Assignment 4 — SCC, Topological Ordering, DAG Shortest/Longest Paths

1. Purpose

This project implements the full pipeline required by the "Smart City / Smart Campus Scheduling" assignment:

1. Find strongly connected components (SCC) with Tarjan.
2. Build condensation graph (SCC → DAG).
3. Topologically sort the DAG (Kahn).
4. Run shortest-path and longest-path (critical path) DP on the DAG.
5. Log metrics (time, ops) for every dataset under `/data/`.

All runs are automated by the integration test `GraphAlgorithmsIntegrationTest` which iterates over all `*.json` in `/data` and writes: - `data/output.json` — structured JSON report per dataset - `data/metrics.csv` — tabular metrics

This README describes the actual code in the repository, not a template.

---

2. Project Structure

```
src/
   main/java/
        Main.java
        graph/scc/
              TarjanSCC.java
              CondensationBuilder.java
        graph/topo/
              KahnTopologicalSort.java
        graph/dagsp/
              DAGShortestPath.java
              DAGLongestPath.java
        graph/util/
               SCCUtils.java

        metrics/
               Metrics.java
               MetricsTracker.java
   test/java/
         GraphAlgorithmsIntegrationTest.java

data/
   small1.json
   small2.json
   small3.json
```

```
medium1.json
medium2.json
medium3.json
large1.json
large2.json
large3.json
metrics.csv        ← generated
output.json        ← generated
```

---

3. Data Summary

file	n	edges	density	weight_model	source
small1.json	6	6	medium	edge	0
small2.json	7	7	medium	edge	0
small3.json	8	9	medium	edge	0
medium1.json	12	13	sparse	edge	0
medium2.json	15	17	sparse	edge	0
medium3.json	18	19	sparse	edge	0
large1.json	20	39	sparse	edge	0
large2.json	24	58	sparse	edge	0
large3.json	30	94	sparse	edge	0

---

4. Results

SCC (Tarjan)

file	vertices	edges	Tarjan_SCC_count	max_scc_size	Tarjan_time_ms
large1.json	20	39	12	9	0.096
large2.json	24	58	7	18	0.026
large3.json	30	94	30	1	0.040
medium1.json	12	13	10	3	0.008
small1.json	6	6	6	1	0.004

Topological Sort (Kahn)

file	vertices	edges	Kahn_time_ms
large3.json	30	94	0.041
large2.json	24	58	0.010

file	vertices	edges	Kahn_time_ms
small1.json	6	6	0.004

DAG Shortest / Longest Path

file	vertices	edges	Shortest_time_ms	Longest_time_ms	Longest_value
large3.json	30	94	0.025	0.029	53
large1.json	20	39	0.025	0.020	26
small1.json	6	6	0.002	0.002	16

---

5. Analysis

- **Tarjan SCC:** linear O(V+E); stable recursion; `large2.json` shows condensation of size 7 from 24 nodes.
- **Kahn Sort:** linear; handles dense DAGs efficiently; queue ops proportional to node count.
- **DAG-SP:** relaxation proportional to edges; longest path critical chain = 53 (`large3.json`).

---

6. Conclusions

1. Run SCC first, then condensation + topo + DAG-SP.
2. Condensation drastically reduces DAG size when cycles exist.
3. DAG-SP efficiently computes both shortest and longest (critical) paths.
4. Metrics confirm linear behavior across datasets.

---

7. Run Instructions

```
mvn clean package
java -cp target/daa-4-1.0.0.jar Main data/small1.json
mvn -Dtest=GraphAlgorithmsIntegrationTest test
```

Outputs: - `data/output.json` - `data/metrics.csv`

---

8. Figures

SCC decomposition of large2.json — one dominant component of size 18.

```
"file" : "large2.json",
"scc" : [
  { "id": 0, "size": 1, "vertices": [12] },
  { "id": 1, "size": 1, "vertices": [16] },
  { "id": 2, "size": 1, "vertices": [23] },
  { "id": 3, "size": 1, "vertices": [0] },
  { "id": 4, "size": 1, "vertices": [11] },
  {
    "id": 5,
    "size": 18,
    "vertices": [21,17,8,2,19,18,14,15,6,5,4,1,13,7,22,10,9,3]
  },
  { "id": 6, "size": 1, "vertices": [20] }
]
```

Topological order of large3.json (each vertex forms its own SCC).

```
"file" : "large3.json",
"Tarjan_SCC_count": 30,
"componentTopo": [0,1,2,3,4,5,7,6,14,8,12,9,15,10,11,13,18,28,29,19,26,16,17,20,21,22,23,24,
```

Longest (critical) path in large3.json with total length 53.

```
"criticalPath" : {
  "length": 53,
  "path": [0, 1, 4, 6, 15, 17, 27, 29]
},
"DAGSP_long_max": 53
```

Small SCC of size 3 in small2.json.

```
"file" : "small2.json",
"scc": [
  { "id": 0, "size": 1, "vertices": [6] },
  { "id": 1, "size": 1, "vertices": [5] },
  { "id": 2, "size": 1, "vertices": [4] },
  { "id": 3, "size": 1, "vertices": [3] },
  { "id": 4, "size": 3, "vertices": [2, 1, 0] }
]
```

---

9. References

- Astana IT University — DAA Course Materials. *Lecture slides.* https://lms.astanait.edu.kz
- GeeksforGeeks. *Tutorials on Tarjan's Algorithm, Topological Sort (Kahn's Algorithm), and Shortest Path in DAG.* https://www.geeksforgeeks.org/
- ChatGPT (OpenAI, 2025). *generating example JSON* https://chatgpt.com