BITS Pilani, K. K. Birla Goa Campus

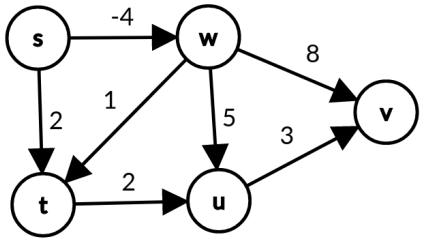
Data Structures and Algorithms (CS F211)

Comprehensive Exam (09/05/2023)
Total Marks: 40
Time Limit: 3 hours

This question paper contains **5** questions carrying **40** marks. All questions are compulsory. Answer each question on a fresh page. Answer all the parts of a question in the same place. Show the necessary steps and justifications for your answers.

Question 1 (8 marks)

```
BELLMAN-FORD(G, w, s)
1  INITIALIZE-SINGLE-SOURCE(G, s)
2  for i = 1 to |G.V| − 1
3      for each edge (u, v) ∈ G.E
4          RELAX(u, v, w)
5  for each edge (u, v) ∈ G.E
6      if v.d > u.d + w(u, v)
7          return FALSE
8  return TRUE
```



(a) (2 marks) Let $G = (V, E)$ be a weighted, directed graph with source vertex s and weight function $w : E \to \mathbb{R}$. Assume that G contains no negative-weight cycles that are reachable from s. Prove or disprove (by giving a counterexample) that the shortest path from a source vertex s to any vertex v will contain at most $|V| - 1$ edges.

(b) (2 marks) For the directed graph shown above, assume that in every iteration Bellman-Ford algorithm will relax the edges in the following order : (t, u), (s, t), (u, v), (w, v), (w, u), (w, t), (s, w). Let vertex s be the source vertex.

(i) Show the d and π values of vertex u after each iteration.

(ii) Show the d and π values of vertex v after each iteration.

(c) (2 marks) Explain lines 5-7 in the BELLMAN-FORD algorithm? What does it do? Why does it work?

(d) (2 marks) Modify the Bellman-Ford algorithm shown above so that it sets $v.d$ to $-\infty$ for all vertices v for which there is a negative-weight cycle on some path from the source vertex s to v.

Question 2 (8 marks)

```
DFS(G)
1  for each vertex u ∈ G.V
2      u.color = WHITE
3      u.π = NIL
4  time = 0
5  for each vertex u ∈ G.V
6      if u.color == WHITE
7          DFS-VISIT(G, u)
```

```
DFS-VISIT(G, u)
1   time = time + 1
2   u.d = time
3   u.color = GRAY
4   for each v ∈ G.Adj[u]
5       if v.color == WHITE
6           v.π = u
7           DFS-VISIT(G, v)
8   u.color = BLACK
9   time = time + 1
10  u.f = time
```

(a) (2 marks) Consider the depth-first search algorithm shown above. Draw the directed graph shown in Question 1 in your answer sheet, and write the discovery and finish times (d/f) for each vertex during the depth-first search. Assume that the **for** loop of lines 5-7 of the DFS procedure considers the vertices in the *reverse* alphabetical order. Also, assume that each adjacency list has vertices in the *reverse* alphabetical order.

(b) (2 marks) Modify the pseudocode for depth-first search shown above so that it prints out every edge in a directed graph G, together with its edge type. The edge type can be tree edge, back edge, forward edge or cross edge.

(c) (2 marks) Perform time complexity analysis of the modified algorithm found in part (b). Provide clear justifications for your answer. (Marks will be given only if the algorithm found in part (b) can correctly identify all the four possible edge types.)

(d) (2 marks) Can depth-first search be used to identify the connected components of an *undirected* graph G? If no, justify your answer. If yes, modify the depth-first search algorithm shown above so that it assigns to each vertex v an integer label $v.cc$ between 1 and k, where k is the number of connected components of G, such that $u.cc = v.cc$ if and only if u and v are in the same connected component. Assume that we have an attribute $v.cc$ associated with each vertex v.

Question 3 (8 marks)

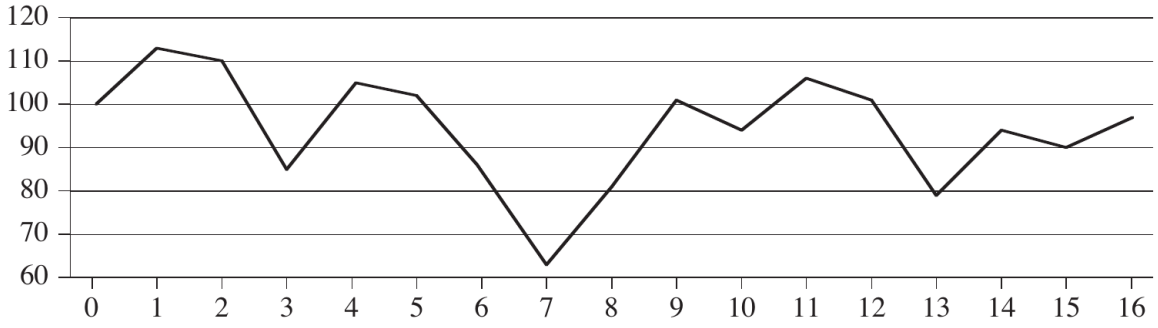
(a) (2 marks) Mention two important similarities and two important differences between a Binary Search Tree and a Red Black Tree.

(b) (2 marks) Suppose a Red Black tree forms a *complete binary tree* (i.e. every level of the binary tree is completely filled, except possibly the last level). The red black tree contains the following 15 keys : 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90 and 95. It is given that the nodes containing keys 40 and 45 are *red* in colour. Assume that all the Red Black tree properties are satisfied. Draw the red black tree. Mention the key inside each node. Mark either R or B next to each node to indicate the color of the node.

(c) (2 marks) Suppose we perform three delete operations on the Red Black tree found in part (b). The delete operations are performed in the following order : delete 25, delete 35, delete 30. Draw the red black tree obtained after the three delete operations are completed. Indicate the color of each node.

(d) (2 marks) Next we perform two insert operations to the Red Black tree found in part (c). The insert operations are performed in the following order : insert 35, insert 30. Draw the red black tree obtained after the two insert operations. Indicate the color of each node.

Question 4 (8 marks)



Day	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Price	100	113	110	85	105	102	86	63	81	101	94	106	101	79	94	90	97

Figure Information about the price of stock in the Volatile Chemical Corporation after the close of trading over a period of 17 days. The horizontal axis of the chart indicates the day, and the vertical axis shows the price.

(a) (4 marks) The figure above shows stock price of a company on different days. The problem is to find the maximum profit that can be obtained by buying the stock on day i and selling it on day j, where $j > i$. In the figure shown above, profit will be maximized when the stock is bought on day 7 and sold on day 11.

Let $P[0..n]$ be the input array that contains the price of the stock from day 0 to day n. You need to find the maximum value of $P[j] - P[i]$, where $j > i$. Devise a divide-and-conquer algorithm that can solve the maximum-profit problem in $\Theta(n)$ time.
(Hint : There is no need to convert the given maximum-profit problem into a maximum-subarray problem. A more efficient divide-and-conquer algorithm can directly work with array P.)

(b) (1 mark) Find the recurrence for the divide-and-conquer algorithm found in part (a). Solve the recurrence. (Marks will be awarded only if the algorithm in part (a) is correct.)

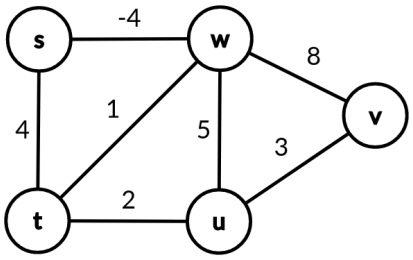
(c) (1 mark) When do *collisions* occur in a hash table? How does *chaining* resolve collisions?

(d) (2 marks) In open address hashing, the problem of *primary clustering* can occur. When does *primary clustering* occur? Which method (linear probing or double hashing) reduces the problem of primary clustering? Why?

Question 5 (8 marks)

GENERIC-MST(G, w)

1 $A = \emptyset$
2 **while** A does not form a spanning tree
3 find an edge (u, v) that is safe for A
4 $A = A \cup \{(u, v)\}$
5 **return** A



Assume that $G = (V, E)$ is a connected, undirected graph with a weight function $w : E \to \mathbb{R}$ for *all* the questions given below. Also, note that (a, b) and (b, a) refer to the same edge.

(a) (2 marks) Prove the following statement if it is true or provide a counterexample :

> The edge (a, b) having the third smallest weight will always be part of a minimum spanning tree if all the edge weights are *distinct* and $|V| > 3$.

(Hint : You can use ideas from Kruskal's algorithm to prove the above statement or to find a counterexample.)

(b) (1 mark) What loop invariant is maintained by the GENERIC-MST algorithm shown above? What is the meaning of a *safe* edge?

(c) (1 mark) Let $A = \{(s, w)\}$. List all possible safe edge(s) that can be added to set A in line 4 of the GENERIC-MST algorithm. (No partial marking)

(d) (2 marks) Prim's algorithm adds safe edges to set A in a certain sequence starting from an edge that is incident on the source vertex. Let $A = \emptyset$. Consider the four different sequences of edge additions to set A shown below:

J. $(u, v), (t, u), (t, w), (s, w)$
K. $(t, u), (u, v), (t, w), (s, w)$
L. $(t, w), (t, u), (s, w), (u, v)$
M. $(t, w), (s, w), (t, u), (u, v)$

Which of the above sequences of edge additions are possible for Prim's algorithm (assuming that the source vertex is chosen suitably)? Justify your answer. (No partial marking)

(e) (2 marks) Once again consider the four sequences of edge additions (J, K, L and M) shown in part (d). Let $A = \emptyset$. Which of the sequences of edge additions are possible for Kruskal's algorithm? Justify your answer. (No partial marking)