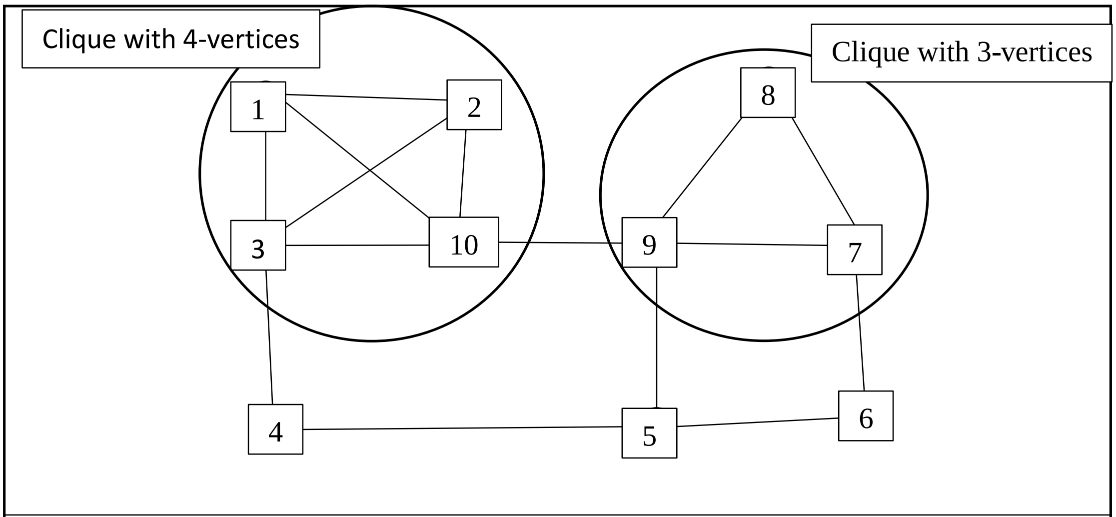
EE225
Assignment

Ajinkya Gorad
(140110033)

Aim :
Finding the 'Cliques' in a given Unidirected Graph using algorithm that uses brute force, and building a program (in C++) which reports the vertices of the Cliques(with number of vertices greater than 3).

Description :
A Clique is the Subgraph of the Unidirected graph that has a Complete Graph, i.e. all the vertices of a clique are connected to every other vertices by an edge. So for a clique with n vertices (nodes) , it will have $\frac{n(n-1)}{2}$ number of edges.

For Example



Given Unidirected Graph with 10 nodes and 15 edges

Assumptions:
Given Unidirected Graph is a connected graph with no isolated node. Represented in a suitably numerically representable format with nodes indexed from $1,2, \dots, N$, where N is the number of Nodes (strictly)

Algorithm:

1. Write the given graph in a suitable representable format.
   eg : int Graph[][2] ={ {1,2},{1,3},{1,10},{2,3},{2,10},{3,10},{3,4},{4,5},{5,6},
   {5,9},{6,7},{7,8},{7,9},{8,9},{10,9},{1,11},{11,12}};
   where each of the braces represent the edge connection.
2. Pick i^{th} node
3. Find all adjacent nodes for i^{th} node

4. Check all possible combinations if they are clique or not (with minimum 3 nodes , to avoid edges being interpreted as cliques)
    a. For checking whether it is a clique or not , every edge connection is checked.
5. If found a clique, report it.
6. Else go for $i + 1^{th}$ node (follows to step 2).
   Order of this algorithm can be roughly exponential, as it checks all the combinations, but not exactly.

Program Code:

- Written in C++ with compiler TDM-GCC 4.8.1 64-bit Release (Dev C++)

```
#include <iostream>
#include <stdlib.h>
#include <vector>
#include <algorithm>
using namespace std;


//enter the Unigraph here
// each bracket represents the connected set of edges
//----------------------------------
int Graph[][2] ={ {1,2},{1,3},{1,10},{2,3},{2,10},{3,10},{3,4},{4,5},{5,6},

        {5,9},{6,7},{7,8},{7,9},{8,9},{10,9},{1,11},{11,12}};
//----------------------------------


void dispVector(vector<int> v)
{
        cout<<"Vector("<<v.size()<<") : ";
        cout<<"("<<v[0];
        for(int i =1;i<v.size();i++)
        {
                cout<<","<<v[i];
        }
        cout<<")";
}


int getNumberOfEdges()
{
        return sizeof(Graph)/sizeof(int)/2;
}
int getNumberOfNodes()
{
        int nE = getNumberOfEdges();
        int maxNum = 0;
        for(int i=0;i<nE;i++)
        {
                if(Graph[i][0]>maxNum)maxNum=Graph[i][0];
```

```
			if(Graph[i][1]>maxNum)maxNum=Graph[i][1];
		}
		return maxNum;
	}
	bool checkIfConnected(int node1, int node2)
	{
		int nE = getNumberOfEdges();
		for(int i=0;i<nE;i++)
		{
			if(Graph[i][0]==node1)if(Graph[i][1]==node2)return true;
			if(Graph[i][1]==node1)if(Graph[i][0]==node2)return true;
		}
		return false;
	}

	vector<int> getAdjacentNodes(int initialNode)
	{
		vector<int> adjacentNodes;
		int nE = getNumberOfEdges();
		for(int i=0;i<nE;i++)
		{

		if(Graph[i][0]==initialNode)adjacentNodes.push_back(Graph[i][1]);
			else if
	(Graph[i][1]==initialNode)adjacentNodes.push_back(Graph[i][0]);
		}
		return adjacentNodes;   //  returns adjacent Nodes
	}
	bool checkIfClique(vector<int> nodes)
	{
		for(int i=0;i<nodes.size()-1;i++)
		{
			for(int j=i+1;j<nodes.size();j++)
			{
				if(!checkIfConnected(nodes[i],nodes[j]))return false;
			}
		}
		return true;
	}
	void getClique(int node)		// prints the clicque at the corresponding node
	{


		vector<int> adjNodes = getAdjacentNodes(node);
		adjNodes.push_back(node);	// also add the original node
		if(adjNodes.size()<3)// all cliques less than 3 nodes are either edges or
	nodes itself
		{
			cout<<"\n\r\t Edge Clique";
			return;
		}
		//brute Force
```

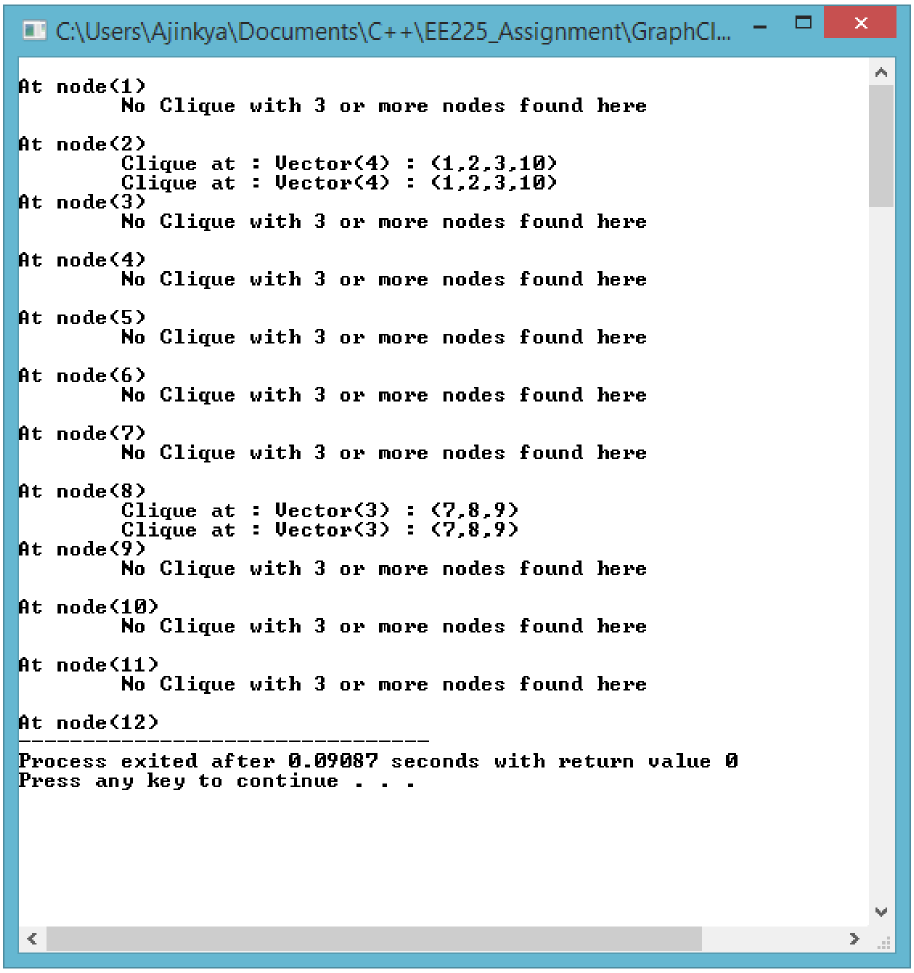
```
		int N = adjNodes.size();
		bool foundClique = false;
		vector<bool> v(N);
		vector<int> testNodes;
		for(int choose=1;choose<=N;choose++)
		{
			foundClique=false;
			fill(v.begin() +N-  choose, v.end(), true);
			testNodes.clear();
			do {
				for(int i=0; i<N; i++)
				{
				if(v[i])testNodes.push_back(adjNodes[i]);
				}
			} while ( next_permutation(v.begin(), v.end()));

			if(checkIfClique(testNodes))
			{
				sort(testNodes.begin(),testNodes.end());
				cout<<"\n\r\t*Clique at : ";dispVector(testNodes);
				foundClique = true;
			}

		}
		if(foundClique==false)
		{
			cout <<"\n\r\tNo Clique with 3 or more nodes found here "
	<<endl;
		}
	}
	int main()
	{
		for(int i=1;i<=getNumberOfNodes();i++)
		{
			cout<<"\n\rAt node("<<i<<")  ";
			getClique(i);
		}
	}
```

Results :

- Snapshot of the output from the program for given graph with 10 nodes.



C:\Users\Ajinkya\Documents\C++\EE225_Assignment\GraphCl...

```
At node(1)
        No Clique with 3 or more nodes found here

At node(2)
        Clique at : Vector(4) : (1,2,3,10)
        Clique at : Vector(4) : (1,2,3,10)
At node(3)
        No Clique with 3 or more nodes found here

At node(4)
        No Clique with 3 or more nodes found here

At node(5)
        No Clique with 3 or more nodes found here

At node(6)
        No Clique with 3 or more nodes found here

At node(7)
        No Clique with 3 or more nodes found here

At node(8)
        Clique at : Vector(3) : (7,8,9)
        Clique at : Vector(3) : (7,8,9)
At node(9)
        No Clique with 3 or more nodes found here

At node(10)
        No Clique with 3 or more nodes found here

At node(11)
        No Clique with 3 or more nodes found here

At node(12)
--------------------------------
Process exited after 0.09087 seconds with return value 0
Press any key to continue . . .
```

Applications:

- 'Cliques' can be used to reduce the amount of space required to store the graph.
  For eg : Graph with ={ {1,2},{1,3},{1,10},{2,3},{2,10},{3,10},{3,4},{4,5},{5,6}, {5,9},{6,7},{7,8},{7,9},{8,9},{10,9}} can be represented as combinations of 8 cliques as {1,2,3,10},{7,8,9},{9,10},{3,4},{4,5},{5,9},{5,6},{6,7} of which , one 4-vertex clique, one 3-vertex clique, and six 2-vertex clique.
- Can be used to reduce the network to clusters which are highly interlinked (in cliques fashion) .

Links :

Project code can be found at github repository *https://github.com/ajinkyagorad/MyC/tree/master/EE225_Assignment*

Resources: https://en.wikipedia.org/wiki/Clique_%28graph_theory%29 for understanding of cliques