Design & Analysis of Algorithms

Assignment - 1

BIRLA INSTITUTE OF TECHNOLOGY AND SCIENCE, PILANI

Sr No	Name	ID No	Contribution
1	Bhaskara Hemanth Karthikeya Ganti	2022A7PS0053H	Bron-Kerbosch Degeneracy Code and Project Report
2	Mohammad Jambughodawala	2022A7PS0009H	Tomita Worst-Case Clique Code and Arboricity Clique Generation Code
3	Harsh Vikram Jajodia	2022A7PS0171H	Arboricity Clique Generation Code and Tomita Worst-Case Clique Code
4	Vaishnav Devaguptapu	2022A7PS0085H	Project Website and Bron-Kerbosch Degeneracy Code
5	Anshul Gopal	2022A7PS0009H	Arboricity Clique Generation Code, Project Report and Readme file

Algorithms

Paper 1: The worst-case time complexity for generating all maximal cliques and computational experiments

The algorithm follows a recursive depth-first search strategy. The main steps are:

1. Initialization: The algorithm starts with an empty set Q, representing the current clique being expanded.
2. Expansion: At each step, a vertex q is added to Q, and a new subgraph is considered consisting of vertices adjacent to q.
3. Pruning: Two pruning techniques are applied:

   Finished Set (FINI): Tracks vertices that have already been processed to avoid duplicate cliques.

   Candidate Set (CAND): Keeps track of vertices that can still be added to Q.

4. Recursion: The process continues recursively, generating larger complete subgraphs until maximal cliques are found.
5. Output Optimization: Instead of storing all maximal cliques explicitly, a tree-like output format is used, reducing space complexity.

The key distinction of this algorithm is its pruning mechanism, which significantly reduces redundant computations compared to previous methods.

The worst-case complexity of the algorithm is derived based on the number of maximal cliques in an n-vertex graph. According to Moon and Moser's theorem, the maximum number of maximal cliques in a graph is O(3^(n/3)). In the context of space complexity, the algorithm primarily uses recursion and tree-based output, reducing the need to store all cliques explicitly. The space complexity is O(n) for the recursive stack in the worst case.

Paper 2: Listing All Maximal Cliques in Sparse Graphs in Near-optimal Time

This algorithm modifies the classic Bron-Kerbosch algorithm by incorporating:

1. Degeneracy Ordering: The graph's vertices are processed in order of degeneracy, ensuring that each recursive call has at most d candidates.
2. Pivoting Strategy: A pivot is selected to minimize recursive calls, following the method of Tomita et al.
3. Efficient Pruning: The algorithm exploits the degeneracy ordering to restrict search space, reducing redundant computations.
4. Graph Data Structure: Uses an adjacency list representation for efficient neighborhood queries.

The algorithm follows these steps:

- Compute a degeneracy ordering of the vertices.
- Process each vertex v in this order, using a modified Bron-Kerbosch approach to find maximal cliques containing v.
- Use pivoting to minimize recursive calls and reduce search space.

The algorithm achieves a worst-case time complexity of O(dn3d/3). This is derived as follows:

- The maximum number of maximal cliques in a d-degenerate graph is bounded by (n-d)3d/3.
- The algorithm ensures that recursive calls are limited by the degeneracy ordering, leading to O(dn3d/3) runtime.
- The space complexity remains O(n+m) due to the adjacency list representation and recursive stack depth being at most O(n).

This complexity is nearly optimal, as it matches the worst-case output size up to a constant factor.

Paper 3: Arboricity & Subgraph Listing Algorithms

1. Initialization:

The graph vertices are numbered in ascending order of their degrees.

Two helper arrays:

S[y]→ Tracks neighbors of vertices not in the clique.

T[y] → Tracks neighbors of vertices in the clique.

The algorithm starts with the first vertex and recursively explores cliques.

2. Expansion and Pruning:

At each step, the algorithm adds vertex iii to the current clique CCC.

Pruning techniques:

- S → Tracks candidates that can still be added.
- T→ Tracks finished vertices to avoid duplicates.

If S(y)=0, the vertex is removed, reducing unnecessary expansions.

3. Maximality Test:

- Before printing, the algorithm verifies clique maximality by checking if adding any neighboring vertex forms a larger clique.
- If no valid expansion exists, the clique is confirmed as maximal and printed.
- This step prevents redundant or non-maximal cliques from being printed.

4. Lexicographic Ordering:

- Ensures only the lexicographically largest cliques are considered.
- Vertices are sorted and validated. Smaller vertices cause the clique to be flagged as invalid, avoiding redundant expansions.

Complexity Analysis

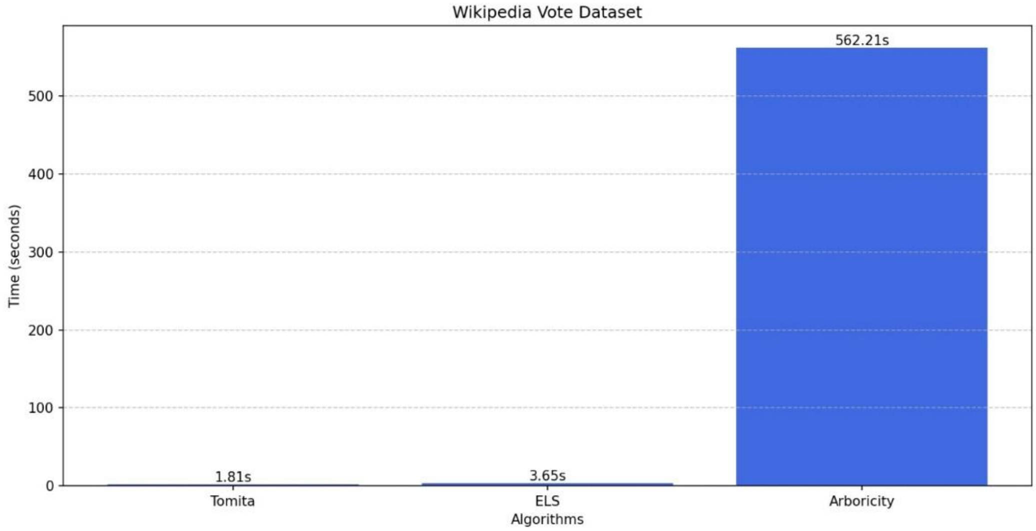
- Time Complexity: O(3^{n/3}) (worst-case)
- Space Complexity: O(n) for the recursion stack, plus O(n^2) for the S and T sets.

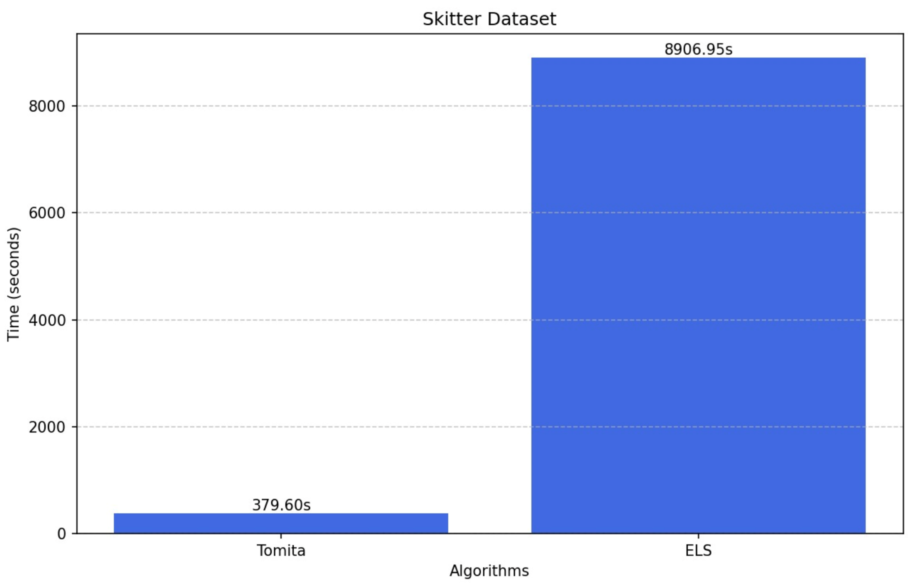
Results:

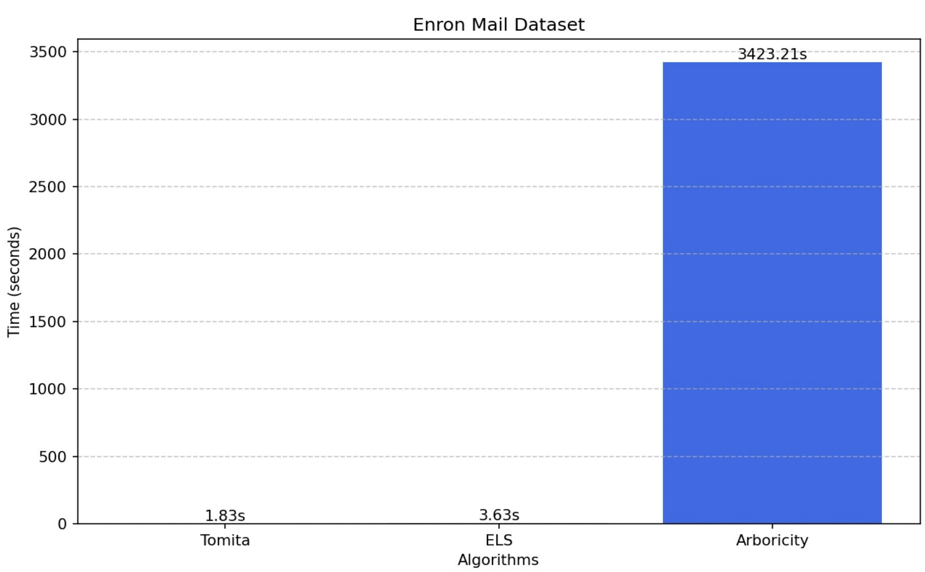
	Enron email network	Wikipedia vote network	Autonomous systems by Skitter
Number of nodes	36692	7115	1696415
Number of edges	183831	103689	11095298
Largest size of the clique in each dataset	20	17	67
Total number of maximal cliques in each dataset	226859	459002	37322355

Execution Time of algorithms on each dataset			
Algorithm	Enron email network	Wikipedia vote network	Autonomous systems by Skitter
Arboricity	3423.21 seconds	562.213 seconds	Ran for 8 hours got 146232 cliques
ELS	1.74138 seconds	1.19985 seconds	8906.95 seconds
Tomita	1.83627 seconds	1.81468 seconds	379.6 seconds

Execution time of Algorithms on each dataset:







Distribution of different size cliques:

Enron email network

Size	Cliques
2	14070
3	7077
4	13319
5	18143
6	22715
7	25896
8	24766
9	22884
10	21393
11	17833
12	15181
13	11487
14	7417
15	3157
16	1178
17	286
18	41
19	10
20	6

Wikipedia vote network

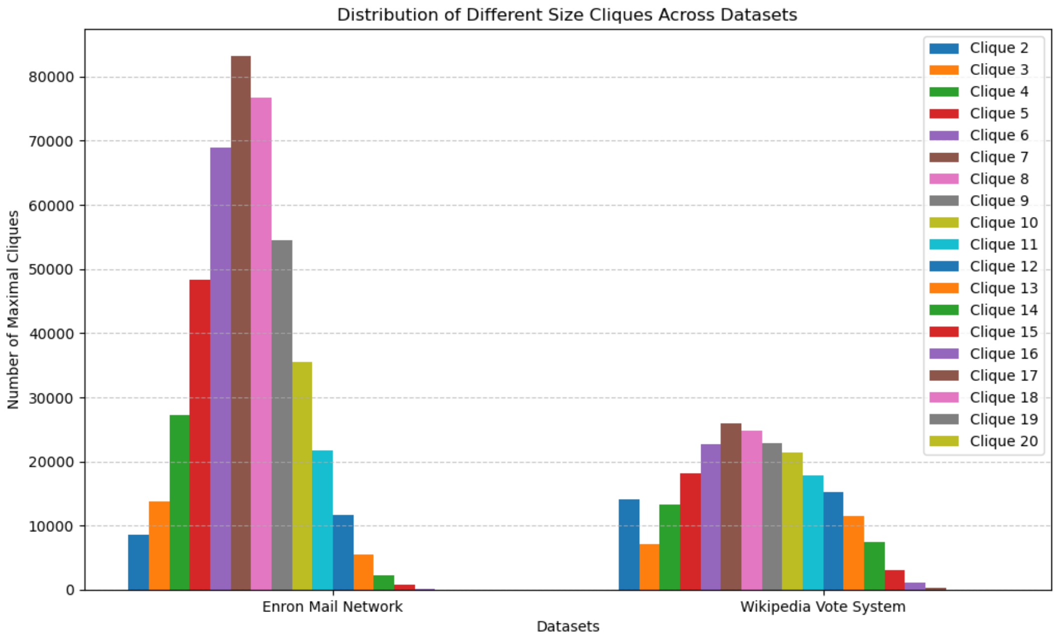
Size	Cliques
2	8655
3	13718
4	27292
5	48416
6	68872
7	83266
8	76732
9	54456
10	35470
11	21736
12	11640
13	5449
14	2329
15	740
16	208
17	23

Autonomous systems by Skitter

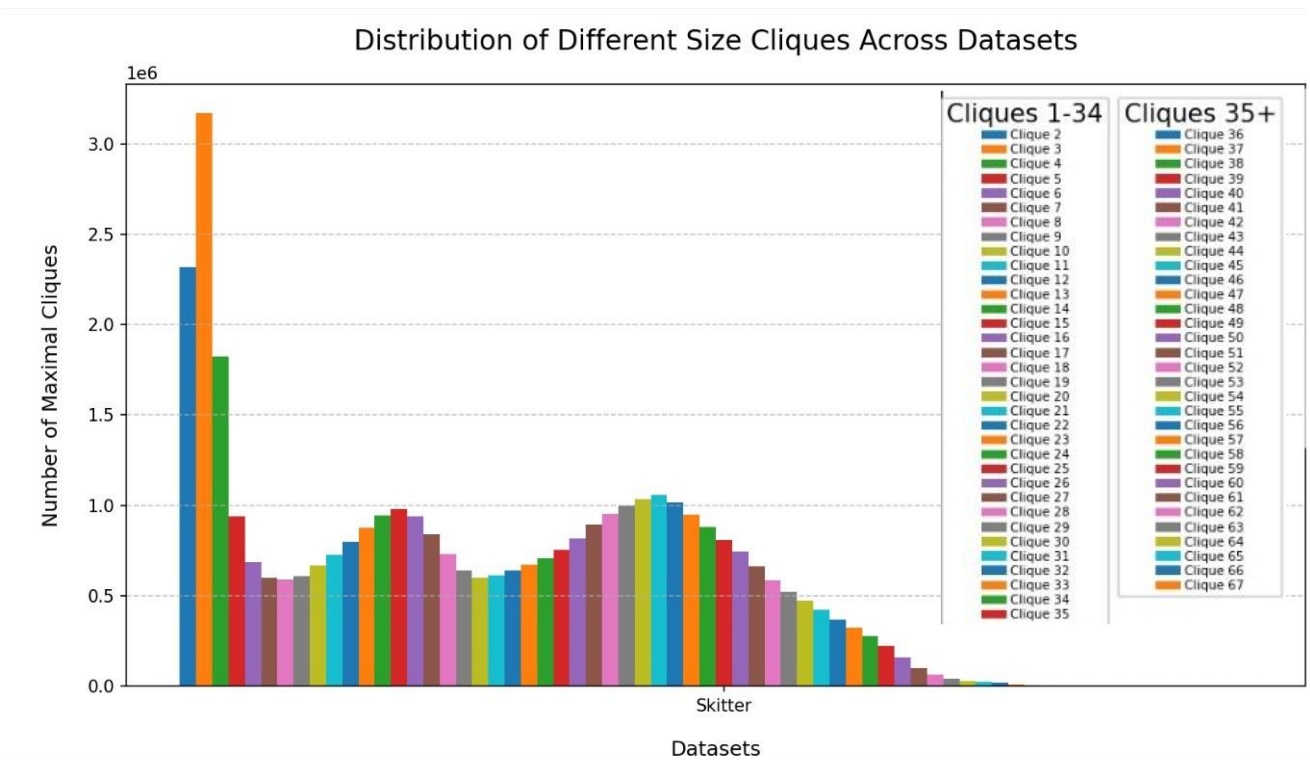
Size	Cliques
2	2319807
3	3171609
4	1823321
5	939336
6	684873
7	598284
8	588889
9	608937
10	665661
11	728098
12	798073
13	877282
14	945194
15	980831
16	939987
17	839330
18	729601
19	639413
20	600192
21	611976
22	640890
23	673924
24	706753
25	753633
26	818353
27	892719
28	955212
29	999860
30	1034106
31	1055653
32	1017560
33	946717
34	878552
35	809485
36	744634
37	663650
38	583922
39	520239
40	474301
41	420796
42	367879
43	321829
44	275995
45	222461
46	158352
47	99522
48	62437
49	39822
50	30011
51	25637
52	17707
53	9514
54	3737
55	2042
56	1080
57	546
58	449
59	447
60	405
61	283
62	242
63	146
64	84
65	49
66	22
67	4

Histogram Representation:

Enron Email Network & Wikipedia Vote System:



Autonomous systems by Skitter :



Thankyou