Design & Analysis Of Algorithms

Implementation and Analysis of Maximal Clique Enumeration Algorithms

Team Members

Name of Student	ID No.	Contribution
Arnav Gupta	2022A7PS0063H	Chiba arbrocity + report
Archith Vinod Kumar	2022A7PS0229H	Tomita + webpage
Manasvi Chervela	2022A7PS2015H	ELS + webpage
Kevin George Mathew	2022A7PS0238H	Chiba arbrocity + report

Introduction

This project focuses on the implementation and comparative analysis of three prominent algorithms for enumerating maximal cliques in graphs: Tomita's CLIQUES procedure, Bron-Kerbosch algorithm with degeneracy ordering, and Chiba-Nishizeki's arboricity-based method.

A clique is a subset of vertices in a graph where every pair of vertices is connected by an edge. A maximal clique is a clique that cannot be extended by adding any other vertex from the graph without violating the clique property.

The goal of this project is to develop efficient implementations of these algorithms in C/C++ and evaluate their performance on diverse graph datasets.

Datasets

1) Wikipedia Vote Network

A. Description of the dataset

Wikipedia is a collaborative free encyclopedia where contributors, including administrators, maintain and edit content. The process of becoming an administrator involves a Request for Adminship (RfA), in which the Wikipedia community votes to approve or reject a candidate.

The Wiki-Vote dataset, extracted from Wikipedia's complete page edit history dump (January 3, 2008), records administrator elections and voting history. It comprises:

- 2,794 elections with 103,663 votes
- 7,066 users who participated either by voting or being voted on
- 1,235 successful promotions and 1,559 failed elections

The dataset is structured as a directed graph, where:

- Nodes represent Wikipedia users.
- Directed edges indicate a vote from one user to another (i.e., if user *A* voted for user *B*, a directed edge from *A* to *B* exists).

This dataset provides insights into voting behaviors, network influence, and community decision-making processes.

B. Source of the dataset

J. Leskovec, D. Huttenlocher, J. Kleinberg. *Signed Networks in Social Media.* CHI 2010.
J. Leskovec, D. Huttenlocher, J. Kleinberg. *Predicting Positive and Negative Links in Online Social Networks.* WWW 2010.

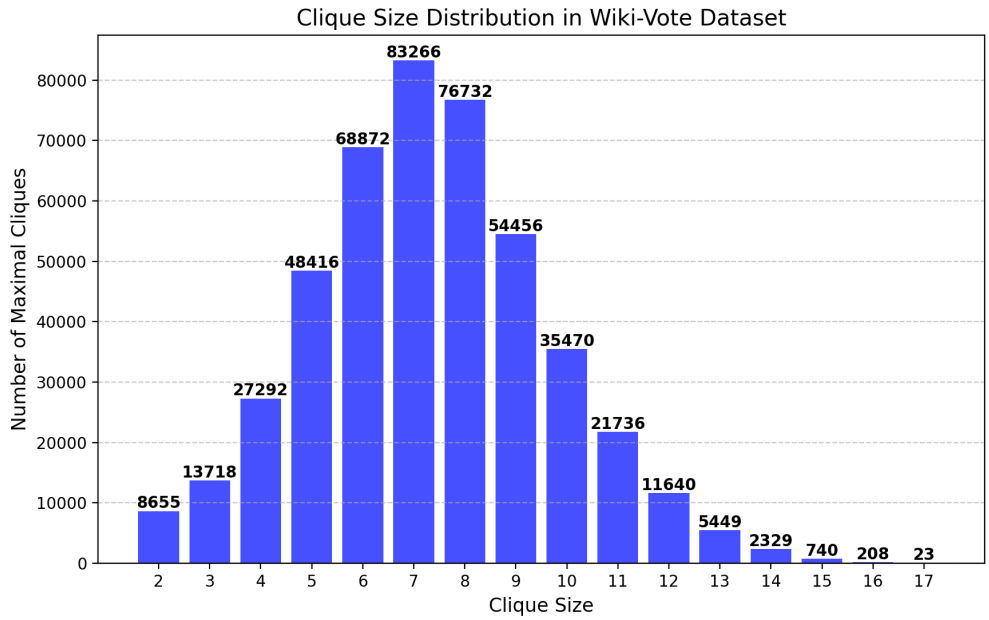
C. Number of nodes and edges

- Nodes: 7,115
- Edges: 103,689

The dataset represents a directed graph, where each edge (A → B) signifies that user A voted for user B to become a Wikipedia administrator.

Each unordered pair of nodes is saved only once, ensuring that duplicate votes are not recorded.

D. Clique Size Distribution



Here is the histogram for the Wiki-Vote dataset, showing the number of maximal cliques (Y-axis) versus clique size (X-axis). I've added annotations for significant values to improve readability.

2) Enron Email Network

A. Description of the dataset

The Email-Enron dataset represents the email communication network within the Enron Corporation. It was made public by the Federal Energy Regulatory Commission (FERC) during its investigation into the company's financial scandal. The dataset contains approximately half a million emails, capturing interactions between employees and external parties.

The dataset is structured as an undirected graph, where:

- Nodes represent unique email addresses.
- Edges indicate at least one email exchange between two addresses (i.e., if an email was sent from address *i* to address *j*, an undirected edge exists between them).

Additionally, non-Enron email addresses appear in the dataset as external nodes (acting as sources and sinks), meaning we only observe their communication with Enron employees, but not their internal interactions.

This dataset provides valuable insights into corporate email communication patterns, social network structures, and has been widely used for graph-based analysis of fraud detection, anomaly detection, and organizational behavior modeling.

B. Source of the dataset

J. Leskovec, K. Lang, A. Dasgupta, M. Mahoney. Community Structure in Large Networks: Natural Cluster Sizes and the Absence of Large Well-Defined Clusters. Internet Mathematics 6(1) 29--123, 2009.

B. Klimmt, Y. Yang. Introducing the Enron corpus. CEAS conference, 2004.

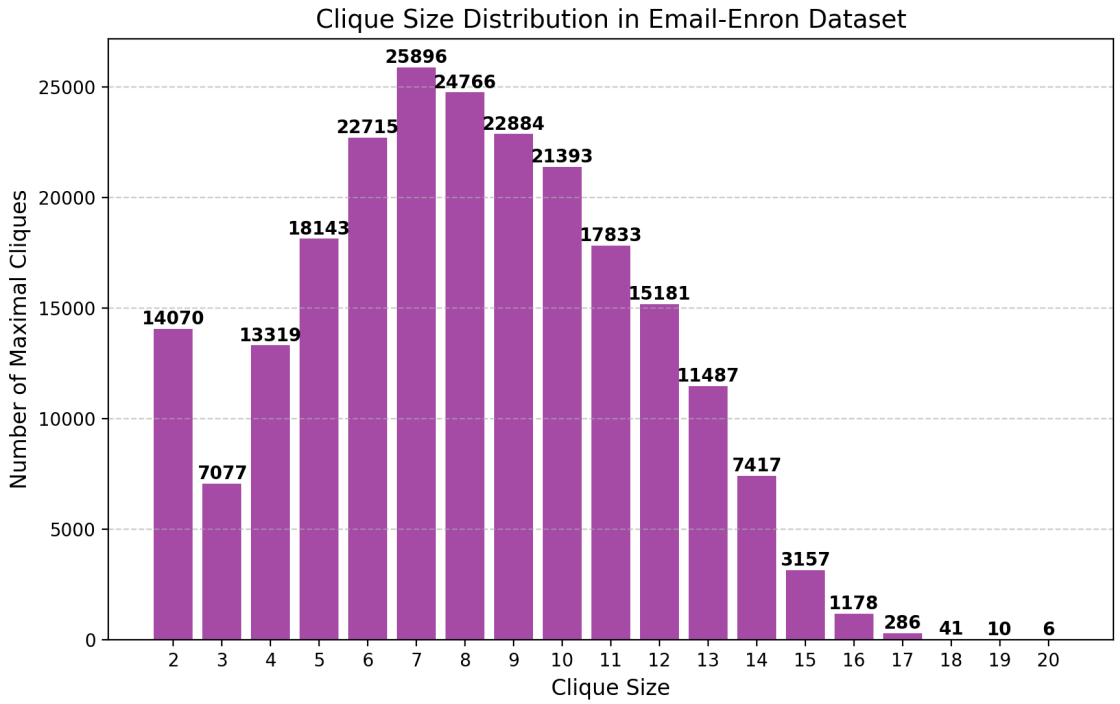
C. Number of nodes and edges

- Nodes: 36,692
- Edges: 367,662

The dataset is represented as an undirected graph, where an edge between two nodes signifies that at least one email was exchanged between the corresponding email addresses.

Each unordered pair of nodes is recorded only once, ensuring no duplicate edges.

D. Clique Size Distribution



Here is the histogram for the Email-Enron dataset, showing the number of maximal cliques (Y-axis) versus clique size (X-axis). I've added annotations for significant values to improve readability.

3) Autonomous Systems by Skitter

A. Description of the dataset

The AS-Skitter dataset represents a large-scale Internet topology graph, constructed from traceroutes collected daily in 2005 by CAIDA (Center for Applied Internet Data Analysis). The dataset captures the structure of the Internet at the Autonomous System (AS) level, mapping the connectivity between different networks.

This dataset was gathered using Skitter, a tool developed in 1998 to actively probe the Internet and analyze topology, routing, and network performance. Skitter was later replaced by the Archipelago (Ark) infrastructure, but its data remains valuable for research in network science and Internet structure analysis.

The dataset is structured as a directed graph, where:

- Nodes represent Autonomous Systems (AS), which are individual networks on the Internet.
- Edges represent observed routing paths between ASes, as discovered through traceroute measurements.

The primary objectives of Skitter data collection include:

- Measuring Forward IP Paths: Capturing the sequence of routers encountered when sending packets from a source to a destination.
- Measuring Round Trip Time (RTT): Recording network latency based on ICMP echo requests.
- Tracking Persistent Routing Changes: Identifying variations in network routing over time.
- Visualizing Network Connectivity: Mapping large-scale Internet structure by probing multiple destination IP addresses.

This dataset provides critical insights into Internet topology, routing dynamics, and network performance, making it a valuable resource for network analysis, cybersecurity, and infrastructure optimization.

B. Source of the dataset

J. Leskovec, J. Kleinberg and C. Faloutsos. Graphs over Time: Densification Laws, Shrinking Diameters and Possible Explanations. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2005.

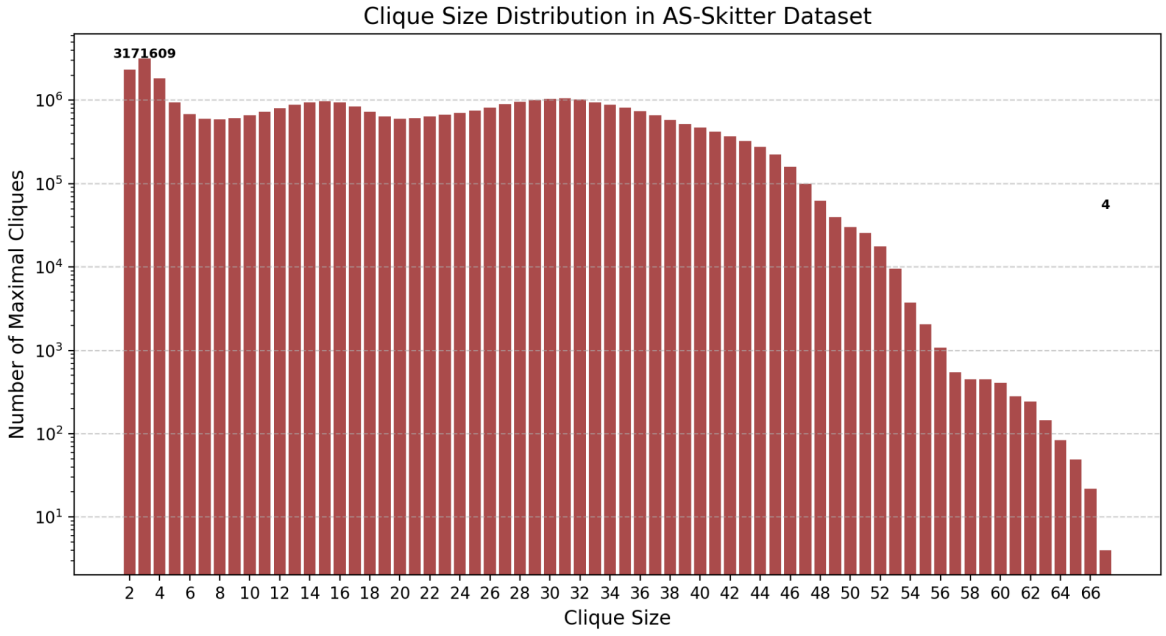
C. Number of nodes and edges

- Nodes: 1,696,415 (including 22,622 isolated nodes with degree 0)
- Edges: 11,095,298

The dataset is represented as an undirected graph, where an edge signifies a connection between two Autonomous Systems (AS) as observed through traceroute measurements.

The presence of isolated nodes (nodes with no connections) indicates that some ASes were detected but had no observed routing links in the dataset.

D. Clique Size Distribution



Here is the clique size distribution histogram for the AS-Skitter dataset. I have displayed the maximum and minimum values for visualization.

Algorithms

1) Tomita's Clique

A. Description

The Tomita Clique Algorithm is a depth-first search (DFS)-based approach for maximal clique enumeration in an undirected graph. It improves upon previous methods, particularly the Bron–Kerbosch algorithm, by introducing efficient pruning techniques to eliminate redundant computations. The algorithm ensures worst-case optimal performance and generates all maximal cliques without duplication. Key Features are:

- Recursive Expansion: The algorithm maintains a growing clique set and expands it by adding vertices that form a complete subgraph.
- Pruning Techniques:
    - Pivot Selection: A pivot vertex is selected to reduce the number of recursive calls by minimizing candidate expansions.
    - Early Termination: If a vertex cannot extend a clique to a maximal one, it is ignored.
- Tree-like Output Format: Unlike traditional approaches that store all maximal cliques explicitly, Tomita's algorithm prints cliques incrementally, reducing memory overhead.

This a brief description of this algorithm.

B. Pseudo Code

```
procedure  CLIQUES(G)
                /* Graph  G = (V, E) */
begin
 0′:/*    Q := ∅;                              */
       /* global variable Q is to constitute a clique */
 1 : EXPAND(V,V)
end of CLIQUES

       procedure EXPAND(SUBG, CAND)
       begin
 2 :     if SUBG = ∅
 3 :       then print ("clique,")
                 /* to represent that Q is a maximal clique */
 4 :       else u := a vertex u in SUBG that maximizes | CAND ∩ Γ(u) |;
                 /* let EXT_u = CAND − Γ(u); */
                 /*     FINI := ∅;            */
 5 :             while CAND − Γ(u) ≠ ∅
 6 :                 do q := a vertex in (CAND − Γ(u));
 7 :                    print (q, ",");
                        /* to represent the next statement */
 7′: /*                 Q := Q ∪ {q};         */
 8 :                    SUBG_q := SUBG ∩ Γ(q);
 9 :                    CAND_q := CAND ∩ Γ(q);
10:                     EXPAND(SUBG_q, CAND_q);
11:                     CAND := CAND − {q};    /* FINI := FINI ∪ {q}; */
12:                     print ("back,");
                        /* to represent the next statement */
12′:/*                  Q := Q − {q}          */
                    od
         fi
       end of EXPAND
```

Fig. 2. Algorithm CLIQUES.

C. Time Complexity

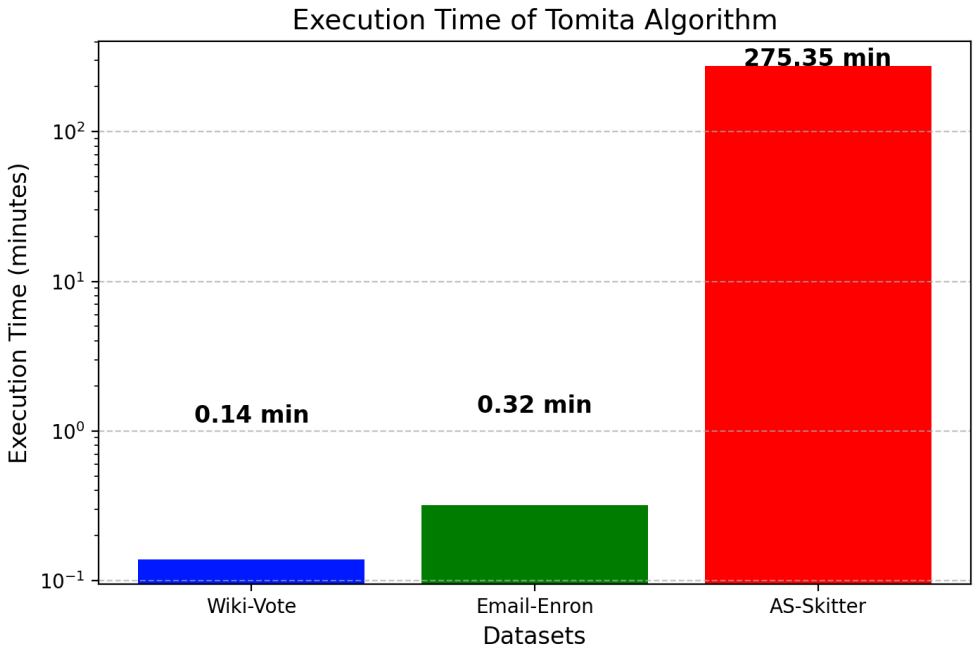
The worst-case time complexity of the Tomita algorithm is $O(3^{n/3})$. This bound is optimal because there exist up to $3^{n/3}$ maximal cliques in an n-vertex graph (as shown by Moon and Moser).

Complexity Breakdown

- Recursive calls: Each recursive step processes a reduced graph. The pivot selection strategy reduces unnecessary branches.
- Pruning via FINI and CAND sets: Ensures that already processed vertices are not reconsidered, avoiding duplicate work.
- Space Complexity: The algorithm avoids storing all cliques explicitly, making it memory-efficient compared to other methods like Bron–Kerbosch.

This is the Worst Case Time Complexity for this algorithm

D. Performance Results



Here is the histogram showing the execution times of the Tomita Algorithm for each dataset. The Y-axis is in minutes, and I have annotated the bars with the exact values since the data is highly skewed. The Y-axis uses a log scale to improve visualization.

2) ELS

A. Description

The Eppstein–Löffler–Strash (ELS) Algorithm is an advanced fixed-parameter tractable (FPT) algorithm for maximal clique enumeration in sparse graphs, designed to achieve near-optimal performance. It builds on the Bron–Kerbosch algorithm but introduces degeneracy ordering to improve efficiency. Key Features are:

- Degeneracy Ordering: The algorithm processes vertices in order of degeneracy, a measure of graph sparsity.
- Recursive Expansion: Similar to Bron–Kerbosch, the algorithm builds cliques recursively.
- Pruning via Pivoting: Uses pivoting to limit recursive calls and improve efficiency.

This makes the ELS algorithm particularly effective for real-world networks (e.g., social networks, citation graphs, biological networks), which tend to have low degeneracy.

B. Pseudo Code

proc BronKerbosch(P, R, X)

1: **if** $P \cup X = \emptyset$ **then**
2: report R as a maximal clique
3: **end if**
4: **for each** vertex $v \in P$ **do**
5: BronKerbosch($P \cap \Gamma(v)$, $R \cup \{v\}$, $X \cap \Gamma(v)$)
6: $P \leftarrow P \setminus \{v\}$
7: $X \leftarrow X \cup \{v\}$
8: **end for**

proc BronKerboschPivot(P, R, X)

1: **if** $P \cup X = \emptyset$ **then**
2: report R as a maximal clique
3: **end if**
4: choose a pivot $u \in P \cup X$ {Tomita et al. choose u to maximize $|P \cap \Gamma(u)|$}
5: **for each** vertex $v \in P \setminus \Gamma(u)$ **do**
6: BronKerboschPivot($P \cap \Gamma(v)$, $R \cup \{v\}$, $X \cap \Gamma(v)$)
7: $P \leftarrow P \setminus \{v\}$
8: $X \leftarrow X \cup \{v\}$
9: **end for**

Fig. 2. The Bron–Kerbosch algorithm without and with pivoting

proc BronKerboschDegeneracy(V, E)

1: **for** each vertex v_i in a degeneracy ordering $v_0, v_1, v_2, \ldots$ of (V, E) **do**
2: $P \leftarrow \Gamma(v_i) \cap \{v_{i+1}, \ldots, v_{n-1}\}$
3: $X \leftarrow \Gamma(v_i) \cap \{v_0, \ldots, v_{i-1}\}$
4: BronKerboschPivot(P, $\{v_i\}$, X)
5: **end for**

Fig. 4. Our algorithm.

C. Time Complexity

The ELS algorithm runs in $O(dn3^{d/3})$ time, where d is the degeneracy of the graph.

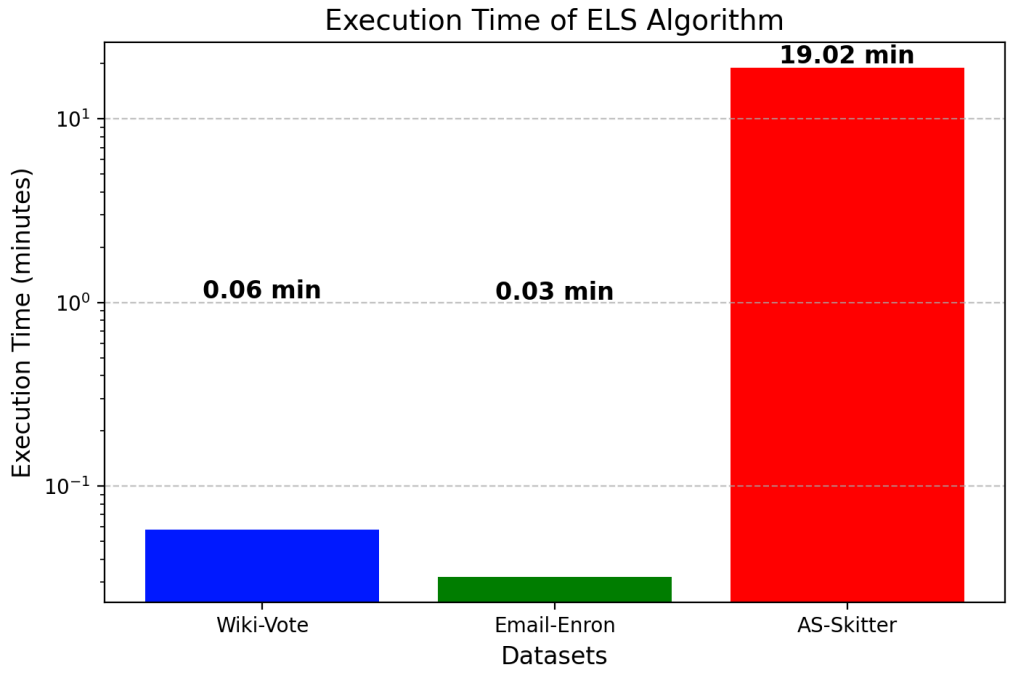
Complexity Breakdown:

- Recursive Calls: Bounded by degeneracy d, reducing unnecessary expansions.
- Pivot Selection: Limits the number of vertices examined in each call.

- Space Complexity: Similar to Bron–Kerbosch, avoiding explicit clique storage.

For graphs with low degeneracy (e.g., real-world networks), this results in practical efficiency gains.

D. Performance Results



Here is the histogram showing the execution times of the ELS Algorithm for each dataset. The Y-axis is in minutes, and I have annotated the bars with the exact values since the data is highly skewed. The Y-axis uses a log scale to improve visualization.

3) Chiba's Arbrocity

A. Description

The Chiba-Nishizeki Algorithm is an efficient maximal clique enumeration method that leverages the arboricity of a graph to improve performance. Unlike traditional methods such as Bron–Kerbosch or Tomita's Algorithm, which focus on pivoting and degeneracy ordering, this algorithm is optimized for sparse graphs by exploiting edge-searching strategies. Key Features are:

- Arboricity-Based Pruning: The algorithm bounds recursive calls using a(G), where a(G) is the arboricity of the graph (the minimum number of edge-disjoint spanning forests required to cover all edges).
- Edge-Searching Strategy: Instead of iterating over vertices, it systematically processes edges to list all cliques efficiently.
- Avoiding Duplicates: By deleting vertices progressively, the algorithm ensures that no clique is listed more than once.
- Optimal for Planar Graphs: Since $a(G) \leq 3$ for planar graphs, the algorithm runs in linear time for such graphs.

This makes the Chiba-Nishizeki algorithm one of the most efficient clique enumeration techniques, particularly for real-world sparse graphs such as social networks, road networks, and biological graphs.

B. Pseudo Code

```
procedure CLIQUE;
  procedure UPDATE (i, C);
   begin
    if i = n+1
     then print out a new clique C
     else
      begin
```



```
1:      if C − N(i) ≠ ∅ then UPDATE (i+1, C);
      {prepare for tests}
        {compute T[y] = |N(y) ∩ C ∩ N(i)| for y ∈ V − C − {i}}
2:      for each vertex x ∈ C ∩ N(i)
         do for each vertex y ∈ N(x) − C − {i}
              do T[y] := T[y]+1;
        {compute S[y] = |N(y) ∩ (C − N(i))| for y ∈ V − C}
3:      for each vertex x ∈ C − N(i)
         do for each vertex y ∈ N(x) − C
              do S[y] := S[y]+1;
        FLAG := true;
      {maximality test}
4:       if there exists a vertex y ∈ N(i) − C such that y < i and T[y] = |C ∩ N(i)|
          then FLAG := false; {(C ∩ N(i)) ∪ {i} is not a clique of G_i}
      {lexico. test}
        {C ∩ N(i) corresponds to C_o in Lemma 6}
5:      sort all the vertices in C − N(i) in ascending order j_1 < j_2 < ··· < j_p, where
            p = |C − N(i)|;
        {case S(y) ≧ 1. See Lemma 6.}
6:      for k := 1 to p
         do for each vertex y ∈ N(j_k) − C such that y < i and T[y] = |C ∩ N(i)|
              do if y ≧ j_k
                   then S[y] := S[y] − 1 {alter S[y] to S(y)}
                   else
                    if (j_k is the first vertex which satisfies y < j_k)
                     then {S[y] = S(y)}
                      if (S[y] + k − 1 = p) and (y ≧ j_{k−1}) {j_0 = 0}
                       then FLAG := false; {C is not lexico. largest}
        {case S(y) = 0}
```

```
7:      if C ∩ N(i) ≠ ∅
          then for each vertex y ∉ C ∪ {i} such that y < i, T[y] = |C ∩ N(i)| and
                    S[y] = 0
                      {access y from the adjacency list of a vertex in C ∩ N(i)}
               do if j_p < y then FLAG := false          {C is not lexico. largest.}
          else if j_p < i − 1 then FLAG := false;        {C is not lexico. largest.}
        {reinitialize S and T}
8:      for each vertex x ∈ C ∩ N(i)
          do for each vertex y ∈ N(x) − C − {i}
               do T[y] := 0;
9:      for each vertex x ∈ C − N(i)
          do for each vertex y ∈ N(x) − C
               do S[y] := 0;
        {FLAG is true if and only if (C ∩ N(i)) ∪ {i} is a clique of G_i and C is the
        lexicographically largest clique of G_{i−1} containing C ∩ N(i).}
10:     if FLAG
          then
          begin
            SAVE := C − N(i);
            C := (C ∩ N(i)) ∪ {i};
            UPDATE (i+1, C);
```



```
            C := (C − {i}) ∪ SAVE
          end
      end
  end;
begin {of CLIQUE}
  number the vertices of a given graph G in such a way that d(1) ≦ d(2) ≦ · · · ≦
      d(n);
  for i := 1 to n {initialize S and T}
    do begin S[i] := 0; T[i] := 0 end;
  C := {1};
  UPDATE (2,C)
end {of CLIQUE};
```

C. Time Complexity

The Chiba-Nishizeki algorithm runs in O(a(G) m) time per clique, where:

- a(G) is the arboricity of the graph.

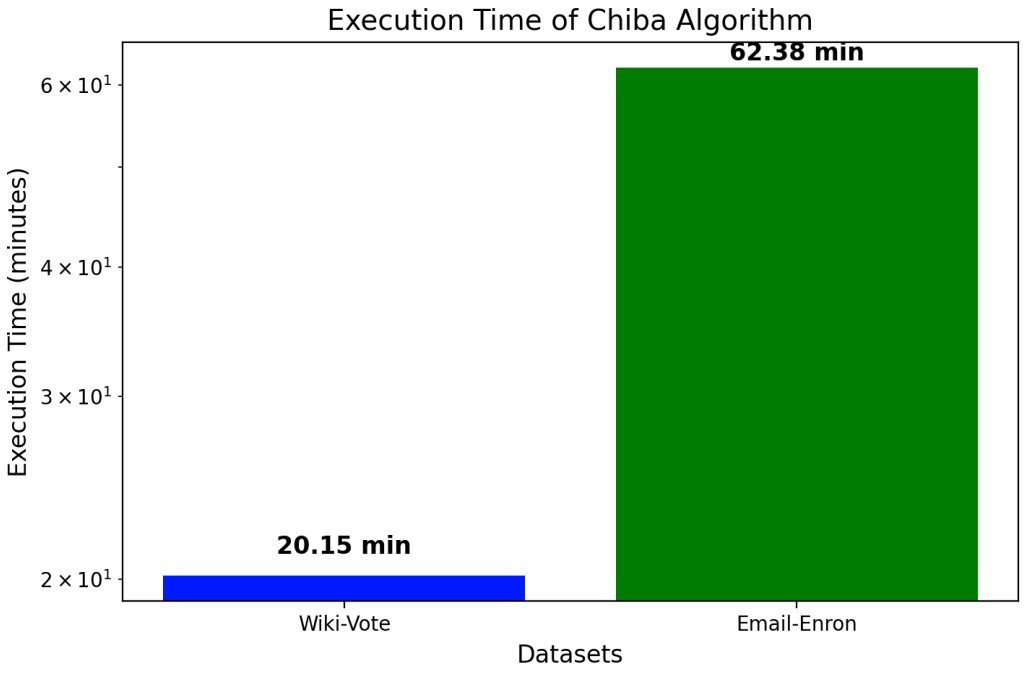
- m is the number of edges.

Complexity Breakdown:

- Edge Searching Strategy: The algorithm avoids vertex-wise enumeration, reducing redundant operations.
- Arboricity-Based Bound: Since a(G) ≤ sqrt(2m + n) / 2, the time complexity is significantly better than $O(n^3)$ methods for sparse graphs.
- Space Complexity: O(m), since it does not require storing all cliques explicitly.

For planar graphs (a(G) ≤ 3), the algorithm runs in linear time (O(n) per clique), making it one of the most efficient clique-finding algorithms for large real-world graphs.

D. Performance Results



Here is the histogram showing the execution times of the Chiba Algorithm for each dataset. The Y-axis is in minutes, and I have annotated the bars with the exact values since the data is highly skewed. The Y-axis uses a log scale to improve visualization.

THANKYOU