DAA Report Group-19

Team Members:

Shivansh Shanker Gupta

2022A7PS0047H

Sidharth Saxena

2021B4A72488H

T.Sai Sathwik

2020B4A71567H

G Sri Vishwahitha

2022A7PS0041H

Snigdha Kaipa

2022A7PS2014H

(I) Report on Clique-Finding Algorithms:

In an undirected graph, a clique is a maximal complete subgraph, which means that each vertex is connected to every other vertex in the subgraph. Finding all cliques in a graph is a basic topic in graph theory, with applications in social network research, biology, and optimization.

This study focuses on efficient algorithms for maximal clique enumeration, including :

1)Chiba-Nishizeki's Arboricity-Based Algorithm

2)Tomita's Branch-and-Bound Algorithm

3)Bron-Kerbosch Algorithm with Pivoting

1. Chiba-Nishizeki’s Arboricity-Based Algorithm

Time complexity = O(a(G) \* m) Per clique Best for: sparse graphs. Chiba and Nishizeki's (1985) technique increases clique enumeration by using arboricity (a(G)), a measure of edge density in a graph.

The fundamental idea is to process vertices in decreasing degree order to minimize redundant labor. • Dynamically update the adjacency list to prevent duplicate counting. Why is it useful? This approach has a linear time complexity of O(n) for planar graphs with a(G) < 3.

1. Tomita’s Depth-First Search (DFS) Algorithm

Time complexity = O(3^(n/3)). (Worst-case optimum) Best for general graphs. Tomita et al. (2006) improved Bron-Kerbosch by implementing a branch-and-bound pruning approach.

1. Extend only promising branches in a DFS tree.

2. Utilize vertex orderings to minimize duplicate searches.

3. Store intermediate findings in a tree structure to save space. This algorithm is asymptotically optimal, although it remains exponential in the worst-case scenario.

3) Bron-Kerbosch Algorithm (with Pivoting)

Time complexity = O(3^(n/3)). (Worst-case optimum) Best for small to medium graphs. The Bron-Kerbosch algorithm (1973) is a backtracking algorithm that recursively creates cliques.

The optimized pivoting technique finds a vertex that reduces search space, considerably boosting performance

Steps:

1. Begin with an empty clique R and extend by adding nearby vertices.

2. Maintain three sets. • Current clique (R) • Possible extensions (P) • Excluded nodes (X) to prevent duplicates.

3. Select a pivot vertex (u) to limit recursive calls. Why it works: • In random graphs, performance is nearly linear per clique. • The pivoting method minimizes recursive calls.

| **Algorithm** | **Best for** | **Time Complexity** |
| --- | --- | --- |
| Chiba-Nishizeki | Sparse graphs | O(a(G) \* m) per clique |
| Tomita DFS | General graphs | O(3^(n/3)) |
|  |  |  |
| Bron-Kerbosch (Pivoting) | Medium-sized graphs | O(3^(n/3)) |

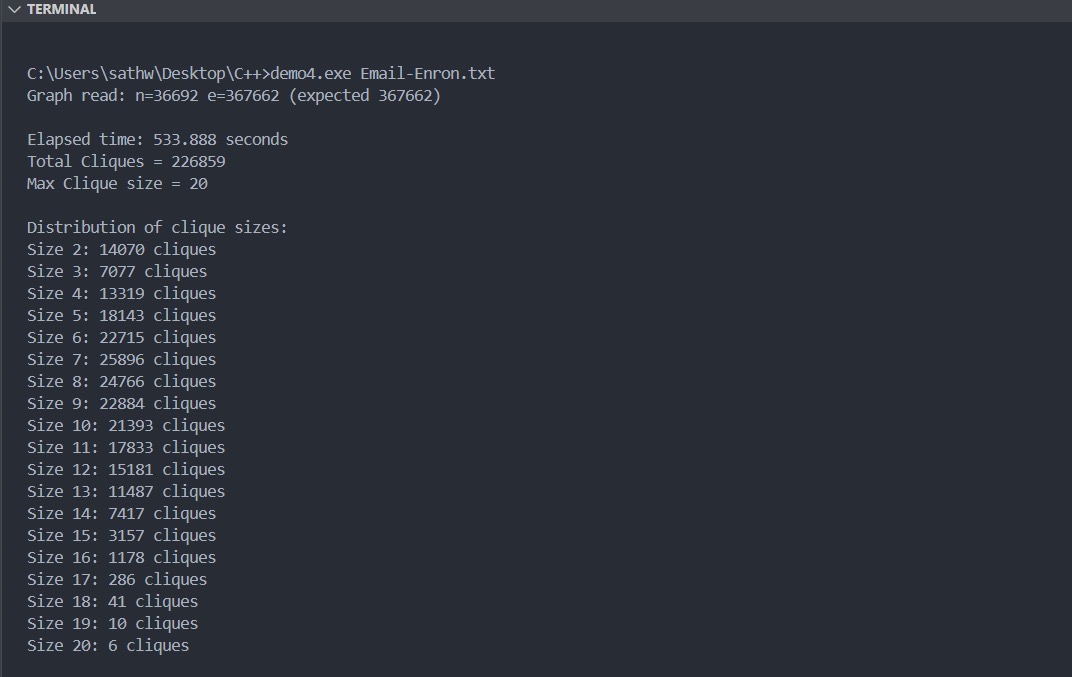
Elapsed Time Comparison :

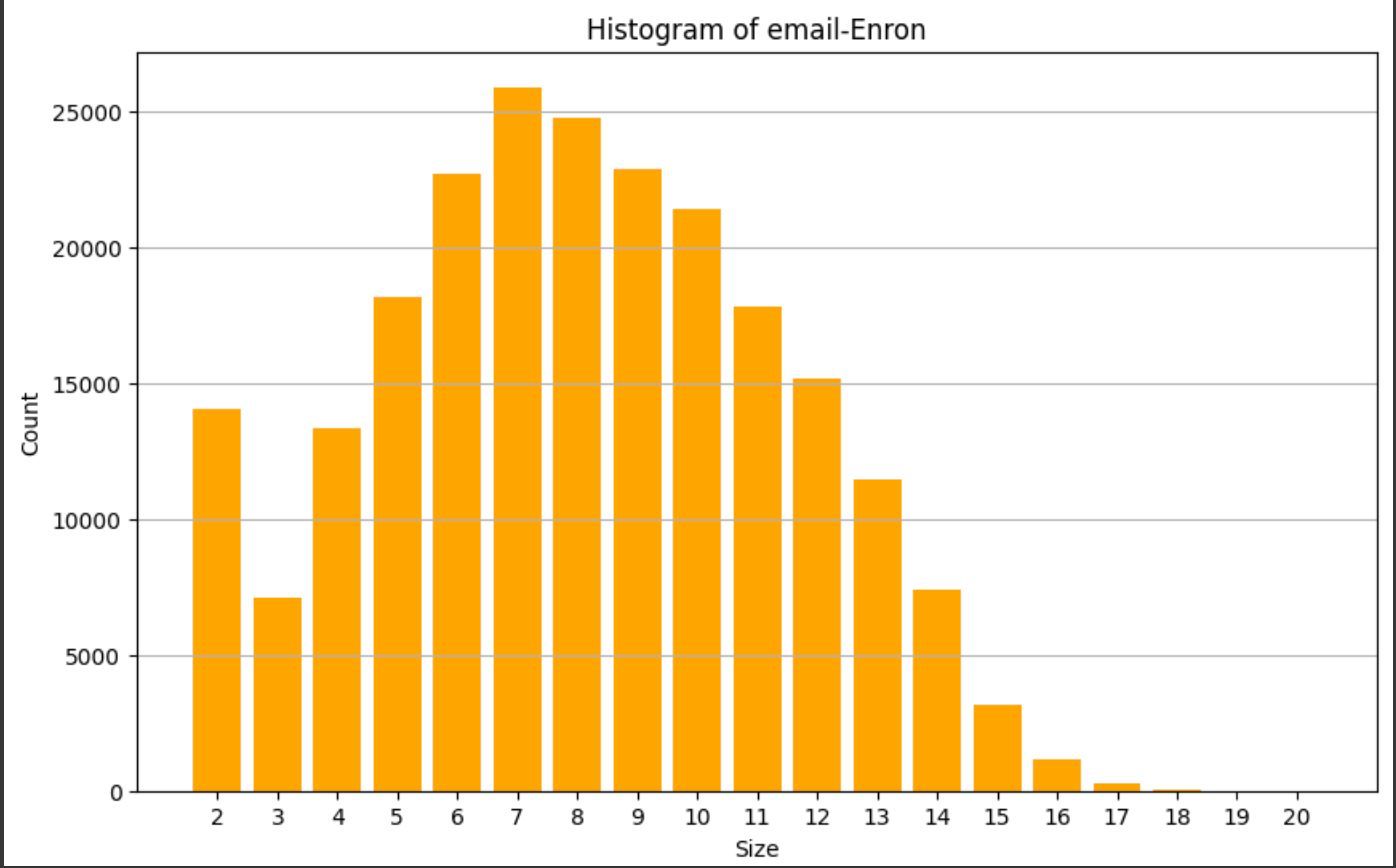
Email-Enron- 533.888 sec

As-Skitter- 2245052 ms

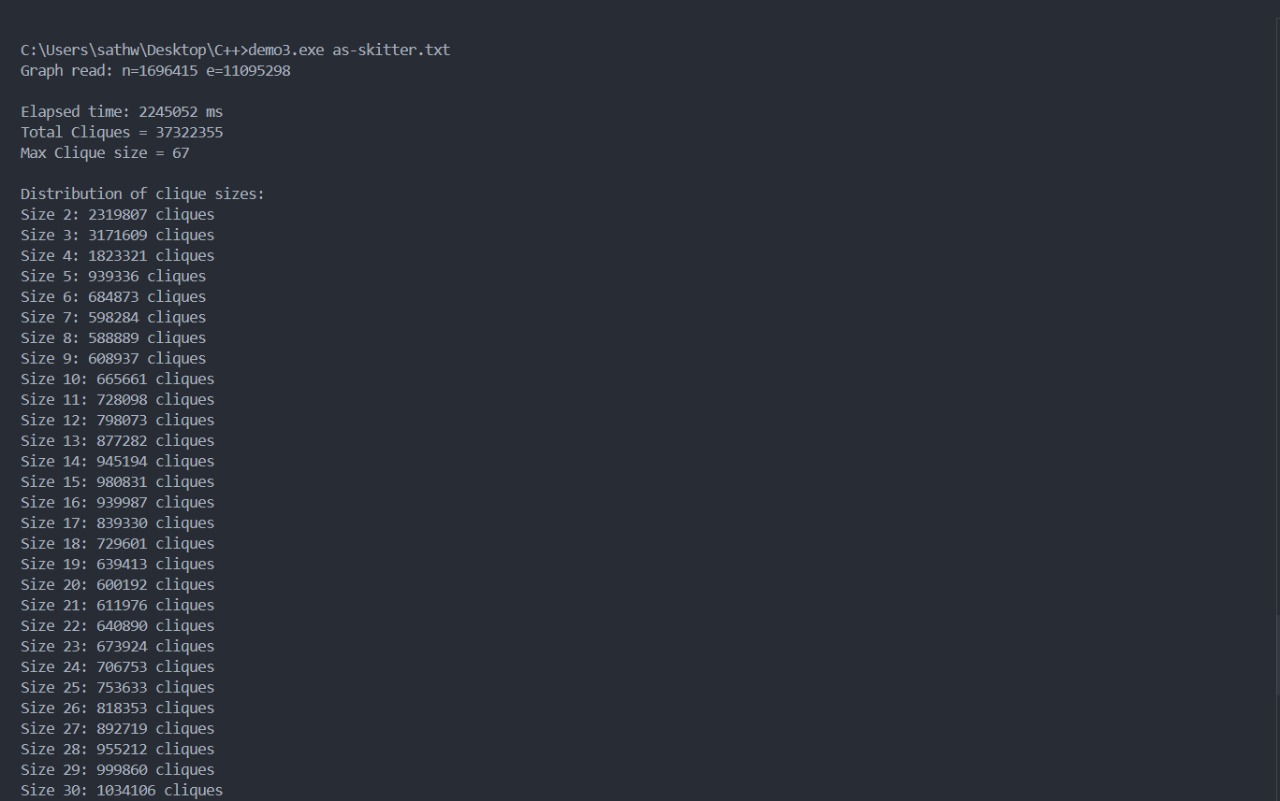
Wiki-vote- 165701 sec

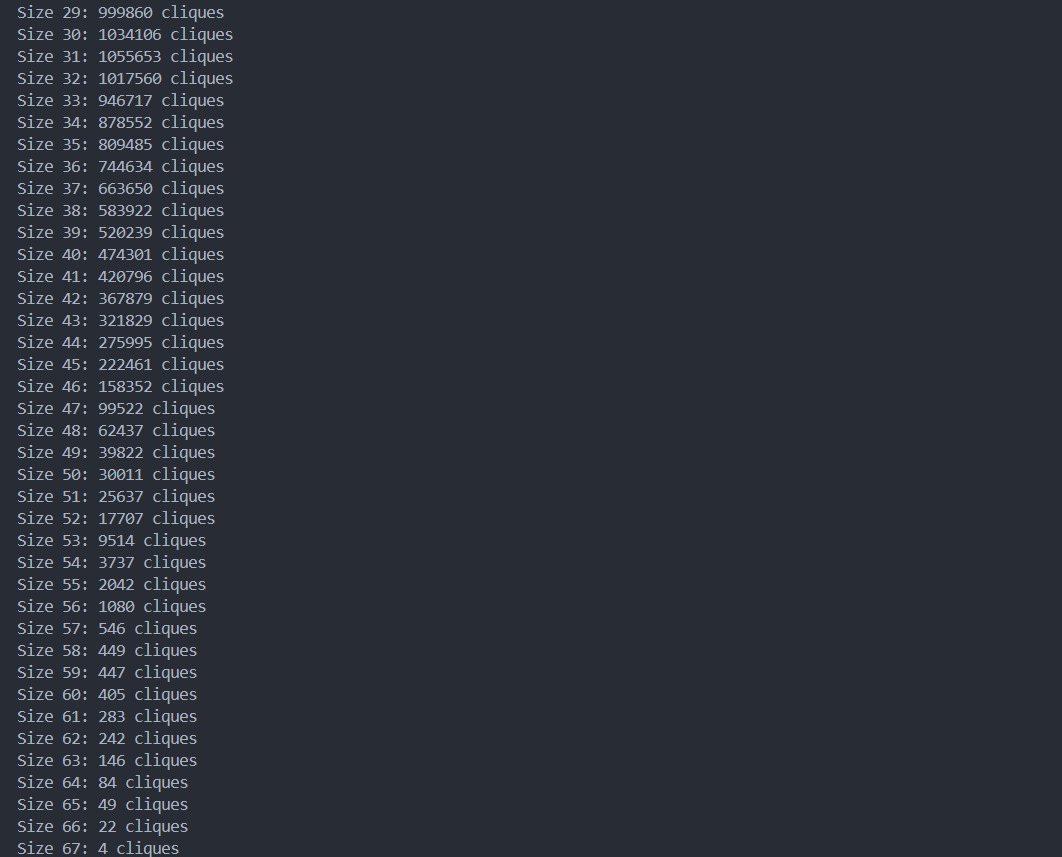
(II) Output & Histogram Observation for Q1,2 & 3 on DataSet: email-Enron –

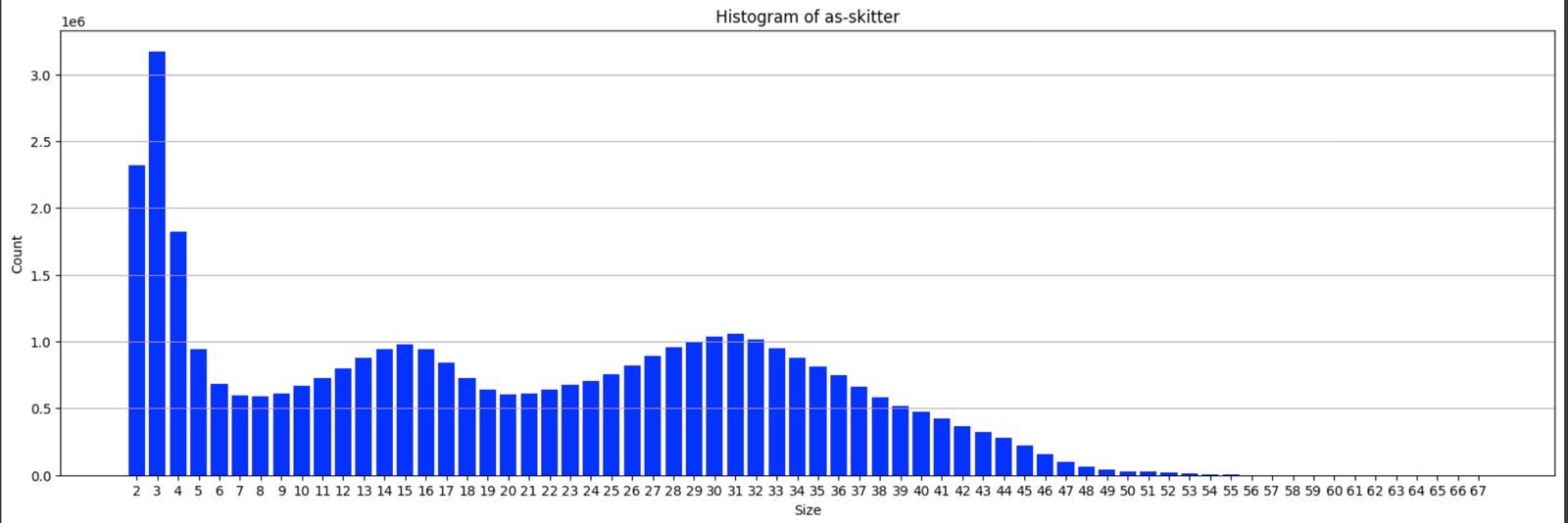




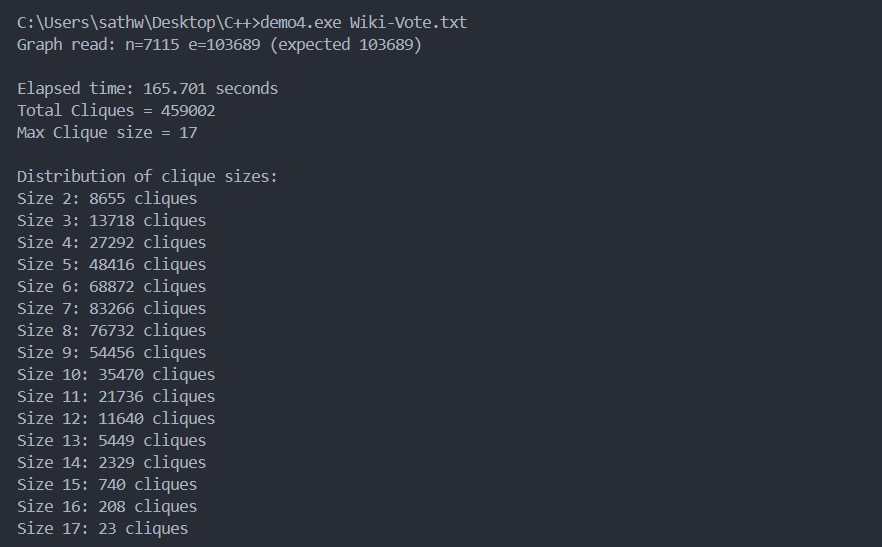
(III) Output & Histogram Observation for Q1,2 & 3 on DataSet: as-Skitter –

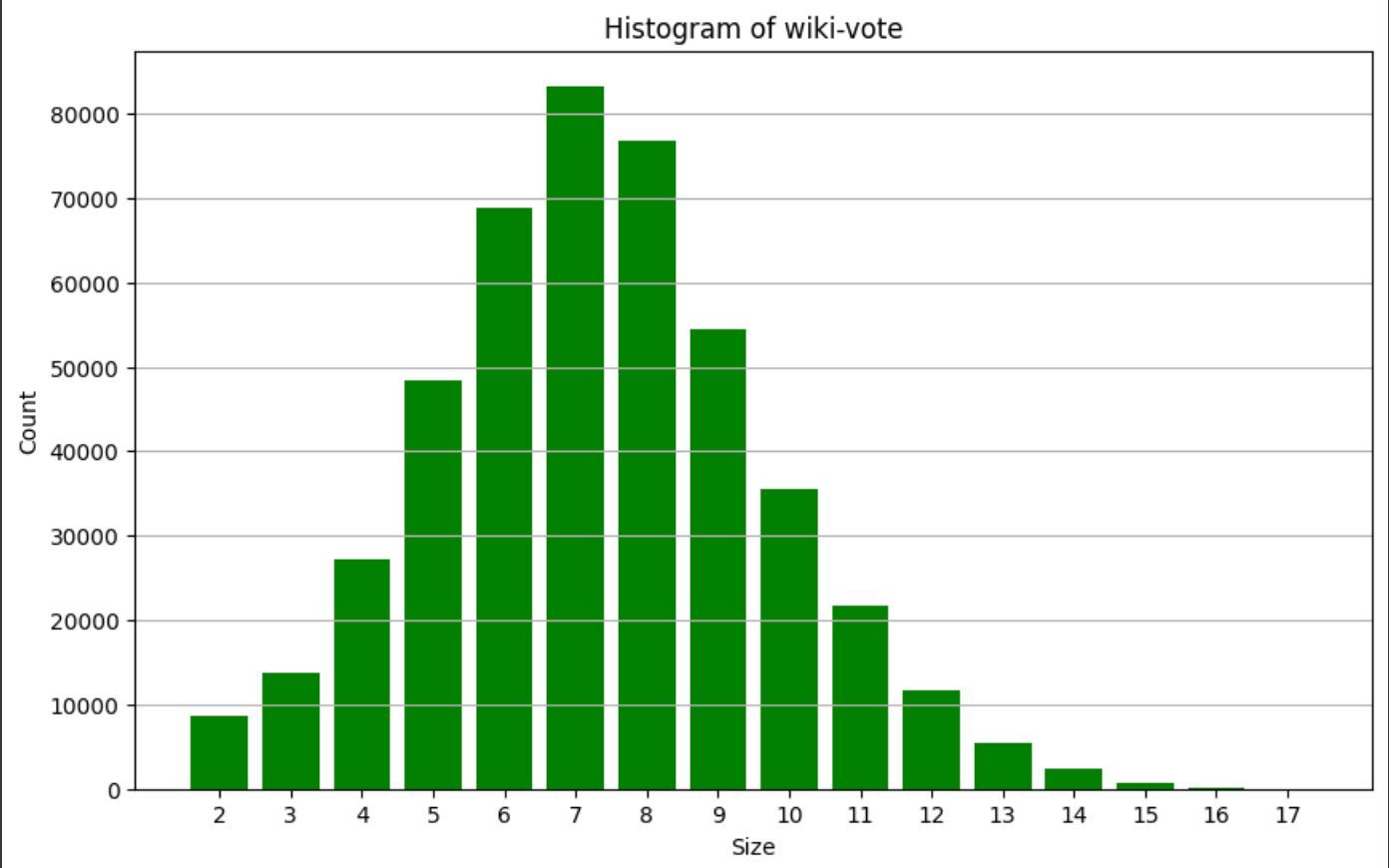






(IV) Output & Histogram Observation for Q1,2 & 3 on DataSet: wiki-Vote –





THANK YOU