

Due in class: Feb 14

Warning: some of the problems require thought - do not wait until the last day to start working on them!

If you cannot come up with algorithms that run in the required time, then provide (correct) slower algorithms for partial credit. Write your answers using *pseudo-code* in the same style as the textbook. These make the algorithm description precise, and easy to read (as opposed to code in C or some other language).

1. You are given a directed graph $G = (V, E)$ in adjacency list format. Suppose the vertices are numbered as $1, 2, \dots, |V|$. You want to rearrange the adjacency list $Adj(u)$ of all vertices u such that $Adj(u)$ is in sorted order for all u . Give a linear-time algorithm for this problem. (**Hint:** Consider the graph obtained by reversing the directions of all edges in G .)
2. You are given a connected, undirected graph $G = (V, E)$ in adjacency list format. Give a linear-time algorithm to compute a path in G that traverses each edge in E exactly once in each direction.
3. Disprove the following: for any directed graph G , if there is a path from u to v , then any depth-first traversal of G must result in $d[v] \leq f[u]$.
4. (**Only for graduate students**) You are given a directed *acyclic* graph G , and two vertices u and v in G . Give a linear-time algorithm to find the number of paths from u to v in G .