EAST WEST UNIVERSITY

Group Project

Group no: 04

Project no : 09

Project Name: Clique Problem

Course Title : Algorithm

Course code : CSE246

Sec : 02

Submitted To:

Redwan Ahmed Rizvee

Lecturer, Department of Computer Science and Engineering

East West University

Submitted By:

Name	ID
Fahad Ahamed	2020-2-60-174
Md Tahsin	2020-2-60-112
Navid Zaman	2020-2-60-044

1. Problem Statement:

In this following term we have to find all the size of cliques found in the given graph. And also clique's information in separate lines according to their ascending order of ids. Common formulations of the clique problem include finding a maximum clique (a clique with the largest possible number of vertices), finding a maximum weight clique in a weighted graph, listing all maximal cliques (cliques that cannot be enlarged), and solving the decision problem of testing whether a graph contains a clique larger than a given size.

2. Algorithm Discussion:

To solve this problem, In our algorithm, We have calculated degrees of all the nodes and kept in array. The main logic is we will check all the vertices having degree equal or greater than to clique_size. After finding the vertex we will check if the following vertices form a subgraph or not. We took three parameters in clique function which are start, value, clique size. Here start parameter represents the starting node from which we will start iterating , value represents the number of nodes are in the keep array and third parameter is represents the size of the click we want to find. Then we will start checking from the starting node if the degree is greater or equal to clique_size - 1. If condition satisfies, we store the node in keep array. After storing the nodes we check if those nodes form a clique or not. If those make a clique then we check the size of clique is equal clique_size or not. If it forms a clique and the number of vertices in the store is equal to the required size of the clique then we store the clique in a vector caller store_clique. If it

```
1.  Clique(int start, int value, int clique_size)
2.  {
3.       limit= v - (clique_size - value)
4.       FOR i = start to limit
5.            IF degree of i >= clique_size-1
6.              keep[value] = i
7.            FOR j=1 to value+1
8.               FOR j+1 to value+1
9.                  Check connection
10.              END IF
11.
12.    IF connected_node= true
13.           IF value = clique_size
14.             store to store_clique
15.           End IF
16.      End IF
17.      IF value < clique_size
18.           Clique( i, value+1, clique_size)
19.      End IF
20.     End IF
21.    End FOR
22. }
```

doesn't form a clique and the number of elements in the store array is less than clique_size, then we call the function clique recursively and increase the number of elements in the store.

3. Complexity Analysis:

Time Complexity

```
1.  Clique(int start, int value, int clique_size)
2.  {
3.       limit= v - (clique_size - value)
4.       FOR i = start to limit     O(V)
5.            IF degree of i >= clique_size-1
6.              keep[value] = i
7.            FOR j=1 to value+1     O(K)
8.               FOR j+1 to value+1     O(K)
9.                  Check connection
10.              END IF
11.
12.    IF connected_node= true
13.           IF value = clique_size
14.             store to store_clique
15.           End IF
16.       End IF
17.       IF value < clique_size
18.       Clique( i, value+1, clique_size) O(K)
19.       End IF
20.      End IF
21.    End FOR
22. }
```

Assuming clique size=K , number of vertex =V ,number of edge=E ,

In the clique checking block, the total running time is $O(k^2E)$. And in the overall clique function, the total function is implemented inside of the clique function, so the total running is $O(k^2E * V^k)$.

And the memory complexity is O(V+E*k).

4.Implementation:

Clique Function :

```
void clique(int start,int value ,int clique_size ){  //clique checking function
   int limit=v-(clique_size-value);  //find minimum number of nodes for which we need to check clique
   for(int  i=start;i<=limit;i++){
         if(degree[i]>=clique_size-1){  //checking degree size
            keep[value]=i;
            bool connected_node=true;
            for(int j=1;j<value+1;j++){
               for(int k=j+1;k<value+1;k++){
                  if(graph[keep[j]][keep[k]]==0){   //checking if its a click
                    connected_node=false;
                    break;
               }}}
            if(connected_node){ //if number of nodes is greater than or equal clique_size
                if(value<clique_size){
                    int xy=value+1;
                    clique(i,xy,clique_size);
                }
                else{
                vector <int> tmp;  //a temporary vector to store cliques
                cnt++;
                for(int l=1;l<value+1;l++){
                    tmp.push_back(keep[l]);
                }
                store_clique.push_back(tmp);
               tmp.clear();
                }
            }}}}
```

Here,

v → number of vertices.

start→ starting node.

Value → the number of vertices currently in the keep[] array.

Clique_size → required size of the cliques.

In clique function , it is checked that if any node has adjacent vertex of given clique size .Then it checks if a clique can be formed of given size .If its possible to form a clique of given size than it checks all the combination of possible clique and store in the vector store_clique .

Check connected or not :

```
for (int j = 1; j < value + 1; j++)
{
    for (int k = j + 1; k < value + 1; k++)
    {
        if (graph[keep[j]][keep[k]] == 0)// checking if its a click
        {
            connected_node = false;
            break;
        }
    }
}
```

It is basically a part of clique function . In this block of code, it is checked that if a node is connected with the possible clique nodes or not. If any node found which is not connected than the connected_node becomes false .

Storing Cliques:

```
vector<int> tmp; // a temporary vector to store cliques
cnt++;
for (int l = 1; l < value + 1; l++)
{
    tmp.push_back(keep[l]);
}
store_clique.push_back(tmp);
tmp.clear();
```

It is also a part of clique function. In this block of code , we store the nodes of cliques in a temporary vector and pushes to another 2D vector called store_clique .

Printing cliques:

```
for (int start = 0; start < cnt; start++) // printing cliques
{
    for (int i = 0; i < store_clique[start].size(); i++)
    {
        cout << store_clique[start][i] << " ";
    }
    cout << endl;
}
```

This block of code is a part of main function .Here we print all the nodes of particular clique.

5.Application

1 .Here we can make an application of ‘Tour Guide’. It will work between two cities where user can input places and they can also return where they started from. For an example: a group of tourists wants to visits between Dhaka to Chittagong. So the application will show suitable places where they can visits as well as they can return back. In that case it’s a complete graph and it is clear that, clique graph can be used for this application.

2. One of the common implementations of a clique is a social network where the vertices of the clique graph represents people and the edges of that graph of that clique graph represent mutual relation among them. The clique graph represents a group of mutual friends where all of them know each other.