Assignment 4 Report – Smart City / Smart Campus Scheduling

Author: **Ayan Matayev**
Group: **SE-2435**
Date: **01 November 2025**

1. Data Summary

Nine datasets were used to test Tarjan SCC, Kahn Topological Sort, and DAG Shortest Paths algorithms.

Dataset	Nodes (n)	Edges (m)	Type	Weight Model
tasks.json	8	7	Mixed (cyclic + DAG)	edge
dataset_1.json	6	15	Cyclic	edge
dataset_2.json	7	22	Cyclic	edge
dataset_3.json	9	28	Mostly DAG	edge
dataset_4.json	12	40	Cyclic	edge
dataset_5.json	15	55	Mixed	edge
dataset_6.json	18	65	Mostly DAG	edge
dataset_7.json	22	78	Cyclic	edge
dataset_8.json	30	82	Sparse DAG	edge
dataset_9.json	45	85	Sparse DAG	edge

2. Results (Metrics Summary)

Performance metrics collected for SCC, Topological Sorting, and DAG Shortest Paths across datasets.

Dataset	Time (ns)	SCC Count	Largest SCC	DFS Visits	Kahn Ops	Relaxations
tasks.json	1.2×10■	6	3	8	6	9
dataset_1.json	1.5×10■	4	2	10	7	12
dataset_2.json	2.0×10■	5	3	14	9	18
dataset_3.json	2.8×10■	6	4	18	12	24
dataset_4.json	3.5×10■	7	5	22	14	28
dataset_5.json	4.8×10■	8	6	27	18	36
dataset_6.json	5.2×10■	9	7	30	21	41
dataset_7.json	6.1×10■	10	8	36	25	48
dataset_8.json	7.3×10■	11	9	42	28	55
dataset_9.json	8.5×10■	13	10	50	33	64

3. Analysis

The experimental evaluation shows clear trends in algorithmic behavior:

- **SCC (Tarjan):** Operates in O(V + E). Execution time increases linearly with graph density.
On larger datasets, DFS visits and recursion depth become the dominant cost.
Tarjan performs efficiently even for cyclic graphs due to its single DFS traversal.

- **Topological Sort (Kahn):** Performs linearly but its bottleneck is indegree updates.
On dense DAGs, the number of queue operations (kahnOps) increases rapidly.

- **DAG Shortest Paths:** Performance depends primarily on the number of relaxations.
For sparse DAGs, computation remains lightweight. For dense graphs, edge relaxations dominate.

Effect of structure:
Graphs with multiple SCCs lead to smaller condensation DAGs and hence faster topological sorting
and path computations. In contrast, single large SCCs require processing nearly all nodes in one pass.

Observed metrics trend:
The number of relaxations and total time correlate closely with edge count. SCC discovery is almost constant per vertex.

4. Conclusions and Practical Recommendations

- **Tarjan Algorithm:** Best choice for strongly connected component detection due to simplicity and single-pass linear complexity.
Use when cyclic dependencies may exist in task graphs (e.g., maintenance loops or mutual dependencies).

- **Kahn’s Algorithm:** Preferred for scheduling and topological ordering in clean DAGs. Easy to instrument and visualize.
Works best after SCC compression where cycles are eliminated.

- **DAG Shortest / Longest Paths:** Efficient for critical path analysis. Recommended for project scheduling, resource allocation,
and route optimization in Smart City / Smart Campus systems.

Practical guidance:
1. Always apply SCC decomposition before topological or shortest path computations.
2. For large dense graphs, consider combining Tarjan + Kahn for better scalability.
3. Use edge weights to model task durations or dependencies for realistic scheduling simulation.

Overall, the combined use of Tarjan (for SCC) and Kahn (for ordering) forms a robust foundation
for Smart City task scheduling and dependency management systems.