PRACTICE PROBLEMS: GRAPH THEORY

"If you're walking down the right path and you're willing to keep walking, eventually you'll make progress." — Barack Obama

Course: CS 5002
Fall 2018
Due: No due date

OBJECTIVES

After you complete this assignment, you will be comfortable with:

- Graph representations
- Graph traversals
- Djikstra's Algorithm

RELEVANT READING

- Rosen, chapter 7: Graphs
  - Chapter 10.1. Graphs and Graph Models
  - Chapter 10.2 Graph Terminology and Special Types of Graphs
  - Chapter 10.3 Representing Graphs and Graph Isomorphis,
  - Chapter 10.6 Shortest-Path Problems

Problem 1: General graphs (10 points)

(a) (4 points) Given an adjacency-list representation of a directed graph, how long does it take to compute the out-degree of every vertex? How long does it take to compute the in-degrees? Give a short description or explanation for each.

Computing out-degree of every vertex takes $\Theta(V+E)$, whereas computing the in-degree takes $\Theta(V \cdot E)$.

(b) (6 points) Give an adjacency-list representation for a complete binary tree on 7 vertices. Then, give an equivalent adjacency-matrix representation. Assume that vertices are numbered from 1 to 7, starting at the top and numbering the nodes left to right at each level.

The adjacency list for a complete binary tree is given as follows:

Vertex 1 (root)	:	$[2,3]$
Vertex 2	:	$[1,4,5]$
Vertex 3	:	$[1,6,7]$
Vertex 4	:	$[2]$
Vertex 5	:	$[2]$
Vertex 6	:	$[3]$
Vertex 4	:	$[3]$

The adjacency matrix representation of the same binary tree is given as follows:

$$A = \begin{bmatrix} 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

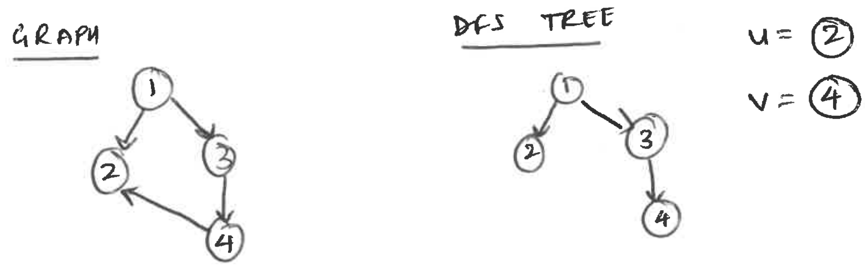


Figure 1: Graphical representation of a counterexample for Problem 2.

Problem 2: DFS (5 points)

Assume a graph G that contains a path from u to v, and $u.depth < v.depth$ (that is, the depth of u is less than the depth of v) in a depth-first search of G. I propose that v is a descendant of u in the traversal produced using depth-first algorithm. Provide a counter-example.

One possible counterexample is depicted in Figure 1.

Problem 3: BFS (5 points)

What is the running time of BFS if the graph is represented by an adjacency matrix? Assume the traversal algorithm is modified as necessary to handle the matrix rather than the lists.

The running time of BFS using adjacency matrix is eqaul to $O(|V|^2)$.

Problem 4: White Hats, Black Hats (15 points)

In the world of politics, there are two kinds of politicians: "white hats" (good guys) and "black hats" (bad guys). Let's assume that we don't really care about which party a politician is a part of for now. But, between any pair of politicians, there may or may not debate. Suppose we have a list of n politicians, and a list of r pairs of politicians that have a debate. Give an algorithm that determines whether it is possible to specify some politicians as white hat, and the rest as black hat, such that each debate is between a white hat and a black hat.

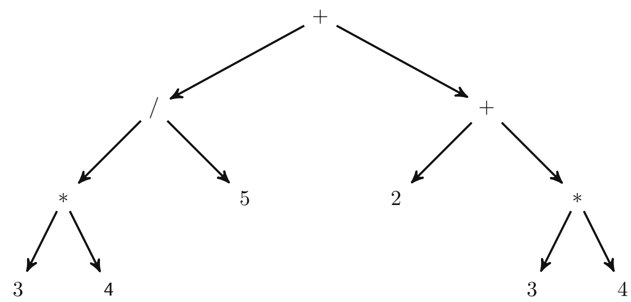
If it is possible to do this determination, your algorithm should produce it. The algorithm should run in $O(n + r)$ time.

The required algorithm is possible, and the possible solution is given in the listing below.

```
IsBipartite(n, r)
  for I: i in |G|
  color[i] = -1
color[r] = 1
push(Q, r)
while (! empty(Q))
  v = pop(Q)
  for u in neighborhood(n, v)
  if color[u] = -1
    color[u] = 1 - color[v]
    push(Q, u)
  else if color[u] = color[v]
    return false
  return true
```

Problem 5: Graph Arithmetic (10 points)

Consider representing an arithmetic expression as a tree. Each leaf is a number (integer), and each internal node is an arithmetical operations $(+, -, *, /)$. For example, the expression $3 * 4/5 + 2 + (3 * 4)$ is represented by the following tree:



Give an $O(n)$ algorithm for evaluating such an expression, where there are n nodes in the tree. In your solution, provide a 1-2 sentence summary of your algorithm, and then provide psuedocode for your algorithm.

The proposed pseudocode for this problem is given in the listing below.

```
eval(G, v)
  u = neighbors(G, v)
  if |u| = 0
    return v.val
  else
    l = eval(G, u[0])
    r = eval(G, u[1])
    if v.op == "+"
      return l + r
    if v.op == "-"
      return l - r
    if v.op == "*"
      return l + r
    if v.op == "/"
      return l - r
```

Problem 6: Communication Networks (5 points)

One usage case for a graph is to use it to model communication networks. In all electronic communication networks, there is some probability that a message between node a and node b will fail— that is, the message won't successfully make it to node b.

Give an efficient algorithm that produces the most reliable path between two given vertices.

You are given a directed graph $G = (V, E)$. Each edge $(u, v) \in E$ has an associated value, $p(u, v)$ such that $0 \leq p(u, v) \leq 1$ and represents the probability of a successful transmission between u and v. Assume that all of these probabilities are independent.

Note: If event A is independent of event B, and event A happens with $p(A)$ probability, and event B happens with $p(B)$ probability, the probability that both event A and B happen is $p(A) \cdot p(B)$.

Give a short summary of your algorithm approach, psuedocode for the algorithm, and an estimate of the run time.

Sketch of the solution: Convert all edge weights to their negative logarithm. This will yiels positive edge weight on which we can now perform Dijkstra's algorithm, and it will yield a path with the highest probability.

Question	Points	Score
General graphs	10	
DFS	5	
BFS	5	
White Hats, Black Hats	15	
Graph Arithmetic	10	
Communication Networks	5	
Total:	50	