CMPE 223 Data Structures and Algorithms 2

Homework 2

10/10/2025-26/10/2025

Name: Mehmet Yiğit

Surname: Açdoyuran

ID Number: 10130437010

Section: 01

Assignment: 02

Department: Department of Computer Engineering

Name: Yusuf

Surname: Karabey

ID Number: 14857072936

Section: 02

Assignment: 02

Department: Department of Computer Engineering

Problem Statement & Code Design

Q1: Smart Factory (Cycle Witness/Topological Order). Determine whether a digraph of task dependencies is a DAG. If so, print one directed cycle; if not, print a valid topological order.

Q2: Intercloud (Strengthen & SCCs). All Strongly Connected Components (SCCs) should be calculated. If there are many SCCs, produce one valid set of new links and print the bare minimum of links needed to make the entire network tightly connected.

Principles of design

1) Adjacency-list representation; no external libraries.
2) The autograder is satisfied by deterministic outcomes.
3)I/O, graph operations, and printing are clearly separated by linear-time algorithms.

Structure chart

Q1 Main → readInput → buildDigraph → dfsCycleOrNone
→ if cycle: printCycle
→ else: kahnLexTopo → printOrder

Q2 Main → readInput → buildDigraph(+reverse)
→ kosarajuSCC → printComponents
→ if |SCC|==1: print "Already strongly connected; 0"
→ else: sourcesSinks → printNeededAndEdges

Implementation & Functionality

Q1: Topological cycle or order
I/O. Line N M comes first, followed by M edges u v, which means u→v.

Algorithms.

1) DFS on-stack cycle identification. state: parent[] for reconstruction, 0/1/2. Climb parent[u] up to v on back edge u→v (state[v]=1) to get v ... u, then append v to end the cycle.
2) Kahn with min-heap (in the absence of a cycle). Calculate indegrees, push all 0 indegree vertices to a min-priority queue, and then continually push newly zero vertices, relax neighbors, and pop the smallest label. This results in the smallest topological order lexicographically.

Output.

Schedulable
Order: t0 t1 ... tN-1
or
Not schedulable
Cycle: v0 v1 ... v0
Complexity. O(N+M) time, O(N+M) space.

Q2: SCCs and creating a strongly connected graph
I/O. Q1's edge format.

Algorithms.

1) Kosaraju–Sharir (two passes). To obtain reverse-postorder, reverse-graph DFS; to label SCC IDs (comp[]), second pass DFS on the original graph in that order. Keep the vertices of each component arranged in ascending order.
2) Condensation DAG statistics. Increase outdeg[comp[u]] and indeg[comp[v]] for every edge u→v with comp[u]≠comp[v]. Let T = #sinks (outdeg=0) and S = #sources (indeg=0).
3) Very few links. The number of new edges needed is max(S,T) if C>1.

4) Deterministic constructive set (precise grader). Sort the sink and source components according to the minimum vertex of each. Output an edge from the maximum vertex of sink[i] to the **minimum vertex of source[(i+1) mod S]` (ring) for i=0..max(S,T)-1.

Output.
1) For every SCC (sorted by smallest member): Part I: v1 v2... vk
2) If a single SCC is already closely connected and new edges are required: 0
3) If not, new edges are required: K, then add: a b lines.
Complexity. O(N+M) time, O(N+M) space.

Testing

Objectives. Discuss determinism, cycles, DAGs, and numerous SCCs.

Exemplary situations.
1) Q1 (DAG). Example from prompt → Order: 0 1 2 3 4 5 (smallest lexicographically).
2) Q1 (Cycle). 0→1→2→0 → Cycle: 0 1 2 0 is not schedulable (any rotation is permitted).
3) Q2 (Many SCCs). components written in the following order: {0,1,2}, {3,4,5}, {6}, {7} → New edges are required: 2 with add: 5 6, add: 7 0.
4) Q2 (One SCC). 0→1→2→3→0 → one component; 0 new edges are required.

Format/bug guards. robust against out-of-range lines and duplicate edges; output labels and spacing are precise.

Final Assessments

1) Problem areas. achieving clean cycle reconstruction and grader-specific determinism (Q1's PQ order; Q2'nin ring eşlemesi).
2) most difficult. applying the max(S,T) rule and condensation DAG reasoning; accurately matching the output format to the specification.
3) What I discovered. DFS cycle detection, Kahn's method, and Kosaraju SCCs were practiced; the concepts of minimal reinforcement and strong connection were reinforced.
4) AI resources (if any). used ChatGPT to test concepts and talk about deterministic conventions. I rewrote and validated all of the code, and I am able to explain each step; my comprehension is reflected in the submission.
5) Ethics and impact. AI assisted in organizing ideas and identifying formatting errors, but I was ultimately responsible for debugging and making judgments. Citing help, avoiding mindless copying and pasting, and turning in work that I can defend on my own are all examples of ethical use.

Appendix

Q1:
```
for v: if state[v]==0 dfs(v)
if cycleFound: print cycle
else: Kahn+minHeap → print order
```
Q2:
```
order = DFS(G^R)
label SCCs by DFS(G) in reverse(order)
sources = {cid|indeg[cid]==0}, sinks = {cid|outdeg[cid]==0}
if C==1: print 0
else: K=max(|sources|,|sinks|)
for i in 0..K-1:
print max(sink[i]) → min(source[(i+1)%|sources|])
```