SE-2407
Damir Ushkempir

Goal

This project integrates two key graph topics into one practical case:

1. Strongly Connected Components (SCC) and Topological Ordering
2. Shortest Paths in Directed Acyclic Graphs (DAGs)

The application models scheduling of dependent city tasks (e.g., repairs, maintenance, analytics).

Implemented Modules

Module	Purpose	Algorithm
graph.scc.TarjanSCC	Find strongly connected components	Tarjan (DFS-based)
app.CondensationBuilder	Build component condensation (DAG)	Custom
graph.topo.Topological	Topological ordering of condensation	Kahn’s algorithm
graph.dagsp.DAGSP	Shortest & Longest paths in DAG	DP over topological order
util.Metrics	Instrumentation (time + operation counters)	Custom class

Data Summary

The dataset is located in /data/tasks.json.

Property	Value
Directed	true
Vertices (n)	8
Edges (m)	7
Weight model	Edge weights (w)
Source node	4
Description	Two disconnected parts: (0–3 cycle) + (4–7 linear DAG)

Edge list:

From	To	Weight
0	1	3
1	2	2
2	3	4
3	1	1
4	5	2
5	6	5
6	7	1

Instrumentation

Implemented through `util.Metrics` and `System.nanoTime()`.

Metric	Meaning	Counted in
dfsVisits	DFS calls	SCC (Tarjan)
dfsEdges	Edges processed	SCC (Tarjan)
kahnPushes	Queue insertions	Topological Sort
kahnPops	Queue removals	Topological Sort
relaxations	Relax steps	DAG Shortest Path
timeNano	Total time	All modules

Example runtime output:

```
Metrics: {timeNano=3400000, dfsVisits=8, dfsEdges=7, kahnPushes=5, kahnPops=6, relaxations=6}
```

Results (for tasks.json)

SCC and Condensation

Component	Vertices	Size	Type
C0	[1, 2, 3]	3	Cyclic
C1	[0]	1	Isolated
C2	[4]	1	DAG
C3	[5]	1	DAG
C4	[6]	1	DAG
C5	[7]	1	DAG

Total SCCs: 6 (1 cyclic + 5 acyclic)

Condensation edges:

- C1 → C0
- C2 → C3 → C4 → C5

Topological Order

```
[ C1, C0, C2, C3, C4, C5 ]
```

→ Derived task order: `[0, 1, 2, 3, 4, 5, 6, 7]`

Shortest Paths (from source = 4)

Target	Distance	Path
4	0	[4]
5	2	[4 → 5]
6	7	[4 → 5 → 6]

Target	Distance	Path
7	8	[4 → 5 → 6 → 7]

Longest Path (Critical Path)

Start	End	Length	Path
4	7	8	4 → 5 → 6 → 7

Metrics Summary

Metric	Value
DFS Visits / Edges	8 / 7
Kahn Push / Pop	5 / 6
Relaxations	6
Execution time	≈ 3.4 ms

Analysis

SCC Stage

- Detected 1 cycle (1–2–3) and 5 simple nodes.
- Condensation DAG simplified structure and split graph into two independent regions.
- DFS edge operations scale linearly (O(V + E)).

Topological Ordering

- Kahn's algorithm completed in linear time.
- Queue operations (kahnPush/kahnPop) proportional to edges.

DAG Shortest / Longest Paths

- Shortest path DP performs linearly in edges.
- Longest path found efficiently with max-DP (same complexity).
- Both share the same critical path (4→5→6→7).

Observed bottlenecks:

- Recursive depth of Tarjan for large SCCs.
- Relaxation loops for dense graphs.

Conclusions

Method	When to use
SCC (Tarjan/Kosaraju)	Detect cycles and compress them to simplify dependency graphs.

Method	When to use
Topological Sort (Kahn)	Determine safe execution order of independent components.
DAG Shortest Path	Find minimum-time schedule from a given task.
DAG Longest Path (Critical Path)	Identify bottleneck sequence of tasks in planning.