

# **Smart Campus Scheduling – Report**

**Author:** Suleymenov Nurbolat

## **1. Project Overview**

This project implements a complete pipeline for scheduling smart campus tasks using graph theory algorithms in Java. It includes identification of Strongly Connected Components (SCCs), building the condensation graph, performing topological sorting, and calculating both shortest and longest paths on the DAG.

The implementation structure follows clean architecture with modularized packages for SCC, DAG shortest/longest path, graph parsing, metrics, and dataset generation.

## **2. Implemented Features**

### **Strongly Connected Components (Tarjan's Algorithm)**

- Implemented using DFS with low-link values.
- Efficiently detects cycles and forms component groups.
- Time complexity:  $O(V + E)$

### **Condensed Graph Construction**

- Collapses SCCs into one node each.
- Removes internal edges and preserves only cross-component edges.

### **Topological Sorting (Kahn's Algorithm)**

- Applied on DAG generated from condensation.
- Used as prerequisite for pathfinding algorithms.

### **Shortest and Longest Paths in DAG**

- Implemented using dynamic programming based on topological order.
- Both methods support edge-weighted graphs.

- Longest path reconstruction using parent pointers.

### ***Dataset Generation***

- 9 JSON datasets auto-generated (3 small, 3 medium, 3 large) with varied density and cycle support.

### ***Metrics Module***

- Tracks execution time and operation counters using System.nanoTime().

### ***JUnit Tests***

- Unit tests implemented for core modules (SCC and DAG shortest path).

## **3. Project Structure**

```
/src
  └── graph/
    ├── scc/      # Tarjan SCC implementation
    ├── cond/     # Condensed graph builder
    ├── topo/     # Topological sort
    ├── dagsp/    # Shortest & longest path on DAG
    └── gen/      # Dataset generator
    └── app/Main.java # Entry point
    └── test/     # JUnit tests
```

Build tool: **Maven**

Java version: **17**

## **4. Key Results**

- SCC detection correctly identifies cycles and isolates independent components.
- Condensation output is always a valid DAG (Directed Acyclic Graph).
- Shortest path returns minimal cumulative edge weights from source.
- Longest path tracing works as expected based on topological ordering.
- Dataset generator produces varied and realistic test graphs with support for cycles and DAGs.

## **5. How to Run**

```
mvn clean package  
java -cp target/smart-campus-1.0-SNAPSHOT.jar app.Main  
data/tasks.json
```

Or test the generated datasets inside data/.

## **6. Conclusion**

This project fully meets the assignment requirements, combining theoretical algorithms with real-world practical implementation. The modularized design, testing, dataset generation, and metrics make the solution robust and extendable.