Problem Set 9

Data Structures and Algorithms, Fall 2022

Due: November 24 (in class).

Problem 1

The square of a directed graph $G = (V, E)$ is the graph $G^2 = (V, E^2)$ such that $(u, v) \in E^2$ if and only G contains a path with *at most* two edges between u and v. Devise the most efficient algorithms you can come up with for computing G^2 from G for both the adjacency-list and adjacency-matrix representations of G. You also need to analyze the running times of your algorithms.

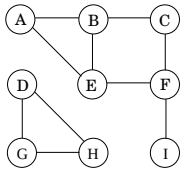
Problem 2 [OJ Problem]

(Solve this problem on the OJ system. Do not hand in written solutions!)

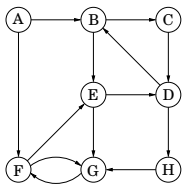
Given the adjacency-matrix representation of a directed graph $G = (V, E)$, devise an algorithm that can determine whether G has a special sink node that has in-degree $|V| - 1$ and out degree 0. You algorithm should have running time $O(|V|)$. (This is interesting as most graph algorithms that take an adjacency-matrix representation as input require running time $\Omega(|V|^2)$.)

Problem 3

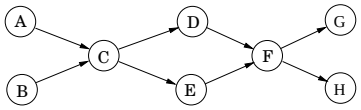
(a) Perform a BFS traversal on the following graph. Whenever there's a choice of vertices, pick the one that is alphabetically first. Show the $distance$ value and the $parent$ value for each vertex.



(b) Perform a DFS traversal on the following graph. Whenever there's a choice of vertices, pick the one that is alphabetically first. Show the classification of edges, and give the start/finish times of vertices.



(c) Run the DFS-based topological sorting algorithm on the following graph. Whenever there's a choice of vertices, pick the one that is alphabetically first. Show the final ordering.



Problem 4

(a) Give an example of a directed graph $G = (V, E)$, a source vertex $s \in V$, and a set of tree edges $E_{tree} \subseteq E$ such that for each vertex $v \in V$, the unique simple path in the graph (V, E_{tree}) from s to v is a shortest path in G, yet the set of edges E_{tree} cannot be produced by running BFS on G, no matter how the vertices are ordered in each adjacency list.

(b) Give a counterexample to the conjecture that if a directed graph G contains a path from u to v, and if $u.d < v.d$ in a depth-first search of G, then v is a descendant of u in the depth-first forest produced.

(c) Prove or disprove: If a directed graph G contains cycles, then the DFS-based topological sorting algorithm introduced in class produces a vertex ordering that minimizes the number of "bad" edges that are inconsistent with the ordering produced.

Problem 5 [OJ Problem]

(Solve this problem on the OJ system. Do not hand in written solutions!)

You are given a binary tree $T = (V, E)$ in adjacency list format, along with a designated root node $r \in V$. You wish to pre-process the tree so that queries of the form "is u an ancestor of v?" can be answered in constant time. Devise a pre-processing algorithm that takes only $O(|V| + |E|)$ time.

Problem 6

Suppose the AI curriculum at NJU consists of n courses, and all of them are mandatory. The prerequisite graph G has a node for each course, and an edge from course v to course w if and only if v is a prerequisite for w. Devise an algorithm that works directly with this graph representation, and computes the minimum number of semesters necessary to complete the curriculum (assume that a student can take any number of courses in one semester). The running time of your algorithm should be $O(n)$.

Problem 7

"Snakes and Ladders" is a classic board game, originating in India no later than the 16th century. The board consists of an $n \times n$ grid of squares, numbered consecutively from 1 to n^2, starting in the bottom left corner and proceeding row by row from bottom to top, with rows alternating to the left and right. Certain pairs of squares in this grid, always in different rows, are connected by either "snakes" (leading down) or "ladders" (leading up). Each square can be an endpoint of at most one snake or ladder.

You start with a token in cell 1, in the bottom left corner. In each move, you advance your token *up to* k positions, for some fixed constant k. If the token ends the move at the *top* end of a snake, it slides down to the bottom of that snake. Similarly, if the token ends the move at the *bottom* end of a ladder, it climbs up to the top of that ladder.

Given n and k as input, devise an algorithm to compute the smallest number of moves required for the token to reach the last square of the grid.

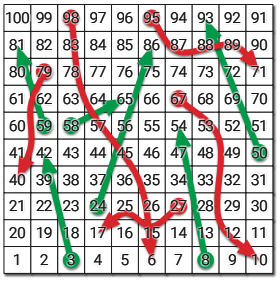


Figure 1: A Snakes and Ladders board. Upward straight arrows are ladders; downward wavy arrows are snakes.