you have put the edges in the array, you need to sort the array of edges based on the following condition - sort on the basis of ascending order of source vertex. If the source vertex is same, sort on the basis of ascending order of weights. If they are also same, sort on the basis of ascending order of the destination vertex. Modify the **merge sort** for sorting the edges using this criteria. Output the sorted array with each element on a new line with 'from' printed first, then 'weight' and then 'to'. Sample input and output are given below. The output should be again printed on a file (**not through redirection**).

Sample Input - "input.txt"

```
5
6
2    1
2    4
1    3
1    5
3    4
4    5
```

Sample Output - "output.txt"

```
1    2    3
1    3    5
2    1    1
2    2    4
3    3    4
4    4    5
```

File Naming Convention

Please note that the output file names used in this document till now are generic and for explanation purpose only. **Your submissions will not be evaluated unless** you follow the below specified file naming convention for naming your files:

1. Program/Code file Naming Convention :
   <ROLLNO(IN CAPS)>_A<Assign_No>.c/cpp
   Eg: 18CS300019_A4.c / 18CS30019_A4.cpp

2. Output file Naming Convention :
   <ROLLNO(IN CAPS)>_A<Assign_No>_ output.txt
   Eg: 18CS300019_A4_output.txt