Graph Algorithm Analysis Report

1. Introduction

The objective of this assignment was to explore graph-theoretical algorithms for scheduling tasks in a smart city/campus environment. Task dependencies, such as street maintenance, cleaning, sensor calibration, and analytics subtasks, may form cyclic or acyclic graphs, necessitating specialized methods for analysis and optimization.

The focus of this work includes:

Detection of Strongly Connected Components (SCCs) in directed graphs using Kosaraju's Algorithm.

Construction of Condensation Graphs and computation of Topological Order.

Determination of Shortest and Longest Paths (Critical Path) in Directed Acyclic Graphs (DAGs), utilizing the topological order for linear-time complexity.

Performance evaluation, including time complexity and scalability.

Weight Model Choice: For path calculations, Edge Weights were chosen, representing the duration or cost of the dependency/transition between two tasks or components. Longest paths were calculated by applying the shortest path algorithm on a DAG with negated edge weights.

2. Dataset Description

Nine datasets were generated to evaluate algorithms across different sizes and densities. The density ratio provides insight into the graph structure.

Таблица1

Category	Dataset	# N (Nodes)	# M (Edges)	Density (M/N)	Graph Type	Description
Small	S1	6	24	4.0	Cyclic	Single cycle, 1 SCC
Small	S2	7	18	2.6	DAG	Purely acyclic, 7 SCCs (all size 1)
Small	S3	9	36	4.0	Mixed	Multiple SCCs, 1 large SCC
Medium	M1	12	44	3.7	DAG	Sparse DAG
Medium	M2	15	46	3.1	Mixed	Multiple SCCs (all size 1)
Medium	M3	18	152	8.4	Cyclic	Single large SCC
Large	L1	22	78	3.5	DAG	Sparse, pure DAG (22 SCCs, all size 1)
Large	L2	30	216	7.2	Mixed	High density, 1 dominant SCC (size 29)
Large	L3	45	990	22.0	DAG	Extremely dense, pure DAG (45 SCCs, all size 1)

3. Results and Metrics

The following tables summarize the performance and structural metrics for all datasets. All time measurements are in nanoseconds.

Table 3.1: SCC and Topological Sort Metrics

This table focuses on the performance of Kosaraju's algorithm for SCC detection and the subsequent topological sort. The complexity is O(V+E).

Таблица1

Dataset	# N (Nodes)	# M (Edges)	# #SCC	# Max SCC Size	# Time (ns)	# DFS Visits	# Edges (SCC)	Pushes/Pops
S1	6	24	1	6	10900	12	24	1/1
S2	7	18	7	1	22500	14	18	7/7
S3	9	36	2	8	24000	18	36	2/2
M1	12	44	6	7	35100	24	44	6/6
M2	15	46	15	1	30500	30	46	15/15
M3	18	152	1	18	44100	36	152	1/1
L1	22	78	22	1	629900	44	78	22/22
L2	30	216	2	29	62600	60	216	2/2
L3	45	990	45	1	178500	90	990	45/45

Table 3.2: Shortest/Longest Path Metrics

This table focuses on the performance of the Shortest Path algorithm on the Condensation DAG, which is also O(Ncomp +Mcomp), where Ncomp ≤N and Mcomp ≤M.

Таблица1

Dataset	# N	# M	# #SCC (Condensation)	Longest Path Value (Critical)	# Relaxations (DAG-SP)	# Checks (DAG-SP)
S1	6	24	1	0.0	0	0
S2	7	18	7	7.0	2	2
S3	9	36	2	0.0	0	0
M1	12	44	6	0.0	0	0
M2	15	46	15	7.0	2	2
M3	18	152	1	0.0	0	0
L1	22	78	22	23.0	10	12
L2	30	216	2	0.0	0	0
L3	45	990	45	88.0	139	276

4. Analysis

4.1. SCC and Topological Sort: Bottlenecks and Complexity

The time complexity of Kosaraju's Algorithm and the subsequent Topological Sort is theoretically dominated by graph traversal, O(N+M).

1. **Effect of Density (M):**
    a. Compare two dense graphs: **L2 (M=216)** and **L3 (M=990)**. L3 has ≈4.6 times more edges than L2, and its time (178,500 ns) is significantly higher than L2's (62,600 ns), clearly demonstrating that the **number of edges (M) is the primary factor** for performance in the SCC phase, adhering to the O(N+M) complexity.
2. **Anomaly in L1:**
    a. The execution time for **L1 (629,900 ns)** is exceptionally high compared to other large datasets (L2: 62,600 ns, L3: 178,500 ns), even though N+M is much smaller than L2 and L3. This suggests a potential issue with the implementation or hardware timing for this specific run (perhaps garbage

collection or context switching), as the SCC metrics (DFS Visits=44, Edges=78) are low and do not justify the observed runtime based on O(N+M). For robust analysis, such outliers should be investigated further or excluded.

3. **Metrics Correlation:**
    a. The DFS Visits metric is exactly 2N (e.g., L3: 2×45=90), confirming that the algorithm correctly performs exactly two full DFS traversals (Kosaraju's two passes).
    b. The Pushes/Pops count precisely matches the number of nodes (N) in the input graph, as every node is pushed onto and popped from the stack exactly once.

4.2. DAG Shortest/Longest Paths: Effect of Structure

The planning phase, which involves finding the critical path (Longest Path), can only be executed on the **Condensation DAG**.

1. **Impact of Cyclicality (The "Unplannable" Case):**
    a. In datasets like **S1, S3, M1, M3, and L2**, the Critical Path Value is **0.0**. This occurs because the condensation graph either contains a single, massive SCC (M3, L2) or the condensation process isolates the cyclic part, leaving no feasible path between components.
    b. For example, **L2** has a Max SCC Size of 29 out of 30 nodes, resulting in only 2 condensation nodes. If the source node for SP/LP is chosen within the large cyclic component, the calculated critical path will be 0, indicating that the **entire set of tasks is interdependent and cannot be scheduled sequentially** without cycle resolution (i.e., changing the dependencies). This represents a **critical bottleneck** for task planning.
2. **Performance on Plannable DAGs:**
    a. Datasets **S2, M2, L1, and L3** (where Max SCC Size=1) represent pure or near-pure DAGs, which are perfectly plannable.
    b. The number of Relaxations and Checks in Table 3.2 is directly proportional to the **number of edges in the Condensation DAG (Mcomp)**.
    c. **L3** is the most complex plannable case: Ncomp =45 and M≈990. The 139 Relaxations and 276 Checks quantify the computational effort required to find the critical path of **88.0**. This demonstrates that even on a pure DAG, high density significantly increases the planning time, although it remains linear O(Ncomp +Mcomp).
3. **Interpretation of Critical Path:**
    a. The Longest Path Value (Critical) of **88.0** for **L3** is the **minimum total time** required to complete all tasks, assuming infinite parallel resources and that the node/edge weights represent task/transition durations. This path defines the **bottleneck sequence** of tasks that must be prioritized to avoid delays.

5. Conclusions and Practical Recommendations

The study confirms that graph algorithms are indispensable tools for scheduling, particularly in environments with complex task dependencies.

Summary of Findings:

1. **SCC and Topological Sort:** The implementation demonstrates complexity consistent with O(N+M), where the number of edges (M) is the main determinant of performance. Kosaraju's algorithm correctly decomposes the graph into independent components for subsequent planning.
2. **Critical Path Determination:** The linear-time O(Ncomp +Mcomp) Shortest/Longest Path algorithm on DAGs is highly efficient for planning tasks **after** cycle resolution.
3. **Scheduling Bottlenecks:**
    a. **Cyclic Dependencies** (e.g., L2, M3) are the most severe planning bottleneck, leading to a critical path value of **0.0**, which signifies an unresolvable sequence of tasks.
    b. **High Density** in pure DAGs (L3) increases the computation time for the critical path but still yields a definitive schedule time (88.0).