IOWA STATE UNIVERSITY

CPR E 525 Final Project

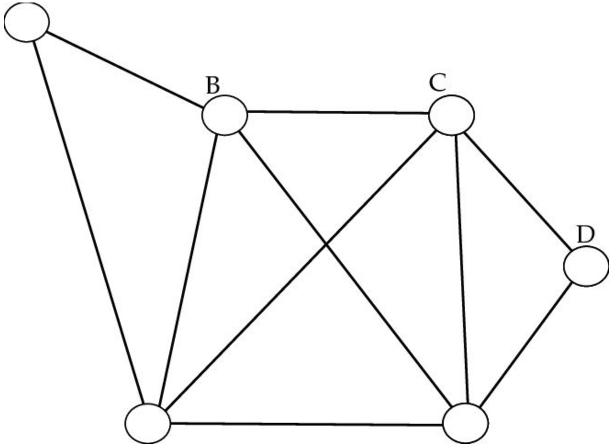
FINDING THE NUMBER OF CLIQUES IN A GRAPH

SANJITA SALUNKHE
#774512756

Cliques in a graph

- a subset of vertices in an undirected graph
- every 2 distinct vertices are adjacent

(subgraph in a graph connected by an edge)



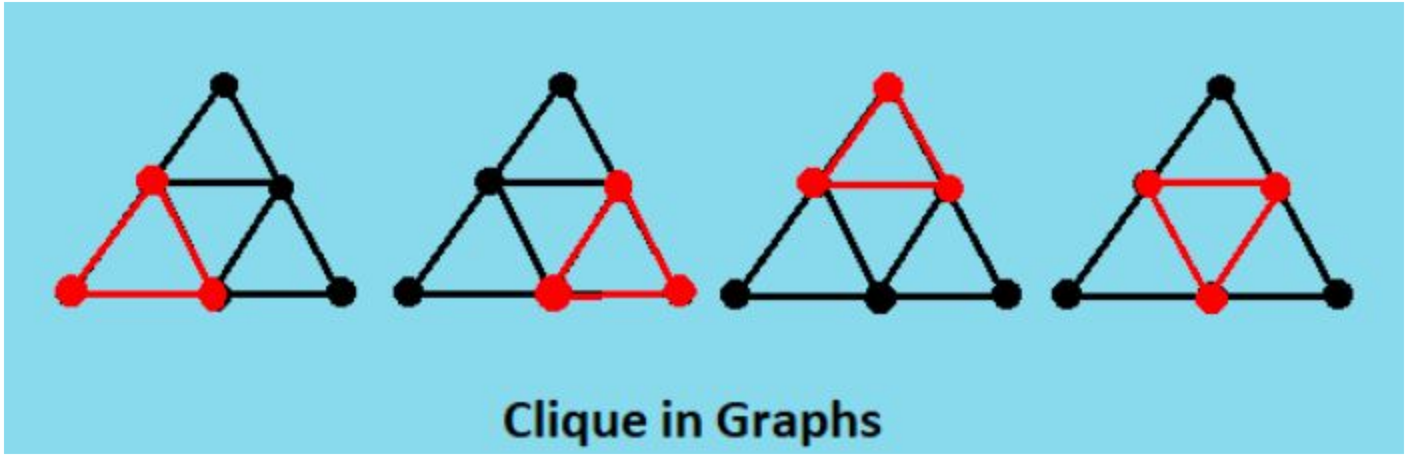


Image reference - https://iq.opengenus.org/clique-in-graphs/

What is the clique problem?

- computational problem of finding cliques in a graph
- has several different formulations depending on which cliques, and what information about the cliques, should be found.

<u>Common formulations of the clique problem:</u>

- finding a maximum clique,
- finding a maximum weight clique in a weighted graph,
- listing all maximal cliques, and
- solving the decision problem of testing whether a graph contains a clique larger than a given size.

Reference - https://iq.opengenus.org/clique-in-graphs/

Uses

- data mining
- mathematical problems
- construction of graphs
- spam detection

Uses

Data mining

- elements of a system and their relationships are modeled as a graph
- graph based data mining is useful for studying unstructured/partially structured data
- cliques are used to identify “tightly knit” clusters
- this provides insights into a variety of different application settings

Reference - Balasundaram B., Pajouh F.M. (2013) Graph Theoretic Clique Relaxations and Applications. In: Pardalos P., Du DZ., Graham R. (eds) Handbook of Combinatorial Optimization. Springer, New York, NY. https://doi.org/10.1007/978-1-4419-7997-1_9

Applications

- Internet graph
- Call graph
- Stock market graphs
- Social network
    - Social network analysis
- Biological networks
    - Protein interaction networks

Reference - Balasundaram B., Pajouh F.M. (2013) Graph Theoretic Clique Relaxations and Applications. In: Pardalos P., Du DZ., Graham R. (eds) Handbook of Combinatorial Optimization. Springer, New York, NY. https://doi.org/10.1007/978-1-4419-7997-1_9

Applications of clique (Examples)

- Consider a social network, where the graph's vertices represent people, and the graph's edges represent mutual acquaintance. Then a **clique represents a subset of people who all know each other**, and algorithms for finding cliques can be used to **discover these groups of mutual friends**.
- Bioinformatics - Ben-Dor, Shamir & Yakhini (1999) model the **problem of clustering gene expression data as one of finding the minimum number of changes needed to transform a graph describing the data into a graph formed as the disjoint union of cliques**.
- Electrical engineering - Prihar (1956) **uses cliques to analyze communications networks**
- Paull & Unger (1959) use them to **design efficient circuits for computing partially specified Boolean functions**.
- Computational chemistry - Kuhl, Crippen & Friesen (1983) **use cliques to model the positions in which two chemicals will bind to each other**.

Reference - https://iq.opengenus.org/clique-in-graphs/

Algorithm used

Clique of fixed size

One can test whether a graph G contains a k-vertex clique, and find any such clique that it contains, using a brute force algorithm.

- This algorithm examines each subgraph with k vertices and checks to see whether it forms a clique.
- the problem may be solved in polynomial time whenever k is a fixed constant.
- when k does not have a fixed value, but instead may vary as part of the input to the problem, the time is exponential.

Reference - https://iq.opengenus.org/clique-in-graphs/

How the program works

Input?

An undirected binary weighted graph -

The first line contains two integers N and E,

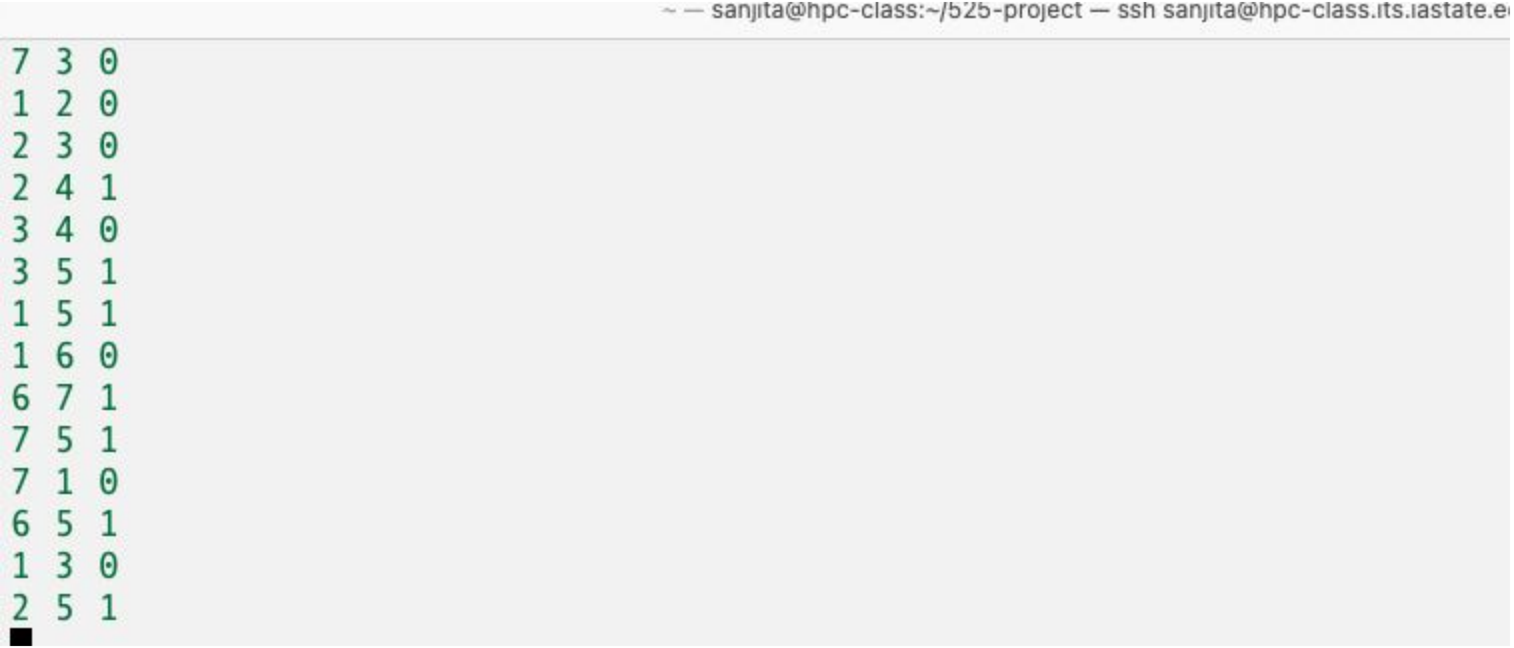
N - the number of vertices

E - the number of edges in the graph

E lines contain 3 integers (2 vertices and a weight) each, representing an edge between these two vertices with a binary weight (0 or 1)

<u>To find</u> - the number of cliques of sizes 3 and 4

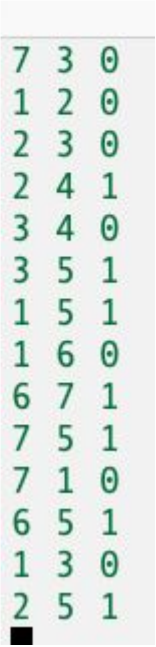
Program input



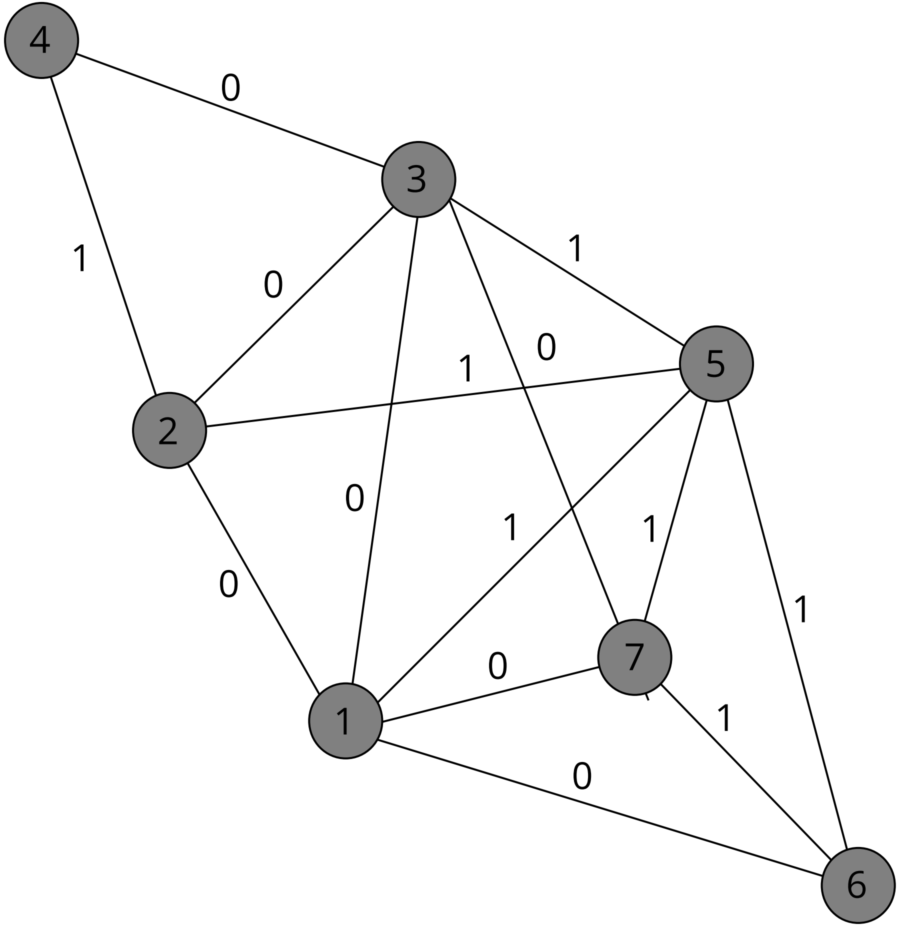
~ — sanjita@hpc-class:~/525-project — ssh sanjita@hpc-class.its.iastate.e

```
7 3 0
1 2 0
2 3 0
2 4 1
3 4 0
3 5 1
1 5 1
1 6 0
6 7 1
7 5 1
7 1 0
6 5 1
1 3 0
2 5 1
```

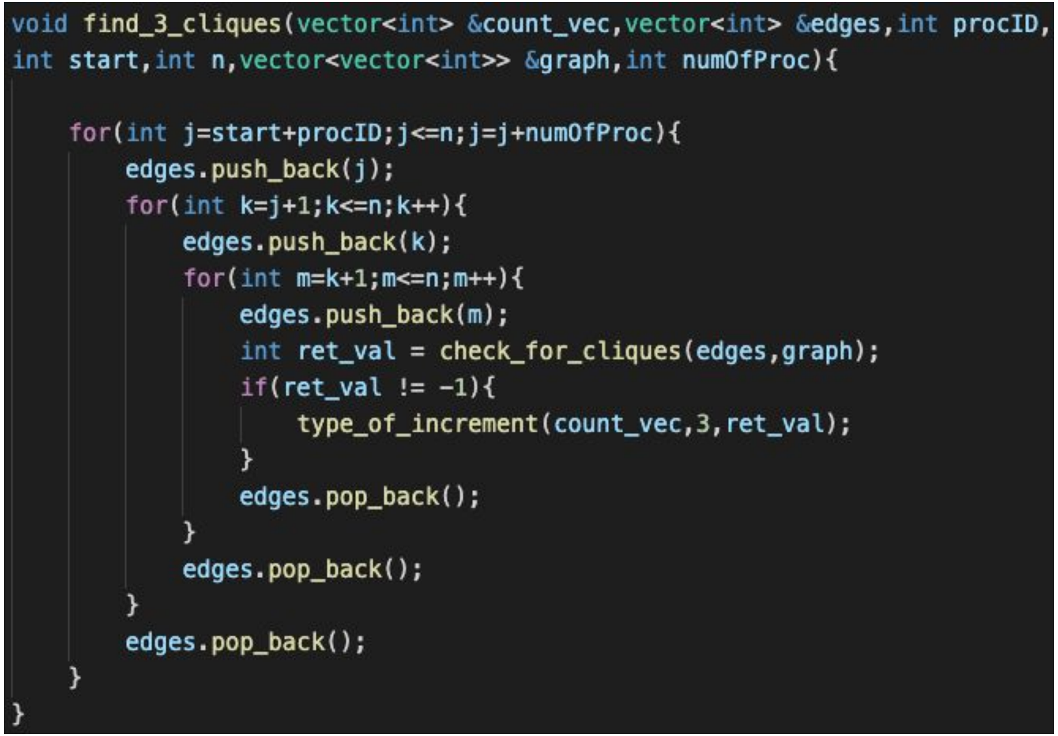
Graph of the Input



```
7 3 0
1 2 0
2 3 0
2 4 1
3 4 0
3 5 1
1 5 1
1 6 0
6 7 1
7 5 1
7 1 0
6 5 1
1 3 0
2 5 1
```

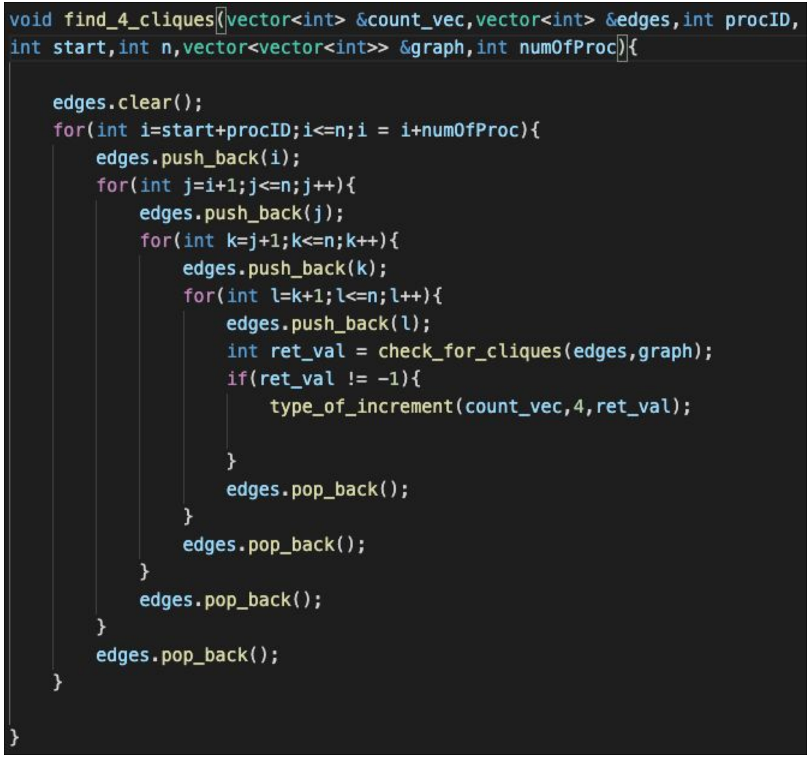


MPI parallelization for a clique of size 3



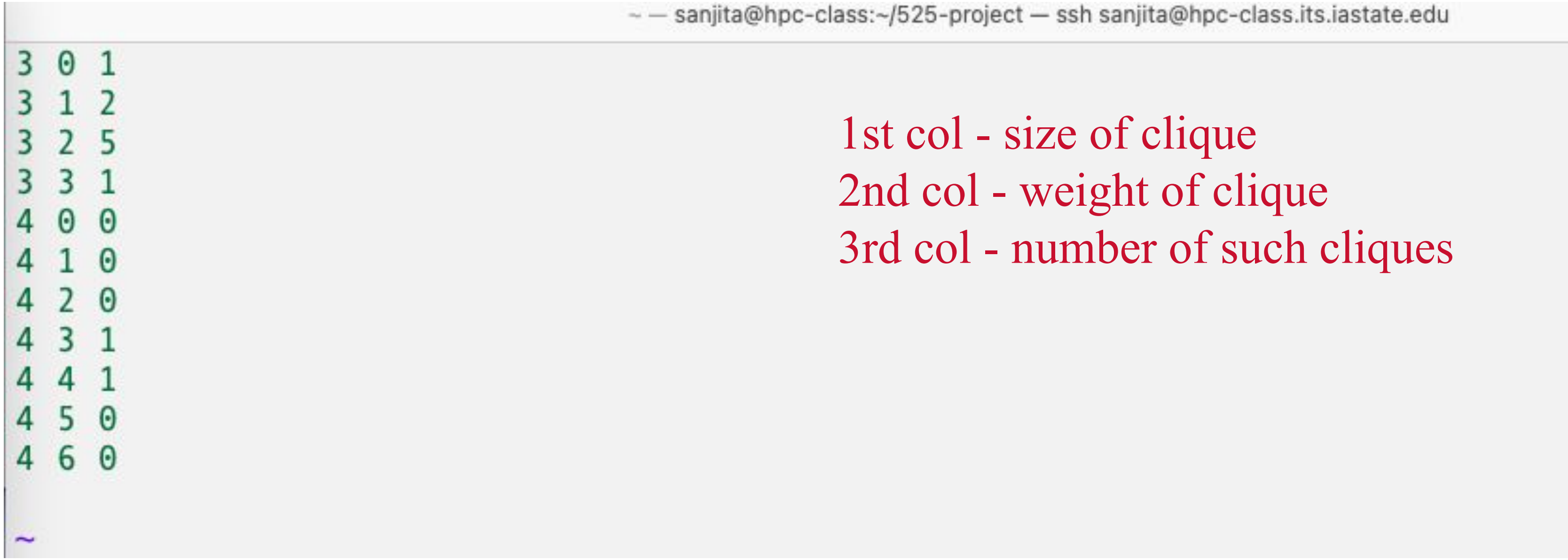
```
void find_3_cliques(vector<int> &count_vec,vector<int> &edges,int procID,
int start,int n,vector<vector<int>> &graph,int numOfProc){

    for(int j=start+procID;j<=n;j=j+numOfProc){
        edges.push_back(j);
        for(int k=j+1;k<=n;k++){
            edges.push_back(k);
            for(int m=k+1;m<=n;m++){
                edges.push_back(m);
                int ret_val = check_for_cliques(edges,graph);
                if(ret_val != -1){
                    type_of_increment(count_vec,3,ret_val);
                }
                edges.pop_back();
            }
            edges.pop_back();
        }
        edges.pop_back();
    }
}
```

MPI parallelization for a clique of size 4



```
void find_4_cliques(vector<int> &count_vec,vector<int> &edges,int procID,
int start,int n,vector<vector<int>> &graph,int numOfProc){

    edges.clear();
    for(int i=start+procID;i<=n;i = i+numOfProc){
        edges.push_back(i);
        for(int j=i+1;j<=n;j++){
            edges.push_back(j);
            for(int k=j+1;k<=n;k++){
                edges.push_back(k);
                for(int l=k+1;l<=n;l++){
                    edges.push_back(l);
                    int ret_val = check_for_cliques(edges,graph);
                    if(ret_val != -1){
                        type_of_increment(count_vec,4,ret_val);

                    }
                    edges.pop_back();
                }
                edges.pop_back();
            }
            edges.pop_back();
        }
        edges.pop_back();
    }

}
```

Program output



~ — sanjita@hpc-class:~/525-project — ssh sanjita@hpc-class.its.iastate.edu

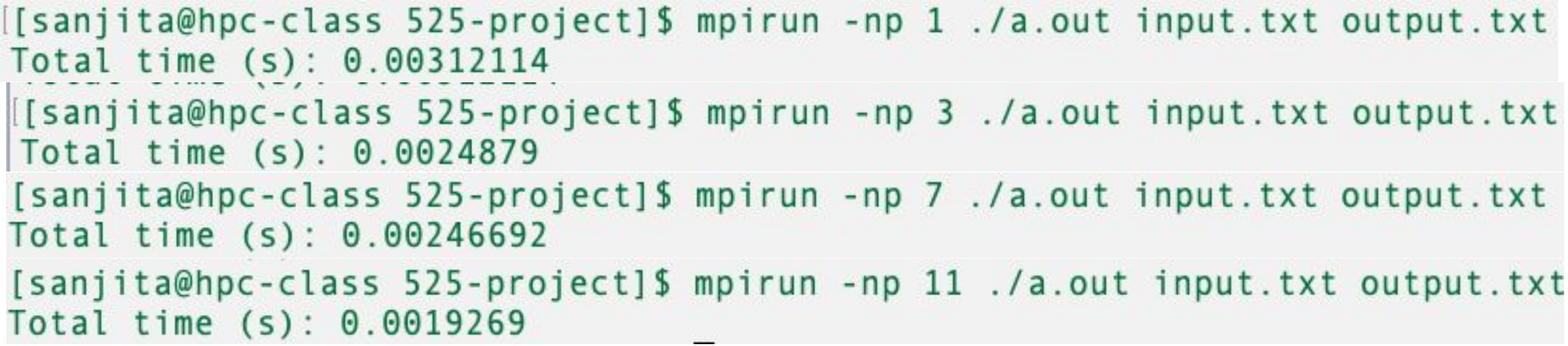
```
3 0 1
3 1 2
3 2 5
3 3 1
4 0 0
4 1 0
4 2 0
4 3 1
4 4 1
4 5 0
4 6 0

~
```

1st col - size of clique

2nd col - weight of clique

3rd col - number of such cliques

Program output



```
[[sanjita@hpc-class 525-project]$ mpirun -np 1 ./a.out input.txt output.txt
Total time (s): 0.00312114
[[sanjita@hpc-class 525-project]$ mpirun -np 3 ./a.out input.txt output.txt
Total time (s): 0.0024879
[sanjita@hpc-class 525-project]$ mpirun -np 7 ./a.out input.txt output.txt
Total time (s): 0.00246692
[sanjita@hpc-class 525-project]$ mpirun -np 11 ./a.out input.txt output.txt
Total time (s): 0.0019269
```

Future Work

Extensions to current project

When dealing with bigger graphs -

- Can find the maximal cliques
- Can work on increasing the speed

Any ideas on what else could be done?

References

[1] Balasundaram B., Pajouh F.M. (2013) Graph Theoretic Clique Relaxations and Applications. In: Pardalos P., Du DZ., Graham R. (eds) Handbook of Combinatorial Optimization. Springer, New York, NY. https://doi.org/10.1007/978-1-4419-7997-1_9

[2] https://mathworld.wolfram.com/Clique.html

[3] https://en.wikipedia.org/wiki/Clique_(graph_theory)

[4] http://maheshgadgilsblog.blogspot.com/2011/11/why-clique-problem-is-important-think.html

[5] https://en.wikipedia.org/wiki/Clique_problem

[6] https://iq.opengenus.org/clique-in-graphs/

[6] Gianinazzi, Lukas, et al. "Parallel Algorithms for Finding Large Cliques in Sparse Graphs." *Proceedings of the 33rd ACM Symposium on Parallelism in Algorithms and Architectures*. 2021.

[7] Szabó, S. "Parallel algorithms for finding cliques in a graph." *Journal of Physics: Conference Series*. Vol. 268. No. 1. IOP Publishing, 2011.

[8] Cheng, James, et al. "Finding maximal cliques in massive networks." *ACM Transactions on Database Systems (TODS)* 36.4 (2011): 1-34.

[9] Wang, Junjie, Shuigeng Zhou, and Jihong Guan. "Detecting potential collusive cliques in futures markets based on trading behaviors from real data." *Neurocomputing* 92 (2012): 44-53.

[10] G. Hua, H. Liao, H. Zhang, D. Ye and J. Ma, "Robust ENF Estimation Based on Harmonic Enhancement and Maximum Weight Clique," in IEEE Transactions on Information Forensics and Security, vol. 16, pp. 3874-3887, 2021, doi: 10.1109/TIFS.2021.3099697.

[11] Hahn, Klaus, Peter R. Massopust, and Sergei Prigarin. "A new method to measure complexity in binary or weighted networks and applications to functional connectivity in the human brain." *BMC bioinformatics* 17.1 (2016): 1-18.