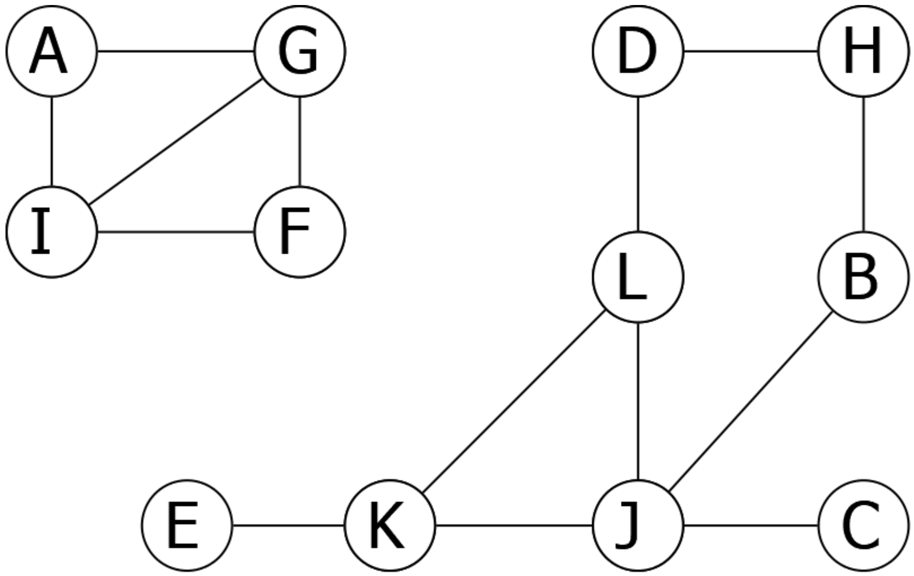
Question 1 (DFS, 30 points). *Compute the pre- and postorders of all vertices when DFS is run on the graph below. Whenever DFS has several options of which order to do things in, always visit the alphabetically first vertex first.*



Question 2 (3SAT to MIS Reduction, 30 points). *Using the reduction discussed in class, what Maximum Independent Set problem can be used to solve the 3SAT instance*

$$(x \vee y \vee z) \wedge (\bar{x} \vee w \vee \bar{z}) \wedge (\bar{x} \vee \bar{z} \vee \bar{w}) \wedge (x \vee \bar{y} \vee w)$$

and how would you interpret the answer to solve the original 3SAT?

Question 3 (Finding Close Points, 35 points). *Let L be a list of n different numbers in $[0,1]$. Give a divide and conquer algorithm that finds a pair of numbers in L that differ by at most $1/(n-1)$ (if more than one pair exists, you only need to return one of them). For full credit your algorithm should run in time $O(n)$ and use divide and conquer.*
Hint: You may want to generalize this so that for any list of n elements in an interval $I = [a,b]$, you can find two elements that differ by at most $(b-a)/(n-1)$.

Question 4 (Marked Vertex Shortest Paths, 35 points). *Let G be an undirected graph with non-negative edge weights and two vertices s and t. Some of the vertices of G including s and t are marked as special. Your goal is to find the length of the shortest $s - t$ path in G so that this path doesn't have more than three non-special vertices in a row.*

For full credit, your algorithm should run in time $O(|V| \log(|V|) + |E|)$.

Question 5 (Small Gap Interval Cover, 35 points). *Let $I = [a, b]$ be an interval of real numbers and let S be a set of n subintervals of I. Design an algorithm that given I and S determines whether or not there is a subset T of S so that:*

1. *No two intervals in T overlap.*
2. *The intervals in T cover all of I except for a number of gaps of length at most 1. In particular the gaps between consecutive elements of T and between the endpoints of I and the closest elements of T are all of length at most 1.*

For full credit, your algorithm should run in time $O(n \log(n))$ or better.

Question 6 (Double Weight MST, 35 points). *In the* `Double-Weight-MST` *problem, you are given a graph G where each edge e is assigned two weights $w_1(e)$ and $w_2(e)$. For any spanning tree T we define two weights $w_1(T)$ and $w_2(T)$ to be the sum of $w_1(e)$ over all edges $e \in T$ and the sum of $w_2(e)$ over these edges, respectively. The goal is to find the tree T so that $\max(w_1(T), w_2(T))$ is as small as possible. Prove that* `Double-Weight-MST` *is NP-Hard.*
Hint: Construct a G so that $w_1(T) + w_2(T)$ is the same no matter what T you pick. Therefore, maximizing their product will require finding a T with $w_1(T) = w_2(T)$.