United International University (UIU)

Dept. of Computer Science & Engineering (CSE)

Final Exam Total Marks: **40** Spring 2023

Course Code: CSE 2217 Course Title: Data Structure and Algorithms II

Time: 2 hours

Any examinee found adopting unfair means will be expelled from the trimester / program as per UIU disciplinary rules.

There are **six** questions. **Answer all of them**. Show full simulation/tabulations wherever necessary. Figures in the right-hand margin indicate full marks.

1. **(a)** Find the Minimum Spanning Tree (MST) for the given graph in Figure 1 using **Kruskal's algorithm**. Show the details of your calculation. **[2]**

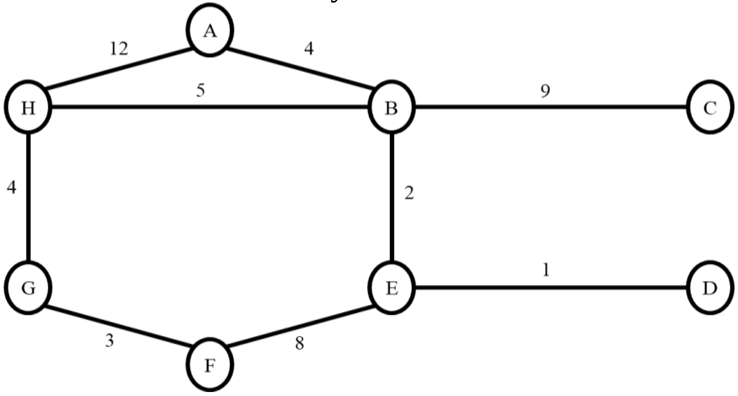


Figure 1

(b) Given a weighted graph where multiple edges have the same weight, you are asked to apply **Prim's algorithm** to find MST. “*Depending on the choice of edges (from edges with same weight), you may have different MSTs*”. Is the statement **True** or **False**? **Justify** your answer with an example. **[2]**

(c) Suppose you are asked to utilize a disjoint set data structure to implement **Kruskal's algorithm** to find MST from a given graph. Explain, with an example, how disjoint set data structure would help you to perform the following operations: **[3]**

i) joining two trees in the forest
ii) detecting a cycle

2. Consider the mentioned notations:

Text, $t = $ “$BATMANCATWOMANSUPERMAN$”
Pattern, $p = $ “MAN”
Modulo, $q = 11$
Hash of a string XYZ, $h(XYZ) = (X + Y + Z)\ mod\ q$;

where X, Y, and Z are the alphabetical sequence of keys [*i.e.: A =1, B=2, C=3, … Y=25, Z=26*]

Now answer the following questions using **Rabin-Karp** String matching algorithm:

(a) What is “spurious hit”? Explain with an example. **[1]**

(b) Show the indices of *valid hit* and *spurious hit* (if any) using the aforementioned hash function. **[3]**

(c) What is the time complexity of this algorithm? **Explain** how it improves the naïve string-matching algorithm? **[2]**

3. **(a)** Consider the graph in Figure 2 and answer the following questions:

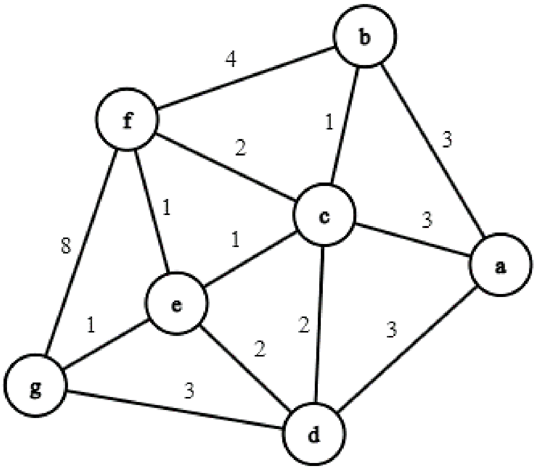


Figure 2

i) Find out the Shortest Path to every Node considering ‘a’ as Source Node using Dijkstra Algorithm. Show all the steps/ calculations. **[3]**

ii) Can we find the shortest path from a Source Node to Destination Node using the MST of the graph? Explain your answer. **[1]**

(b) Find the Shortest path tree for the given graph in Figure 3 using the ***Bellman-Ford*** algorithm, where the source is vertex ‘b’. Show the details of your calculation. **[3]**

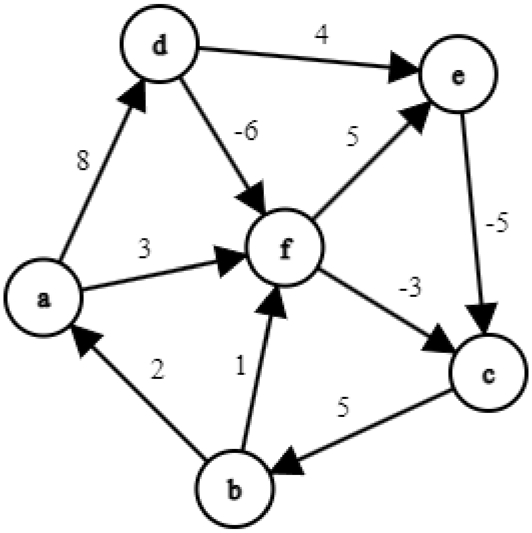


Figure 3

4. **(a)** Show the status of Figure 4 after **each** of the following operations:

i) Make_set(10) **[1]**

ii) Find_set(0) **[1]**

iii) Find_set(9) **[1]**

iv) Union(4,5). **[2]**

Note that the operations are executed **sequentially**. **You must use the union-by-rank and path-compression heuristic.**

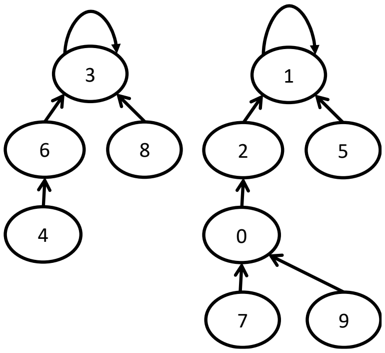


Figure 4

(b) In the Disjoint-Set Forests data structure, why do we use path-compression heuristic? Explain with an example. **[2]**

5. **(a)** Consider an open addressing hash table as shown below with the following items added beforehand.

Index	0	1	2	3	4	5	6	7	8	9
Value				26				14	44	

This function uses the following hash function to manage collisions:

$$h(k,\, i) \;=\; \big(h_1(k) \;+\; i\, h_2(k)\big)\, mod\; 10$$

Where, $h_1(k) \;=\; (3k \;+\; 5)\; mod\; 10$ and $h_2(k) \;=\; (2k \;+\; 3)\; mod\; 10$

Now, show calculations for the following operations and redraw the given hash table after the insertions.

i) Insert 72 **[2]**

ii) Insert 64 **[2]**

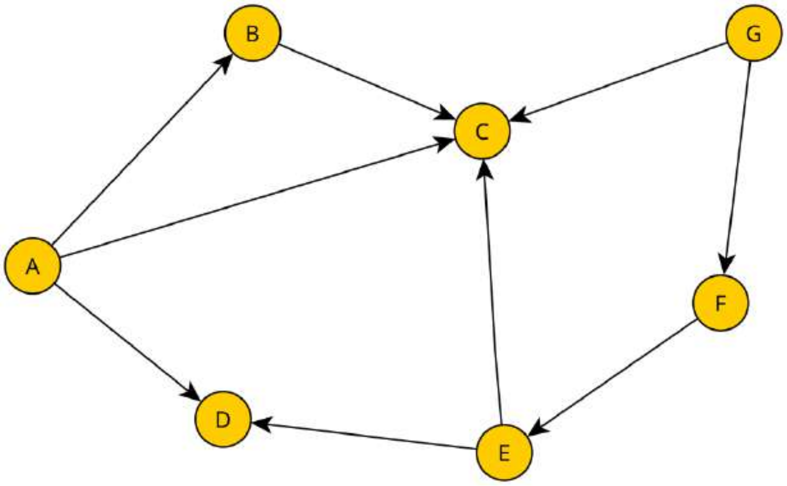
iii) Search 44 **[2]**

(b) If you try to **insert the value 16** in the hash table calculated from 4(a), you wouldn't be able to. Explain why this is the case and how you would solve this problem. **[1]**

Note: You are not allowed to switch from double hashing to any other technique.

6. **(a)** While running DFS, what is the maximum possible discovery and finishing time for a graph with 7 nodes? **[1]**

(b) What is a DAG in Graph Theory? How can you determine if a graph is a DAG or not? **[2]**

	(c) Sort the vertices of the Directed Acyclic Graph (DAG) of Figure 5 using topological sorting. You may start from any node.  Figure 5	**[3]**