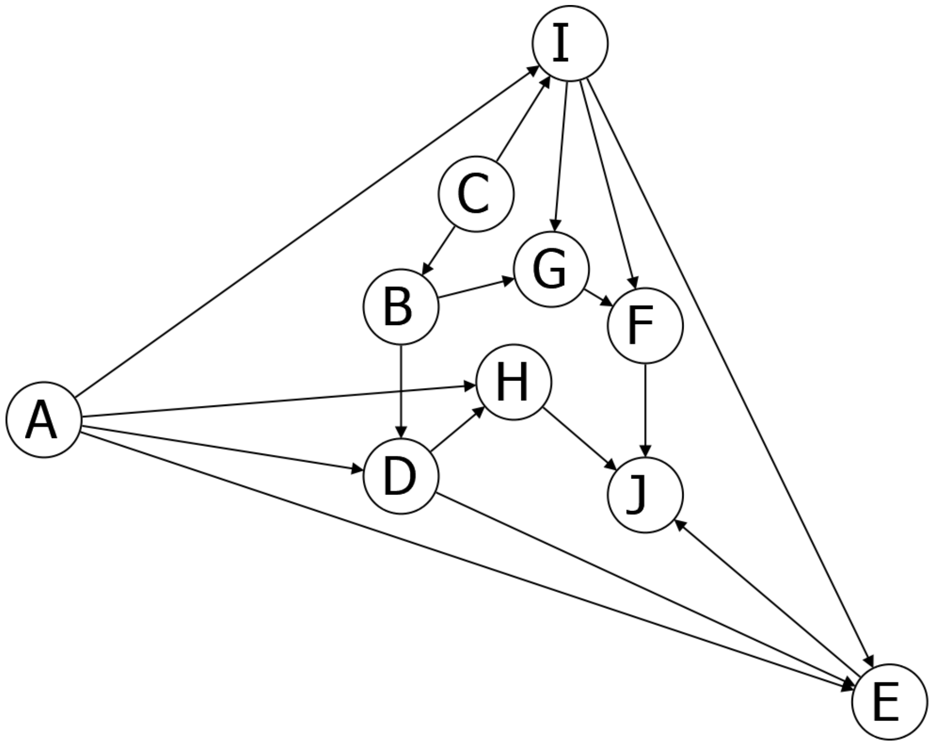
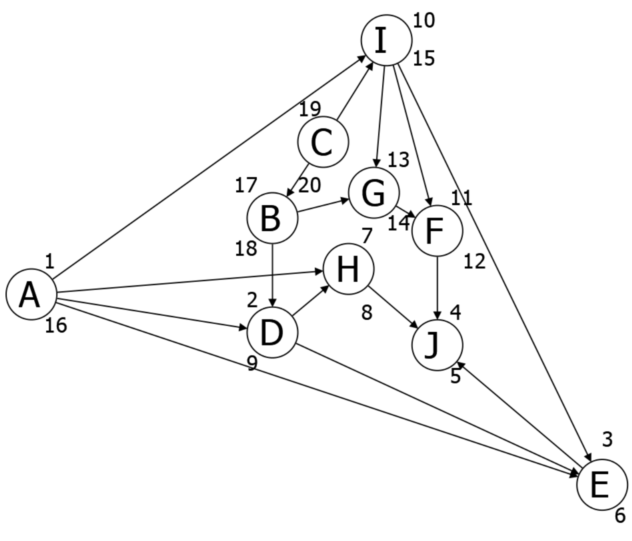
CSE 101 Exam 1 Solutions

Winter 2021

Question 1 (Topological Sort, 30 points)**.** *Give a topological ordering of the graph below:*



We compute DFS with pre- and post- orders:



We then put the vertices in reverse post-order. Namely: C, B, A, I, G, F, D, H, E, J. **Note:** Other correct solutions are possible.

Question 2 (Coming Together, 35 points). *Rose and Greg are stranded on opposite sides of Graphania. The map of Graphania is given by a directed graph where each edge in the graph takes 1 minute to traverse. Each minute each of the two travellers can either traverse an edge or stay where they are. Give an algorithm that given the map of Graphania along with the initial locations of Rose and Greg computes the minimum amount of time required for them to meet at the same location. For full credit, your algorithm should run in linear time or better.*

We first run BFS starting from Rose's location computing distances $d_R(v)$ for each vertex v. We then run BFS from Greg's location computing distances $d_G(v)$. For each vertex v we then compute $\max(d_R(v), d_G(v))$ and return the smallest value obtained.

The runtime is $O(|V| + |E|)$ to run BFS twice and then $O(|V|)$ to compute $\max(d_R(v), d_G(v))$ for each vertex and find a minimum. So the runtime is $O(|V| + |E|)$.

To show correctness, we note that $d_R(v)$ is the minimum number of minutes that it takes Rose to get to vertex v and that $d_G(v)$ is the minimum number of minutes needed for Greg to reach this vertex. Therefore, it takes them at least $\max(d_R(v), d_G(v))$ minutes to meet at vertex v. So the time it takes for them to meet is at least the minimum of this over all vertices. On the other hand, it is not hard to see that they can meet in vertex v in time $\max(d_R(v), d_G(v))$ as Rose can get there in time $d_R(v)$ and Greg in time $d_G(v)$ and if one arrives early, they can wait for the other.

Question 3 (Cycle Finding, 35 points)**.** *Provide an algorithm that given a directed graph G finds a cycle in G if one exists, and prove the correctness of your algorithm. For full credit, your algorithm should run in linear time or better.*

First, compute the strongly connected components of G. Find some edge (v, w) with both v and w in the same component (if there are no such edges, return that G has no cycles). Run explore to find a path P from w to v in G and append the edge (v, w) to it to obtain a cycle.

For the runtime, note that we can compute the SCCs in time $O(|V| + |E|)$. We can check each edge to see if its endpoints are in the same component in time $O(|E|)$. Finding such an edge, we can run explore in time $O(|V| + |E|)$. Thus, the final runtime is $O(|V| + |E|)$.

To prove correctness, note first that if G has a cycle all vertices in the cycle must lie in the same SCC (since following the cycle can get you from any vertex in the cycle to any other and back). Thus, any edge of this cycle must connect two vertices in the same SCC. Therefore, if no such edge (v, w) is found, G must have no cycles. If we do find such an edge, since v and w are in the same SCC, there must be a path from w to v, which will be found by our call to explore. Adding the edge (v, w) to this path completes the loop and creates a cycle.