CS 344: Design and Analysis of Computer Algorithms

Instructor: Kangning Wang

Homework 5: Due on **May 3, 2025**

Acknowledgment: **Our team members are Alexander Lin (al1655), Pranav Tikkawar (pt422), and Ivan Zheng (iz72)** .

Problem 1

We have a rooted tree with n vertices. The height of the tree is h. We want to create a data structure that can quickly find the *lowest common ancestor* of two queried vertices.

We say that a vertex u is an ancestor of a vertex v if u is on the simple path from v to the root (including v and the root). The lowest common ancestor of two vertices is the deepest vertex that is an ancestor of both of the two vertices.

We can compute the depth of all the vertices in $O(n)$ time using DFS or BFS. In addition, we want to record $f(v, k)$ for each vertex v and each $k \in \{0, 1, \ldots, \lfloor \log_2 h \rfloor\}$, where $f(v, k)$ is the 2^k-th ancestor of the vertex v. (The 0-th ancestor of v is v itself, and the $(k+1)$-st ancestor of v is the parent of the k-th ancestor of v.)

Explain how to compute all of $f(v, k)$ in $O(n \log h)$ time and how to use this information (along with the depths of all vertices) to answer any query in $O(\log h)$ time.

Solution. Here is my solution.

Problem 2

We have an ordered sequence of n operations and a robot that performs these operations. There are two types of operations:

- Forward: Move one step in the direction it faces.
- Turn: Turn $180°$ (and face the opposite direction).

We are given a parameter k, and we can choose to modify at most k operations in the sequence (Forward to Turn, or Turn to Forward). What is the farthest place (farthest in terms of the distance to the starting point) that the robot can arrive at after performing the entire modified operation sequence? Give a polynomial-time algorithm to solve this problem.

Solution. Here is my solution.

Problem 3

Consider the following linear-time algorithm that finds a longest simple path on any tree. We assume that the edges do not have weights for simplicity, but the algorithm also works if the edges have positive weights.

- Pick an arbitrary vertex u.
- Use DFS or BFS to find a vertex v that is the farthest away from u.
- Use DFS or BFS to find a vertex w that is the farthest away from v.
- Output the simple path from v to w.

Prove the correctness of this algorithm.

Solution. Here is my solution.

Problem 4

Give a polynomial-time algorithm to find all strongly connected components in a given directed graph $G = (V, E)$. Two vertices u and v are in the same strongly connected components if and only if there is a walk from u to v and there is a walk from v to u.

In fact, this problem can be solved in $O(|V| + |E|)$ time. As a voluntary challenge, look up and understand such an algorithm.

Solution. Here is my solution.

Problem 5

Give a polynomial-time algorithm to solve the 2SAT problem. The 2SAT problem is similar to the 3SAT problem except that every clause of it has exactly 2 literals. (Hint: The 2SAT problem is related to strongly connected components.)

Solution. Here is my solution.