Problem 1 (25 points)

The transpose of a directed graph $G = (V, E)$ is the graph $G^T = (V, E^T)$, where $E^T = (v, u) \in V \times V : (u, v) \in E$. Thus, $G^T$ is $G$ with all its edges reversed.

(a) Describe efficient algorithms for computing $G^T$ from $G$, for both the adjacency-list and adjacency-matrix representations of $G$.

(b) Analyze the running times of your algorithms.

Problem 2 (30 points)

(a) Draw the depth-first search forest. (Suppose we start from node $a$, and explore edges from left to right.)

(b) On the original graph, label nontree edges B, C, or F according to whether they are back, cross, or forward edges.

(c) Write down the nodes in order of decreasing $f[u]$ for each vertex $u$, where $f[u]$ is the finishing time of $u$ in a depth-first search.

(d) Find the strongly connected components.
Problem 3 (25 points)

(a) Consider the bipartite graph above. The thick edges indicate a maximal matching. Find an augmenting path that will transform the maximal matching into a maximum matching.

(b) Find a cut whose size equals the size of the maximum matching you found in (a).

Problem 4 (20 points)

Usually there are many different topological sorts of a dag (directed acyclic graph) \( G \). The \textsc{Topological-Sort} algorithm presented in class produces an ordering that is the reverse of the depth-first finishing times. Now show that not all topological sorts can be produced: there exists a dag \( G \) such that one of the topological sorts of \( G \) cannot be produced by \textsc{Topological-Sort}, no matter what adjacency-list structure is given for \( G \).

Problem 5 (20 points)

Prove that a connected directed graph has an \textsc{Euler Tour} if and only if each vertex in the graph has its in-degree equal to its out-degree. An Euler tour is a cycle that uses each edge exactly once (but it is allowed to visit vertices multiple times).