Report of Maximal Clique Enumeration Algorithms

Algorithm Descriptions

1. Chiba-Nishizeki (1985)

Paper: Arboricity and Subgraph Listing Algorithms

- Chiba and Nishizeki introduce a strategy using the concept of *arboricity* $a(G)$, which is the minimum number of edge-disjoint forests that cover all edges of a graph.
- The algorithm processes vertices in **non-increasing degree order**.
- For each vertex v, the algorithm scans all edges among the neighbors of v to detect cliques that contain v.
- After processing, v is removed from the graph to avoid redundant work.
- The key insight is that deletion in decreasing degree order reduces the number of times high-degree vertices appear in recursive paths.
- **Time Complexity:** $O(a(G) \cdot m)$ per clique, where m is the number of edges.
- Particularly efficient for graphs with low arboricity, such as planar or sparse graphs.

2. Tomita et al. (2006)

Paper: The Worst-case Time Complexity for Generating All Maximal Cliques

- A refined DFS-based backtracking method using the **Bron–Kerbosch algorithm with pivoting**.
- Introduces a theoretical worst-case optimal output method using a tree-like output structure.
- Maintains three sets: R (current clique), P (potential candidates), X (excluded vertices).
- Pivoting is used to avoid redundant recursive calls: select a pivot vertex u from $P \cup X$ such that $|P \cap \Gamma(u)|$ is maximized.
- Recursive calls are only made for vertices in $P \setminus \Gamma(u)$, minimizing the branching factor.
- **Time Complexity:** $O(3^{n/3})$, proven to be optimal in the worst case due to the Moon-Moser bound.
- Although worst-case optimal, it may be slower on large sparse graphs due to higher recursion overhead.

3. Eppstein–Löffler–Strash (2010)

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

- Improves Bron–Kerbosch by using **degeneracy ordering** to guide recursion.

- Degeneracy d of a graph is the smallest integer such that every subgraph has a vertex with degree $\leq d$.
- For each vertex v in degeneracy order, the algorithm makes a Bron–Kerbosch call restricted to the neighborhood later in the order.
- Inside each recursive call, pivoting is applied (same as Tomita et al.).
- Guarantees that recursive calls handle at most d vertices in their candidate sets, reducing depth and size of recursion tree.
- **Time Complexity:** $O(d \cdot n \cdot 3^{d/3})$, fixed-parameter tractable in degeneracy d.
- Efficient and scalable for sparse real-world graphs with small d.

Experimental Results

Table 1: Runtime comparison on real-world datasets

Algorithm	Email-Enron	Wiki-how	Skitter
Chiba	1 hour 8m	45m 24s	24h 53m
ELS	9s	7.82s	20m 36s
Tomita	7.63s	3.46s	16h 32m

Experimental analysis

- **Email-Enron (Small, Sparse)**:
  - Both ELS and Tomita perform extremely well.
  - Chiba suffers from expensive per-clique costs due to high branching and poor edge locality.
  - Tomita is slightly faster than ELS due to fewer recursive layers on such small graphs.
- **Wiki-how (Moderate Size)**:
  - All three algorithms scale reasonably well.
  - ELS maintains speed due to effective pruning from degeneracy order.
  - Tomita continues to be fast, but Chiba becomes significantly slower as the number of cliques grows.
- **Skitter (Large Scale Graph)**:
  - ELS vastly outperforms the others, finishing in under 21 minutes, owing to small degeneracy (sparse topology).
  - Chiba's performance degrades dramatically with clique explosion and quadratic edge operations.
  - Tomita, despite being worst-case optimal, takes over 16 hours due to depth and width of the search tree.