University of Iowa | Homework 3 | CS 3330
Department of CS | Fall 2024

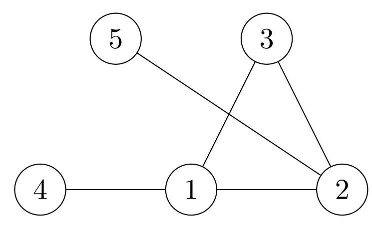
**Assigned:** Friday 11:59 PM, September 13, 2024
**Due:** Friday 11:59 PM, September 20, 2024
**Reading:** Kleinberg and Tardos, Chapter 3.1-3.2, Chapter 3.4-3.6, Slides of Week 3

1. [**Human Compiler**]:

   (a) [**5 points**] Execute the following pseudocode of the DFS algorithm and display the *Explored* array at the beginning and at the end of each call of the DFS function. We initiate the function with a call to DFS(1) for the first time, and suppose that in the **foreach** loop, edges are visited based on the numbers on the vertices, i.e., the first edge will be $\{1, 2\}$.

*Explored* $\leftarrow$ array of all zeros;
**Function** `DFS` *(starting vertex s)*
    *Explored*$[s] \leftarrow 1$;
    **foreach** $(s, v)$ *in* $E$ **do**
        **if** $v$ *is not explored* **then**
            Call `DFS`$(v)$;
        **end**
    **end**

| | Beginning of the call | End of the call |
|---|---|---|
| DFS(1): | [0,0,0,0,0] | [1,1,1,1,1] |
| | | |
| | | |
| | | |
| | | |
| | | |
| | | |




   (b) [**10 points**] Modify the pseudocode above so that it returns a cycle if one exists and *none* if the connected component of $s$ is acyclic.

2. [**Exploring a maze**]: You and your friends are visiting one of the famous corn mazes in Iowa. The maze is square in shape, and you must start at one end and reach the other. You are also given a map of the maze. Somewhere in the middle, you and your friends get lost. You realize that the CS 3330 lecture on graphs can bail you out here.

   You start by constructing a graph with $n^2$ nodes in a $n \times n$ grid. The edges represent obstacle-free pathways for movement. Here, an edge only connects neighboring nodes that are side by side or up and down (i.e., there are no diagonal edges). You label the nodes intersecting the maze wall using the letter 'w'. As these nodes represent walls, you cannot move across them.

   (a) [5 points] Draw a graph as described above in a $6 \times 6$ grid. Mark the top-left most node as the start node $s$ and the bottom-left most node as the exit node $e$. Mark some nodes as 'w' while ensuring a path from $s$ to $e$. Highlight one such path.

(b) [15 points] Describe an algorithm that finds the shortest path from $s$ to $e$ in a graph drawn over a $n \times n$ grid. Write down the pseudocode.

(c) [8 points] Now enter a second maze with a single entry point but multiple exits. Emboldened by the Algorithm you just designed, your group decided to have some fun by taking the exit that is furthest away. How will you use the algorithm you developed in part (b) to find the shortest path to the furthest exit? Describe any modifications that are necessary (if any).

3. **[15 points] [Surviving COVID-2035]**: A time-traveler from 2040 has arrived at the current time to warn us about a deadly variant of COVID virus that spreads in 2035. The virus is so infectious that it spreads to anyone who comes in contact with an infectious individual and the transmission happens instantaneously, i.e., whoever comes in contact with an infectious person also becomes infectious immediately.

   You are terrified at the prospect of losing yourself and your loved ones to the virus. You realize you can maximize the chances of saving yourself and your loved ones by moving to an isolated community that can sustain itself. To do so, you first collect the data from multiple sources to construct a graph of the US population. Here, a node represents a person, and an edge represents the interaction between two individuals. You realize that an isolated community of less than 1500 people cannot sustain itself. Hence, you want to move to the smallest possible community with more than or equal to 1500 residents. Design an algorithm that visits each node in the population graph exactly once to find the smallest isolated community with more than or equal to 1500 residents. Provide pseudocode. Describe what each line of your algorithm does. Argue why your algorithm is correct (remember, your life depends on it).

4. **[Graph representations]**Consider the problem of computing in and out degrees for each node in a directed graph $G$ with $n$ nodes and $m$ edges. Note that in-degree is the number of incoming edges a node has, and out-degree is the number of outgoing edges a node has.

   (a) [8 points] Write down the pseudocode for the most efficient algorithm that computes in-degrees of all the nodes given an adjacency list representation of the directed graph. What is your algorithm's worst-case running time?

   (b) [8 points] Write down the pseudocode for the most efficient algorithm that computes out-degrees of all the nodes given an adjacency list representation of the directed graph. What is your algorithm's worst-case running time?

   (c) [8 points] Write down the pseudocode for the most efficient algorithm that computes in-degrees of all the nodes given an adjacency matrix representation of the directed graph. What is your algorithm's worst-case running time?

   (d) [8 points] Write down the pseudocode for the most efficient algorithm that computes out-degrees of all the nodes given an adjacency matrix representation of the directed graph. What is your algorithm's worst-case running time?

5. **[10 points] [Topological Sorting]** Given a DAG $G(V, E)$, give an algorithm to generate all possible topological sorting of $G$. Write down the pseudocode.