ICS 311: Algorithms Spring 2025

Problem Set 10

Kyle Berney *Due: Friday, Mar 28, 2024 at 4pm*

You may discuss the problems with your classmates, however **you must write up the solutions on your own** and **list the names** of every person with whom you discussed each problem.

Start **every** problem on a separate sheet of paper, with the exception of Problems 1 (Peer credit assignment) – Problem 1 can be on the same page as any other problem. Any problem that starts on the same sheet of paper as some other problem will receive 0 points!

1 Peer Credit Assignment (1 point extra credit for replying)

Please list the names of the other members of your peer group for this week and the number of extra credit points you think they deserve for their participation in group work.

- You have a total of 60 points to allocate across all of your peers.
- You can distribute the points equally, give them all to one person, or do something in between.
- You need not allocate all the points available to you.
- ***You cannot allocate any points to yourself!*** Points allocated to yourself will not be recorded.

2 Bipartite Graph (30 pts)

A graph (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) **(15 pts)** Prove that every tree (not necessarily binary) is a bipartite graph. *Hint: demonstrate how to partition the vertices of an arbitrary tree into the two sets and prove that this partitioning satisfies the definition/conditions of a bipartite graph.*

(b) **(15 pts)** Design an efficient algorithm that determines whether a given undirected connected graph is bipartite. Write down the pseudocode, analyze its running time and prove that your algorithm is correct. ***No points will be given without the proof.***

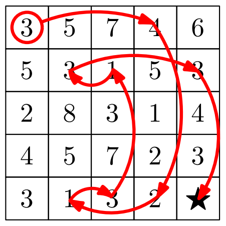
3 Number Maze (30 pts)

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.

In this problem, you will design and analyze an efficient 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. Describe the solution to this problem at a high level, justify why it works, write down the pseudocode for your algorithm and analyze its running time. ***All four parts must be present to receive any credit!***

Hint: It is highly advisable that you read Section 5.7 in Jeff Erickson's online textbook before attempting to solve this problem.

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



A 5×5 maze that can be solved in eight moves

4 Race Against the Lava (40 pts)

A geologist is conducting field research near Kilauea, where they have set up important data collection equipment at various research sites. Unexpectedly, a volcanic eruption occurs, causing lava to spread through the area. The geologist must safely escape while collecting as much research equipment as possible before it is destroyed by the lava.

The terrain is modeled as a directed acyclic graph $G = (V, E)$, where each node represents a location and each edge represents a downhill trail that allows movement from one location to another. Due to the steep terrain, the geologist can only travel along the direction of the edges, as climbing uphill is too time consuming and physically demanding in this time of stress. Likewise, lava flows downhill, following the same edge directions.

The geologist starts at a designated location s and must reach a safe destination t. Each location $v \in V$ has an associated research equipment value (non-negative) specified by an array val, where $val[v]$ denotes the value of the equipment at that location ($val[v] = 0$ if the location does not contain any equipment, in particular, the starting and destination locations will never contain any equipment).

Initially, a single location $\ell \in V$ contains lava (the starting and destination locations are guaranteed to be safe initially, so that $s, t \neq \ell$). The lava spreads at the same speed as the geologist: at each time step, lava flows from all currently affected locations to all directly connected locations following the directed edges. Similarly, in each time step, the geologist can move along a downhill trail to a neighboring location and automatically collect any research equipment present there.

Once a location is engulfed by lava, it remains hazardous indefinitely, preventing the geologist from traversing through it. The objective is to determine the maximum possible value of equipment that can be collected while ensuring the geologist successfully escapes to the safe destination without harm.

The graph G is provided using an adjacency list representation, where for each node $v \in V$, $G.adj[v]$ gives a list of vertices specifiying which locations can be directly reached from v. As an example, given the directed acyclic graph shown below, where $s = v_1$, $t = v_{10}$, $\ell = v_4$, and $val = [0, 3, 5, 2, 6, 8, 4, 7, 2, 0]$ (the value of the equipment at location v_i is stored in $val[i]$). The maximum possible value of equipment that can be collected is 13, using the path $v_1 \to v_3 \to v_6 \to v_{10}$. In comparison, the path $v_1 \to v_2 \to v_5 \to v_3 \to v_{10}$ is invalid, as the lava flow reaches location v_5 at the same time the geologist arrives. For clarity, the lava initially starts at location v_4 and in each subsequent time step it spreads as follows: first to v_7, then to v_5 and v_8, followed by v_3, v_6, and v_9, and lastly to v_{10} (all other locations are unreachable from v_4).

$$\begin{tikzcd}
 & & & v_1 \arrow[dll] \arrow[dr] & & & \\
 & v_2 \arrow[rrr] \arrow[dl] \arrow[dr] & & & v_3 \arrow[d] \arrow[ddrr] & & \\
v_4 \arrow[dr] & & v_5 \arrow[urr] \arrow[rr] & & v_6 \arrow[d] \arrow[drr] & & \\
 & v_7 \arrow[ur] \arrow[rrr] & & & v_8 \arrow[r] & v_9 \arrow[r] & v_{10}
\end{tikzcd}$$

$s = v_1$
$t = v_{10}$
$\ell = v_4$

(a) **(10 pts)** Given that the lava flow starts at node ℓ at time step 0, determine for each vertex $v \in V - \{\ell\}$ the time step that it will be engulfed by lava. (If v is not reachable from ℓ, we say that it will be engulfed by lava at time step ∞.)

(b) **(15 pts)** Design a ***recursive backtracking*** (brute-force) algorithm that determines the maximum value of equipment that can be collected while ensuring the geologist escapes to the destination location without harm. *(Hint: How can the times computed in part (a) be used to determine which paths the geologist can safely take?)*

Write down the pseudocode of your algorithm and prove its correctness using a proof by induction. A solution with no proof will receive 0 points! ***Make your pseudocode as simple as possible. You will be graded on the elegance of your solution, not just the correctness.***

To help get you started, LavaRace(G, s, t, ℓ, val) should return the maximum value of equipment that the geologist can safely collect, starting from location s and ending at location t, with a lava flow initially starting at location ℓ. And, LavaRace-Aux$(G, u, t, val, time)$, should be a recursive backtracking function that returns the maximum value of equipment that the geologist can safely collect, starting from location u at time step $time$ and ending at location t.

(c) **(15 pts)** Modify your pseudocode in part (b) to use memoization. Write down the modified pseudocode and explain your choice of the memoization table and the indices used for memoization.

5 DFS and Cycles (OPTIONAL - 0 pts)

DFS classifies edges as tree edges, back edges, forward edges, and cross edges (see page 569 in the CLRS textbook).

(a) How can you modify the DFS pseudocode from the CLRS textbook to detect back-edges? Write down your modified pseudocode and explain why your modifications are sufficient.

(b) How can you modify the DFS pseudocode from the CLRS textbook to determine whether a graph has a cycle? Write down the pseudocode and explain why your code returns a correct answer. *Your procedure should return immediately as soon as it finds a cycle.*

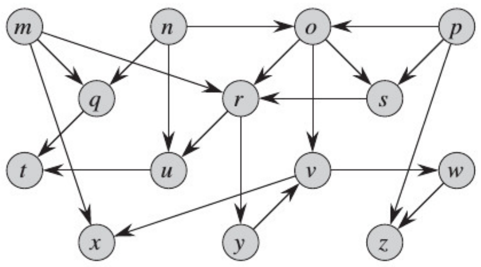
(c) What is the runnning time of your cycle-detecting algorithm on **directed** graphs? ***Justify your answer.***

(d) What is the running time of your cycle-detecting algorithm on **undirected** graphs? ***Justify your answer.***

6 Counting Simple Paths in a DAG (OPTIONAL - 0 pts)

In this problem you will design an algorithm that takes as input a directed acyclic graph $G = (V, E)$ and two vertices s and t, and returns the number of simple paths from s to t in G.

For example, the directed acyclic graph below contains exactly ***four*** simple paths from vertex p to vertex v: *pov*, *poryv*, *posryv*, and *psryv*. Notice: your algorithm needs only to count the simple paths, not list them.



(a) Design a ***recursive backtracking*** (brute-force) algorithm that determines the number of paths from s to t.

Write down the pseudocode of your algorithm and prove its correctness, i.e., convince us that it works beyond any doubt. *(Hint: using induction.)* A solution with no proof will receive 0 points! ***Make your pseudocode as simple as possible. You will be graded on the elegance of your solution, not just the correctness.***

(b) Modify your pseudocode in part (a) to use memoization. Write down the modified pseudocode and explain your choice of the memoization table and the indices used for memoization. *(Hint: What is a natural way to memoize solutions to subproblems that had already been computed?)*