CS101 Algorithms and Data Structures
Fall 2023
Homework 8

Due date: 23:59, December 10th, 2023

1. Please write your solutions in English.
2. Submit your solutions to gradescope.com.
3. Set your FULL name to your Chinese name and your STUDENT ID correctly in Account Settings.
4. If you want to submit a handwritten version, scan it clearly. `CamScanner` is recommended.
5. When submitting, match your solutions to the problems correctly.
6. No late submission will be accepted.
7. Violations to any of the above may result in zero points.

1. (12 points) Multiple Choices

Each question has **one or more** correct answer(s). Select all the correct answer(s). For each question, you will get 0 points if you select one or more wrong answers, but you will get 1 point if you select a non-empty subset of the correct answers.

Write your answers in the following table.

(a)	(b)	(c)	(d)	(e)	(f)

(a) (2') Which of the following is(are) incorrect? ($|V|$ is the number of vertexes, $|E|$ is the number of edges.)

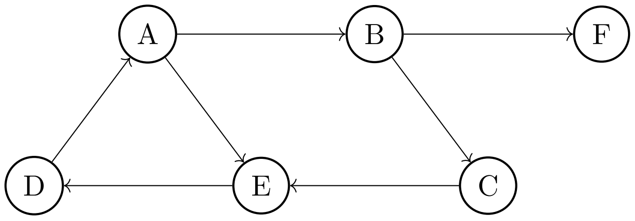
A. In DAGs(directed acyclic graphs), A simple path can visit the same vertex twice.

B. For a forest, the number of trees is $|V| - |E|$.

C. In weighted directed graphs, if both (v_j, v_k) and (v_k, v_j) are edges, it is required that they have the same weight.

D. The maximum number of edges in a simple undirected graph is $O(|V|^2)$.

(b) (2') If we use the breadth-first algorithm to traverse the following graph, which are the possible orders of visiting the nodes?



A. BAEDFC

B. ABECFD

C. DABECF

D. ABCEDF

(c) (2') Which of the following statements are true for graph traversal? ($|V|$ is the number of vertexes, $|E|$ is the number of edges.)

A. Time complexity of DFS and BFS are both $\Theta(|V| + |E|)$.

B. For a directed graph, DFS starting at any vertex can traverse all the nodes.

C. Assuming we use a queue to implement BFS. Let $d(v)$ be the minimum number of edges between v and s, s is the start vertex. For any two vertices u, v in the queue at the same time, $|d(u) - d(v)| \leq 1$.

D. A directed graph with n nodes and $2n$ edges is strongly connected.

(d) (2') Which of the following statements are true for graph traversal?

A. Undirected graph $G = (V, E)$ is stored in an adjacency matrix A. The degree of V_i is $\sum_{j=1}^{|V|} A[i][j]$. $(A[i][j] = 0, 1)$

B. Graph with an odd number of vertices cannot be a bipartite graph.

C. If a graph with n vertices has $n - 1$ edges, it must be a tree.

D. Given two vertices s and t in a graph G, we can use both BFS and DFS to determine whether there exists a path from s to t.

(e) (2') Consider a tree with 64 nodes. It is generated by a disjoint set union with union-by-rank (height). Select the possible height(s) of the tree.

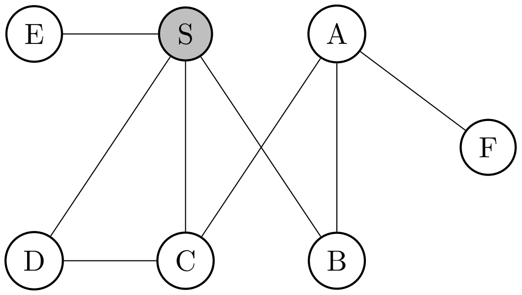
A. 1
B. 2
C. 6
D. 7

(f) (2') Which of the following statements are true for MST(Minimum Spanning Tree)?

A. If the weights of the graph are distinct, i.e. every edge has a different weight, the MST of the graph is unique.
B. If we use heap or priority queue to optimize Prim's algorithm when choosing the next edge, it will always have a better time complexity than the unoptimized algorithm on any graph.
C. Prim's algorithm is a divide-and-conquer algorithm because it divides the graph into S and $V - S$ then solve.
D. If we add a new edge$e = (u, v)$ into a graph $G = (V, E)$ with unique MST to get a new graph $G' = (V, E \cup \{e\})$. There is at most 1 edge difference between the MST of G and G'.

2. (9 points) Graph traversal

Consider the following undirected graph.



(a) (3') Give the adjacency list for the graph. You should write the node in alphabetical order. (Leave it blank if the node has no neighbour).

$$\begin{aligned}
adj(S) &= [S \to B \to C \to D \to E],\\
adj(A) &= [A \to B \to C \to F],\\
adj(B) &= [B \to A \to S],\\
adj(C) &= [C \to A \to D \to S],\\
adj(D) &= [D \to C \to S],\\
adj(E) &= [E \to S],\\
adj(F) &= [F \to A],
\end{aligned}$$

(b) (3') Give the visited node order using the above adjacency list for Breadth First Search starting with S.(Each nodes only visit once)

Solution: S B C D E A F

(c) (3') Give the visited node order using the above adjacency list for Depth First Search starting with S. (Each nodes only visit once)
Notes: The adjacency list should be inserted into the stack in reverse order. For example, if you access a point with the adjacency list [A, S], then after pushing the adjacency list onto the stack, A will be on the top of the stack

Solution: S B A C D E F

3. (4 points) Disjoint Set practice

Let's performing a series of merge opertions on a disjoint set structure. Please show the final disjoint set tree for each of the following optimization strategies.

op 1. initialize: $\{1\}, \{2\}, \{3\}, \{4\}, \{5\}, \{6\}, \{7\}, \{8\}, \{9\}$

op 2. merge 1,4

op 3. merge 3,2

op 4. merge 7,6

op 5. merge 7,3

op 6. find 2

op 7. merge 5,8

op 8. merge 1,9

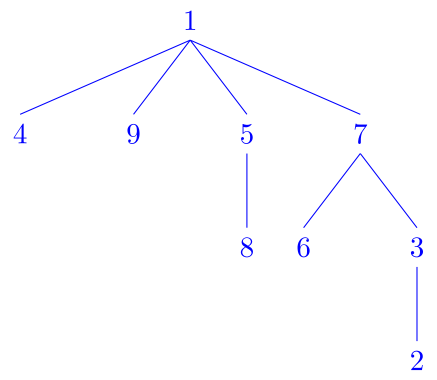
op 9. merge 9,8

op 10. merge 9,2

op 11. find 6

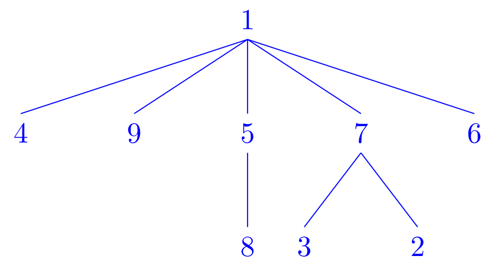
(a) (2') Only with union-by-height optimization. (When two trees have the same height, the set specified first in the union will be the root of the merged set.)

Solution:



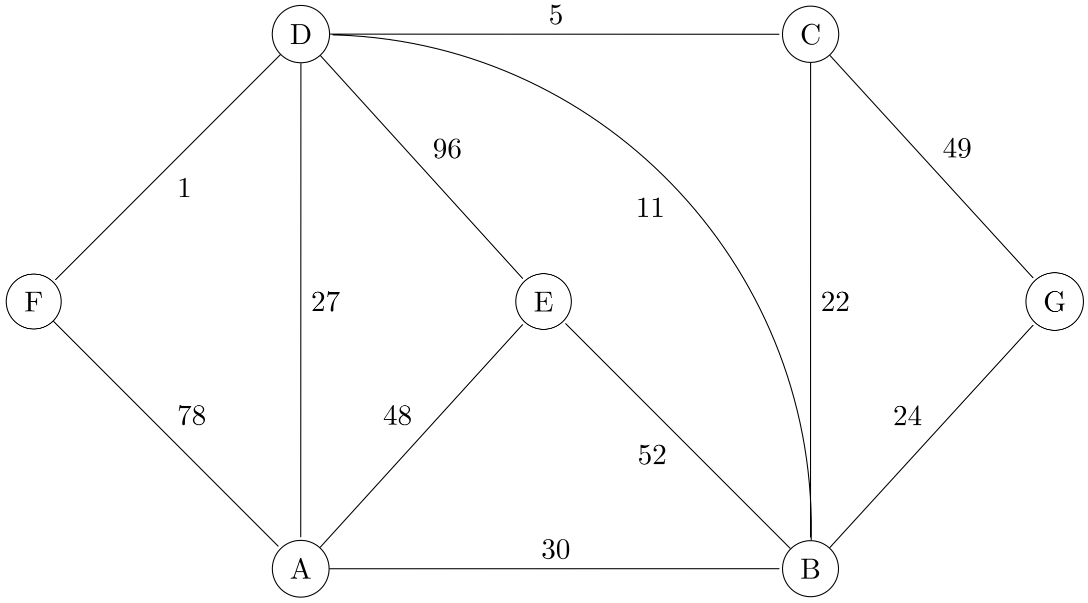
(b) (2') Only with path compression (The set specified first in the union will always be the root of the merged set).

Solution:



4. (6 points) In Or Not In?

In this problem, we want you to run Kruskal's algorithm on the given graph. For each edge, indicate either the order it was added to the Minimum Spanning Tree (MST) or if it's not in the MST. For instance, if edge (u, v) was the 3rd edge added to the MST, mark "In MST" and write 3 in the corresponding order box. If an edge is not in the MST, mark "Not In MST" and leave the order box blank.



You should write your answer in the table below:

Edge	Whether In MST	Order
AB	[✓ **Not In MST**] ○ In MST	
AD	○ Not In MST [✓ **In MST**]	5
AE	○ Not In MST [✓ **In MST**]	6
AF	[✓ **Not In MST**] ○ In MST	
BC	[✓ **Not In MST**] ○ In MST	
BD	○ Not In MST [✓ **In MST**]	3
BE	[✓ **Not In MST**] ○ In MST	
BG	○ Not In MST [✓ **In MST**]	4
CD	○ Not In MST [✓ **In MST**]	2
CG	[✓ **Not In MST**] ○ In MST	
DE	[✓ **Not In MST**] ○ In MST	
DF	○ Not In MST [✓ **In MST**]	1

5. (8 points) Traffic Network

In SC101 country, there are n cities and m **broken** roads, with each broken road connecting two different cities. You can consider this as a graph $G = (V, E)$.

Now the government wants to build a traffic net in the SC101 country. There are 2 crucial steps to take in constructing this traffic network:

1. Establish an airport in the i-th city with a cost of a_i
2. Repair the broken road $e_j = (u_j, v_j)$ to connect the city u_j and v_j, cost b_j.

In the final network, every city will either have an airport or be connected to a city with an airport. Your task is to design an algorithm to find the minimum cost to build the network.

Hint: You may find that the sum of the number of airports to be built and the number of roads to be repaired is n.

Solution:

6. (6 points) MST with special edge

Given a weighted undirected graph $G = (V, E \cup \{\mathbf{e}\})$. Other than normal graphs, one edge is defined as "special" in the graph. For each edge in E, it can be represented as a triple: $e_i = (u_i, v_i, w_i)$. u_i and v_i mean the indices of the vertices connected by the edge, and w_i means the weight of the edge. The special edge is $\mathbf{e}$. We want you to find the minimal spanning tree containing the special edge $\mathbf{e}$. Please fill in the pseudocode below to solve the question.

(You may use disjoint sets in the problem. You can use "make-set", "union", "find-set" to refer to the functions of those in the disjoint set.)

Hint: You can use Kruskal's algorithm to solve this problem, thinking about how to add the constraint that the spanning tree must contain $\mathbf{e}$.

Algorithm 1 Minimum Spanning Tree with Special Edge

```
Require: $V$ vertex set, $E$ edge set, $\mathbf{e}$ special edge
Ensure: Minimum Spanning Tree containing $\mathbf{e}$
 1: $A \leftarrow \varnothing$
 2: for each vertex $v$ in set $V$ do
 3:     Make-Set($v$)
 4: end for
 5: $(\mathbf{u}, \mathbf{v}, \mathbf{w}) \leftarrow \mathbf{e}$
 6: ________________
 7: Add $\mathbf{e}$ to set $A$
 8: Sort(E) {Sort the edge in ascending weight}
 9: for each edge $e$ in set $E$ do
10:     $(u, v, w) \leftarrow e$
11:     if ________________ then
12:         Add $(u, v, w)$ to set $A$
13:         ________________
14:     end if
15: end for
16: return $A$
```