CS 101 Fall 2022 - Quiz 6-B

15/11/2022 - 20 Minutes **Name:** **ID number:**

---

1. (8 points) True or False

For each statement, choose T if the statement is correct, otherwise, choose F.

Note: You should write down your answers in the box below.

(a)	(b)	(c)	(d)	(e)	(f)	(g)	(h)
F	F	T	F	T	T	F	F

(a) (1') The time complexity of graph traversal on a connected graph is $\Theta(V)$ where V is the number of vertices.

(b) (1') A directed graph with n vertices has $n-1$ edges may be strongly connected.

(c) (1') In a simple undirected graph with n vertices and 3 connected components, the maximum number of edges in the graph is $\frac{(n-3)(n-2)}{2}$

(d) (1') It's convenient to use a stack to implement BFS.

(e) (1') Union-Find/Disjoint-Set-Union cannot be used to determine reachability(whether a directed path from a given vertex to another vertex exists) on a directed graph.

(f) (1') Union-Find/Disjoint-Set-Union can be used to solve connected component size problem (couting vertices in a connected component) on a undirected graph.

(g) (1') A union tree generated by a union-by-rank DSU of n nodes has $\Theta(\log n)$ height.

(h) (1') A union tree generated by a union-by-rank DSU of height h has $O(2^h)$ nodes.

The **Union-by-rank** strategy: When merging two sets A, B whose root is u and v respectively, if $\text{height}(u) < \text{height}(v)$, make u a child of v. That is: merge the tree with lower rank into the other one.

2. (5 points) Maze

Background(you can skip this:) After your friend gets lost in the dungeon for the 1234th time(in video game), he begs you to design a unified maze solver.

Goal: The maze is stored in a $(n \times m)$-sized matrix. Each cell holds the information of whether its position has an obstacle (of course you can't pass the cell if there's an obstacle). You start from $(1, 1)$ and you want to go to (m, n). Design an algorithm to find the shortest path from $1, 1$ to (m, n).

Briefly explain your algorithm with **natural language**.

(hint: you should try build a graph first).

Example:

The output of your graph should be shwon in the picture. The length of the path is 10.

Solution: Build the graph first: The input matrix can be seen as a undirected graph, each point (i, j) as a vertex. Connect all the adjacent cells without obstacle.

Use BFS to search this graph starting at point (1, 1). For each node add to the stack, record the node it comes from (or record the path). When we reach point (m,n), stop.

We can calculate the shortest path by the additional recorded information.