COL351: Analysis and Design of Algorithms **Fall 2024**

Tutorial Sheet 4

Announced on: Aug 15 (Thurs)

Problems marked with ($\star$) will not be asked in the tutorial quiz.

1. Consider running the Topological-Sort-via-DFS algorithm on a directed graph G that is not acyclic. The algorithm will not compute a topological ordering (as none exists). Does it compute an ordering that minimizes the number of edges that travel backward? Justify your answer.

2. The game of Snakes and Ladders has a board with n cells where you seek to travel from cell 1 to cell n. To move, a player throws a six-sided dice to determine how many cells forward they move. The board also contains snakes and ladders that connect certain pairs of cells. A player who lands on the mouth of a snake immediately falls back down to the cell at the other end. A player who lands on the base of a ladder immediately travels up to the cell at the top of the ladder. Suppose you have rigged the dice to give you full control of the number for each roll. Design an efficient algorithm to find the minimum number of dice throws to win.

3. Let u and v be two vertices in a directed graph $G = (V, E)$. Design a linear-time algorithm to find the number of distinct shortest paths (not necessarily vertex disjoint) between u and v.

4. A directed graph $G = (V, E)$ is *weakly connected* if for every pair of vertices (u, v), there is a path from u to v or from v to u or bothways. Design a linear time algorithm to check if G is weakly connected.

5. A *vertex cover* of an undirected graph $G = (V, E)$ is a subset of vertices $U \subseteq V$ such that each edge in E is incident to at least one vertex of V. Design an efficient algorithm (using BFS or DFS) to find a minimum-size vertex cover if G is a tree.

6. An *independent set* of an undirected graph $G = (V, E)$ is a subset of vertices $U \subseteq V$ such that no edge in E is incident to two vertices of U. Design an efficient algorithm (using BFS or DFS) to find a maximum-size independent set if G is a tree.

7. Given an undirected graph G, an *independent vertex cover* of G is a subset of vertices that is both an independent set and a vertex cover of G. Design an efficient algorithm for testing whether G contains an independent vertex cover.

8. ($\star$) Given an undirected graph $G = (V, E)$, a vertex $v \in V$ is said to be an *articulation point* if $G \setminus \{v\}$ has more connected components than G. Design an $\mathcal{O}(|V| + |E|)$ algorithm to find all articulation points of G.

9. ($\star$) In the 2SAT problem, you are given a set of clauses, each of which is the disjunction (logical "`OR`") of two literals. (A literal is a Boolean variable or the negation of a Boolean variable.) You would like to assign a value `TRUE` or `FALSE` to each of the variables so that all the clauses are satisfied, with at least one true literal in each clause. For example, if the input contains the three clauses $x_1 \vee x_2$, $\neg x_1 \vee x_3$, and $\neg x_2 \vee \neg x_3$, then one way to satisfy all of them is to set x_1 and x_3 to "`TRUE`" and x_2 to "`FALSE`".

   Design an algorithm that determines whether or not a given 2SAT instance has at least one satisfying assignment. (Your algorithm is responsible only for deciding whether or not a satisfying assignment exists; it need not exhibit such an assignment.) Your algorithm should run in $\mathcal{O}(m + n)$ time, where m and n are the number of clauses and variables, respectively.

   [Hint: Show how to solve the problem by computing the strongly connected components of a suitably defined directed graph.]