ASA Project Report

ASA 2025/2026 Project 1 Report

Group: ALxxx/TPyyy **Student(s):** Name1 (97xxx) and Name2 (102xxx)

Problem and Solution Description

The problem involves calculating the number of distinct paths between pairs of intersections in a directed acyclic graph (DAG) and assigning truck routes based on this count. The proposed solution uses an iterative approach based on topological sorting to ensure nodes are processed in the correct dependency order, avoiding recursion and stack overflow issues.

To optimize cache and memory usage, we implemented a **Batched Dynamic Programming** scheme. We process source nodes in blocks (e.g., 128), maintaining path counters only for the current batch. Path propagation follows the linearized topological order (Kahn's algorithm), where for each node, we sum paths from its predecessors and propagate them to its successors, updating counters modulo M.

Theoretical Analysis

The algorithm is divided into the following main steps:

1. **Graph Reading and In-Degree Calculation**: Iterating over the K edges to build the adjacency list and degree vector. **Complexity:** $O(N + K)$

2. **Topological Sort (Kahn's Algorithm)**: Inserting nodes with degree 0 into a queue and iteratively removing them, visiting each edge once. **Complexity:** $O(N + K)$

3. **Batch Processing (Core Solution)**: For each block of sources, we traverse the graph in topological order. In the worst case (dense graph), we process all edges for each batch. With B being the batch size, we have N/B outer iterations. **Complexity:** $O(\frac{N}{B} \times (N + K))$

Global Complexity: Since B is a constant, the complexity simplifies to:

$$O(N \times (N + K))$$

In the worst case (dense graph where $K \approx N^2$), it is $O(N^3)$. For sparse graphs, it approaches $O(N^2)$.

Experimental Evaluation

Experiments were conducted on a machine with an Apple M-series CPU, using randomly generated instances with constant density. The goal was to verify the correlation between execution time and the predicted theoretical complexity $O(N \times (N + K))$.

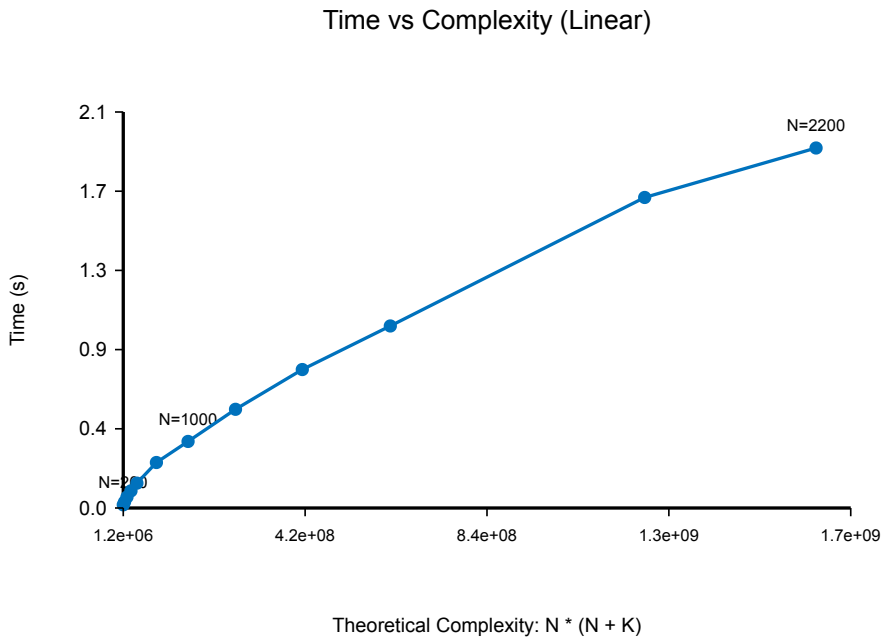
Results Table

More than 10 instances of incremental size were generated.

N (Vertices)	K (Edges)	Time (s)
200	5937	0.0172
300	13620	0.0326
400	24075	0.0595
500	37572	0.0930
600	53856	0.1348
800	96280	0.2466
1000	149645	0.3608
1200	215511	0.5350
1400	294370	0.7506
1600	384045	0.9868
2000	599803	1.6834
2200	724530	1.9516

Performance Graph

The graph below plots Real Time (Y-axis) against Theoretical Complexity $N \times (N + K)$ (X-axis).



Time vs Complexity

The linearity observed in the graph confirms that the implementation follows the theoretical prediction.