CS 6212 Algorithms December 17, 2021
Youssef

FINAL EXAM
Duration: 2.5 hours

Problem 1: (25 points)
Let G(V,E) be this undirected graph: V = {1, 2, 3, … , 11}, E = {(1,2) , (1,3), (2,3) , (1,4) , (1,5), (4,5) , (4,6) , (5,6) , (5,7) , (5,8) , (5,11) , (7,8) , (8,11) , (8,9) , (8,10) , (9,10)}. Note: In the BFS and DFS below, break ties by always visiting the <u>smallest</u> unvisited neighbor first.

a) Perform BFS on G from node 1, showing the BFS tree.
b) Perform DFS on G from node 1, showing the tree and backward edges, compute the L and DFN of every node, identify the articulations points, and indicate the reason why those nodes are articulation points.

Problem 2: (25 points)
Let $G = (V, E)$ be a bipartite graph. That is, $V = A \cup B$ the union of two non-overlapping sets A and B, where every edge is between a node in A and a node in B. Assume that A has n nodes, B has m nodes, and $n \leq m$. A *complete matching* in G is any set of <u>n edges</u> in G where <u>no two edges share a node</u>. Write a Backtracking algorithm to generate all the complete matchings in G.

Problem 3: (25 points)
Let $G = (V, E)$ be a bipartite graph as in problem 3, but this time it is a weighted graph. The weight of a complete matching is the sum of the weights of its edges. We are interested in finding a minimum-weight complete matching in G.

a) Give a legitimate $\hat{C}$ for a branch-and-bound (B&B) algorithm that finds a minimum-weight complete matching in G, and prove that your $\hat{C}$ is valid. <u>Your $\hat{C}$ **cannot** be just the *cost so far*</u>.
b) Using your $\hat{C}$, apply B&B to find a minimum-weight complete matching in the following weighted bipartite graph $G: A = \{1,2,3\}, B = \{4,5,6,7\}$,
$E = \{[(1,4),3], [(1,5),4], [(1,7),15], [(2,4),1], [(2,5),8], [(2,6),3], [(3,4),3], [(3,5),9], [(3,6),5]\}$.
Show the solution tree, the $\hat{C}$ of every tree node generated, and the optimal solution. Also, mark the order in which each node in the solution tree is visited.

Problem 4: (25 points)
Let A[1:n] be an arbitrary array of real numbers, such that no two numbers are equal. A *ramp* in the array A is any sequence A[i:j] that is sorted in increasing order. Give a divide-and-conquer algorithm that finds in the array A the longest ramp, and analyze the time complexity of your algorithm.