

# Smart City / Smart Campus Scheduling — Full Project Report

**Student Name:** Bekzat Murat

**Course:** DAA

**Date:** 02.11

---

## 1. Introduction

This report presents a full pipeline implementation of graph-based planning and analytics for Smart-City and Smart-Campus operations such as street cleaning, sensor maintenance, and campus logistics. The problem involves managing task dependencies, detecting cycles, and computing efficient execution orders using advanced graph algorithms.

The project integrates:

- Strongly Connected Components (SCC)
- Condensation DAG construction
- Topological sorting
- Shortest-path analysis
- Critical-path evaluation

These techniques ensure efficient scheduling, prevent cyclic execution deadlocks, and help identify bottleneck workflows.

---

## 2. System Architecture & Workflow

The system reads task dependency data from JSON files and processes it through the following algorithmic stages:

1. **Detect SCCs** to find cycles in task dependencies
2. **Compress each SCC** into a single DAG node (condensation graph)
3. **Compute topological order** for the DAG to ensure valid execution sequence
4. **Find shortest paths** to minimize execution time of dependent tasks
5. **Compute the critical path** (longest path) to detect bottleneck operations

This pipeline ensures safe and efficient scheduling for smart-infrastructure operations.

---

## 3. Algorithms Implemented

1. Tarjan's Strongly Connected Components (SCC) Algorithm
2. Condensation Graph Construction (SCC  $\rightarrow$  DAG)

- 3. Kahn’s Topological Sort
  - 4. Dynamic Programming for Shortest Path on DAG
  - 5. Dynamic Programming for Longest Path (Critical Path)
  - 6. Performance Metrics Collection using System.nanoTime() and metric counters
- 

4. Datasets

Nine datasets were generated for testing:

Category	Count	Nodes	Description
Small	3	6–10	Simple graphs, 1–2 cycles
Medium	3	10–20	Mixed cyclic & acyclic structures
Large	3	20–50	Stress-test & performance profiling

---

5. Weight Model

- **Edge-weight** model is used (w field in JSON)
  - Parallel edges between SCCs are preserved to maintain accurate timing
- 

6. Example Execution Results

Dataset: data/small\_manual.json

- **SCC Count:** 6
  - **Example SCCs:** {1,2,3}, {0}, {4}, {5}, {6}, {7}
  - **Condensation DAG:** 6 nodes, 4 edges
  - **Topological Order:** Valid execution order generated
  - **Shortest Path (from task 4):** 4 → 5 → 6 → 7 (cost = 8)
  - **Critical Path:** 4 → 5 → 6 → 7 (bottleneck chain, cost = 8)
- 

7. Performance Analysis

Performance was analyzed across nine randomly-generated datasets. Observed time complexity trends:

Algorithm	Complexity Notes	
Tarjan SCC	$O(V + E)$	Near-linear performance
Topological Sort	$O(V + E)$	Efficient for large DAGs
DAG Shortest Path	$O(V + E)$	Fast via topo-order DP
DAG Longest Path	$O(V + E)$	Similar to shortest path

Memory usage scales with nodes + edges, suitable for practical smart-city workloads.

---

## 8. Conclusions

This system demonstrates practical graph-based scheduling for smart-city services. SCC compression enables cycle detection and resolution, while topological sorting ensures valid task execution order. Shortest- and longest-path analysis support optimal scheduling and identification of critical tasks.

The approach is scalable, modular, and suitable for automation in smart-infrastructure systems.