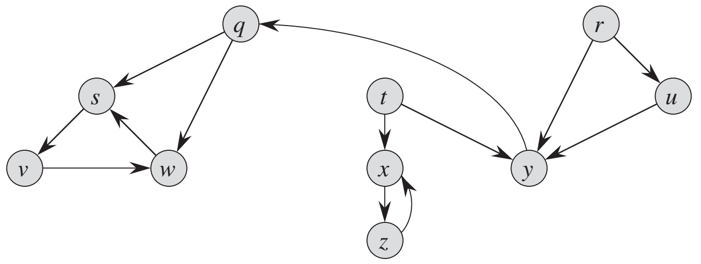
Problem Set 9

Data Structures and Algorithms, Fall 2021

Due: December 2, in class.

Problem 1

Consider the following graph:



(a) Show how BFS works on the above graph. Specifically, assume that the BFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. Show the $distance$ value and the $parent$ value for each vertex.

(b) Show how DFS works on the above graph. Specifically, assume that the DFS procedure considers the vertices in alphabetical order, and assume that each adjacency list is ordered alphabetically. Show the discovery and finishing times for each vertex, and show the classification of each edge.

Problem 2

(a) Give an example of a directed graph $G = (V, E)$, a source vertex $s \in V$, and a set of tree edges $E_{tree} \subseteq E$ such that for each vertex $v \in V$, the unique simple path in the graph (V, E_{tree}) from s to v is a shortest path in G, yet the set of edges E_{tree} cannot be produced by running BFS on G, no matter how the vertices are ordered in each adjacency list.

(b) Make a 3-by-3 chart with row and column labels `WHITE`, `GRAY`, and `BLACK`. In each cell (i, j), indicate whether, at any point during a depth-first search of a directed graph, there can be an edge from a vertex of color i to a vertex of color j. For each possible edge, indicate what edge types it can be.

(c) Give a counterexample to the conjecture that if a directed graph G contains a path from u to v, then any depth-first search must result in $v.d \leq u.f$.

Problem 3

Show that we can use a depth-first search of an undirected graph $G = (V, E)$ to identify the connected components of G, and that the depth-first forest contains as many trees as G has connected components. More precisely, show how to modify depth-first search so that it assigns to each vertex v an integer label $v.cc$ between 1 and k, where k is the number of connected components of G, such that $u.cc = v.cc$ if and only if u and v are in the same connected component. You need to give the pseudocode of your algorithm, and your algorithm's runtime should be $O(|V| + |E|)$.

Problem 4

An undirected graph $G = (V, E)$ is bipartite if the vertices V can be partitioned into two subsets L and R, such that every edge has one vertex in L and the other in R.

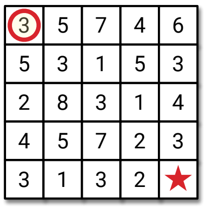
(a) Prove that every tree is a bipartite graph.

(b) Prove that an undirected graph G is bipartite iff every cycle in G has an even number of edges.

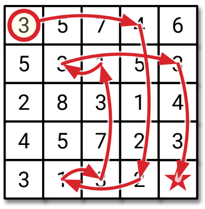
(c) Describe an algorithm that determines whether a given undirected graph G is bipartite. You need to give the pseudocode of your algorithm, and your algorithm's runtime should be $O(|V| + |E|)$.

Problem 5

A number maze is an $n \times n$ grid of positive integers. A token starts in the upper left corner; your goal is to move the token to the lower-right corner. On each turn, you are allowed to move the token up, down, left, or right; the distance you may move the token is determined by the number on its current square. For example, if the token is on a square labeled 3, then you may move the token three steps up, three steps down, three steps left, or three steps right. However, you are never allowed to move the token off the edge of the board. Devise an algorithm that either returns the minimum number of moves required to solve a given number maze, or correctly reports that the maze has no solution. For example, given the number maze in the following figure, your algorithm should return the integer 8. You need to give the pseudocode of your algorithm and analyze its runtime.



3	5	7	4	6
5	3	1	5	3
2	8	3	1	4
4	5	7	2	3
3	1	3	2	★



Problem 6

Consider a directed graph $G = (V, E)$, where each edge is colored either red, white, or blue. A walk in G is called a *French flag walk* if its sequence of edge colors is red, white, blue, red, white, blue, and so on. More formally, a walk $v_0 \to v_1 \to v_2 \to \cdots \to v_k$ is a French flag walk if, for every integer i, the edge $v_i \to v_{i+1}$ is red if $i \mod 3 = 0$, white if $i \mod 3 = 1$, and blue if $i \mod 3 = 2$. Given G and a vertex $v \in V$, devise an algorithm to find all vertices in G that can be reached from v through a French flag walk. You need to give the pseudocode of your algorithm and analyze its runtime.