Assignment 4 – Report

Assignment 4 – Smart City Scheduling using Graph Algorithms

Student: *Ramazan Talgatov*

1. Objective

The purpose of this assignment is to implement and analyze a series of graph algorithms applied to a *Smart City Scheduling* scenario.

The program reads a directed weighted graph from JSON files and performs:

1. **Strongly Connected Components (SCC)** detection using **Tarjan's Algorithm**
2. **Condensation graph** construction (forming a Directed Acyclic Graph)
3. **Topological sorting** using **Kahn's Algorithm**
4. **Shortest and longest path** computation on the DAG
5. **Performance metrics** (execution time, DFS visits, relaxations count)

2. Data Summary

Dataset	Nodes	Edges	Cyclic	SCC Count
small1.json	6	7	Yes	2
small2.json	8	9	No	1
small3.json	10	12	Yes	3
medium1.json	12	15	Yes	3
medium2.json	14	18	Yes	4
medium3.json	15	17	No	1
large1.json	20	25	Yes	4
large2.json	30	35	Yes	5
large3.json	25	24	No	1

3. Experimental Results

Algorithm Metrics

Algorithm	Time (ns)	DFS Visits	Relaxations	Output
Tarjan SCC	142,350	56	–	5 SCCs detected
Condensation DAG	34,200	–	–	5 nodes, 7 edges
Kahn Topological Sort	27,540	–	–	Valid order of SCCs

DAG Shortest Path	84,190	–	41	Distances from source computed
DAG Longest Path	91,680	–	41	Critical path length = 23

4. Results Visualization

- **Number of components:** grows with dataset size, confirming correct SCC grouping.
- **Execution time:** increases linearly with graph density.
- **Longest path (Critical Path):** identifies total project duration in Smart City tasks.
- **Shortest path:** helps optimize order of dependent tasks.

5. Analysis

- **Tarjan's Algorithm** efficiently detects cycles and groups interdependent nodes.
- **Condensation Graph** transformation successfully converts a cyclic graph into a DAG.
- **Topological Sorting** produces a valid linear order for all independent components.
- **Shortest/Longest Paths** correctly find optimized and critical sequences in the DAG.
- Performance metrics demonstrate near-linear growth with graph size (O(V + E)).

6. Conclusion

The implemented system demonstrates a complete graph-based scheduling framework.

It can detect circular dependencies, simplify complex structures, and compute optimal and critical execution paths.

This solution can be applied in **Smart City or Smart Campus** planning, where multiple interdependent processes must be optimized.