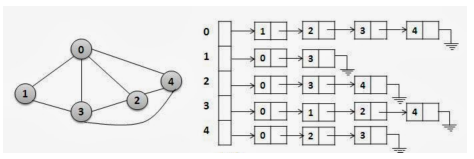
A Computer Network is stored in the following format: an array, called net, of pointers

where each pointer is the head of a list. The list at index i contains the ids of all the

computers to which computer i is connected; the order of the ids in a list does not matter.



A class which stores a network is defined below. Implement all the methods stated in

the following definition.

**class** Network

{

**struct** Computer{

**int** id;

Computer \* next;

//method to enable if(n[i][j]) cout<<"i and j are connected.";

**bool operator []** (**int** j);

};

vector<Computer\*> net; //built-in dynamic array

//add id into the list pointed to by head

**void addConnection**(Computer\*&head, **int** id);

**public**:

//for empty network

**Network**();

//read a network from a file

**Network**(string filename);

//deep copy methods

**Network**(**const** Network& obj);

**const** Network& **operator =**(**const** Network& obj);

//create net array of size, with no connections

**Network**(**int** size);

//connect computers x and y

//use the utility method addConnection

**void addConnection**(**int** x, **int** y);

//merge two networks (take union)

//computers, connections in any one of the networks appear in result

Network **operator +** (**const** Network& obj);

//intersect two networks (extract the common core)

//links and computers present in both networks appear in the result

Network **operator \*** (**const** Network& obj);

//Remove the common connections of obj and this network

Network **operator -** (**const** Network& obj);

//Take complement of the Network

//Returns a network with the same computers

//but which contains complementary connections

//resultant contains connections which are absent in this network

Network **operator -** ();

//print the network

**friend** ostream & **operator <<** (ostream & out, Network & obj);

//method to enable if(n[i][j]){cout<<"i and j are connected.";}

Computer & **operator []** (**int** i);

//add another computer to the network

Network **operator ++** (**int**);

//logical methods

//subNetwork returns true if obj is a sub-network of this network

**bool subNetwork**(**const** Network& obj);

//get all neighbors of computer nid

vector<**int**> **getNeighbors**(**int** nid);

//get all unique neighbors-of-neighbors of computer nid

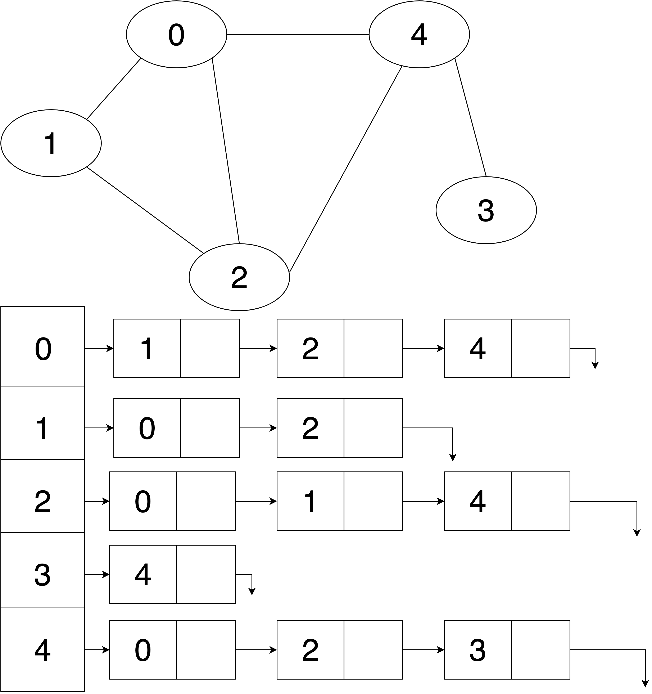
vector<**int**> **getNeighborsOfNeighbors**(**int** nid);

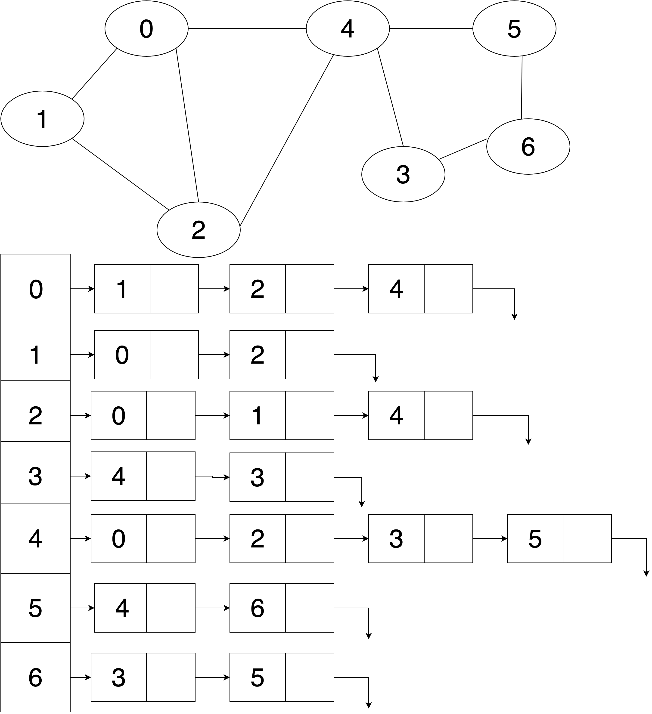
//de-allocate network

**~Network**();

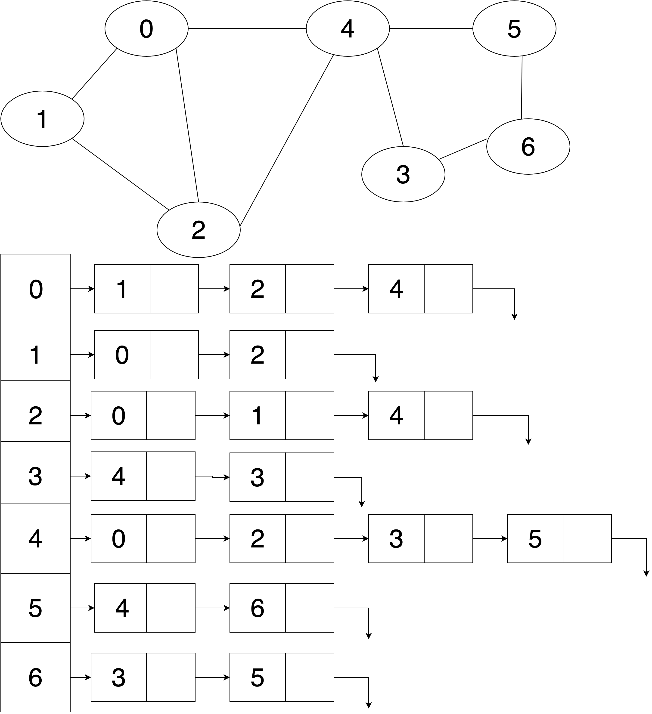
};

Example of union:





Taking union of these two networks result in the following:



Similarly, taking intersection will result in the following:

