Data Structure Midterm 2

11/28/2023

Important Instructions

To respond to the test questions, you have the flexibility to employ algorithms that were not specifically addressed in our lectures. If you explicitly mention the algorithm's name, additional explanations are not necessary. However, if the algorithm is used incorrectly, it will be considered an incorrect response. If the algorithm's name is not explicitly stated, please provide an explanation of its functioning.

In addition, you are permitted to utilize data structures that were not covered in our lectures to address the problems. Nevertheless, it is essential to use them accurately.

True or False

> For each question, provide a straightforward answer of T (True) or F (False). No explanations are necessary. (2 points for each question)

T 1. Any full binary tree is a complete binary tree.

T 2. Any heap is a complete binary tree.

T 3. Assuming **T** is a binary tree. If **T** is a full binary tree, using an array representation is more space efficient; if **T** is a skewed binary tree, using a linked list representation is more efficient.

F 4. Assuming a binary tree **T** is represented using the linked-list representation. Then the inorder traversal of **T** gives the same result as BFS traversal of **T**.

F 5. Any DAG (Directed Acyclic Graph) **G** is also a tree.

F 6. The worst-case time complexity of inserting an element into a min heap is O(N), where N is the number of elements already included in the heap.

T 7. The worst-case time complexity of inserting an element into a BST (Binary Search Tree) is O(N), where N is the number of elements already included in the BST.

F 8. In a BST, all elements in the left sub-tree are smaller than the parent node, and all elements in the right sub-tree are larger than the parent node. Therefore, the min and max elements in a non-empty BST are always leaf nodes.

T 9. Assuming **U** is a union represented by a tree without the Collapsing rule or Weighting rule. The worst-case time complexity of finding a single element in **U** is O(N), where N is the number of elements already included in the BST.

F 10. Assuming **M** is a matrix representation of a directed graph **G**. If **M**[i][j] = **M**[j][i] for all (i, j), then SCC(**G**) = **G**.

Basic Questions

> For each question, please provide your answer along with a brief and concise explanation. Answers submitted without explanations will not receive any points. You can earn a maximum of 6 points for each question.

11. Given the infix expression (((A-B)*C)/(D+E)), please draw the corresponding expression tree. Afterward, utilize the expression tree to determine its prefix and postfix expressions. You are

required to demonstrate each step of your work, explicitly explaining how operators and operands are positioned within the tree. Additionally, illustrate the rules for traversing the tree when finding the prefix or postfix expression.

12. Let's assume that we want to create a Binary Search Tree (BST), denoted as **T**, by inserting a sequence of numbers sequentially into an initially empty tree. The given sequence is [1, 13, 2, 10, 5, 7, 8, 9, 4, 15]. Please demonstrate the tree's state after each insertion and provide the final representation of **T**.

13. Given the following 5 disjoint subsets:

    - $S_1 = \{1, 2, 3\}$
    - $S_2 = \{4, 5\}$
    - $S_3 = \{6\}$
    - $S_4 = \{7, 8\}$
    - $S_5 = \{9\}$

    Please utilize the **Tree Representation** of disjoint sets and follow the **Weighting Rule** (not the **Collapsing Rule**) to perform the following operations sequentially, and display the result of each operation:

    1. S_6 = Union(S_1, S_2)
    2. S_7 = Union(S_3, S_6)
    3. Find(3)
    4. Find(7)
    5. S_8 = Union(S_7, S_7)
    6. Find(9)
    7. S_9 = Union(S_5, S_8)

14. Execute Prim's Algorithm on the provided undirected graph **G** in figure-1 and demonstrate your step-by-step process.

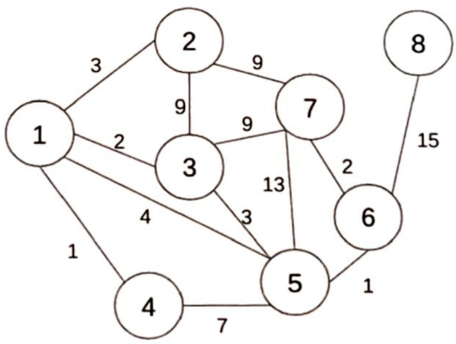

figure-1

15. Apply the Floyd-Warshall Algorithm to the provided directed graph **G** in figure-2. Please demonstrate your work step by step, including A^0, A^1, A^2, and A^3, and provide the final A^4.

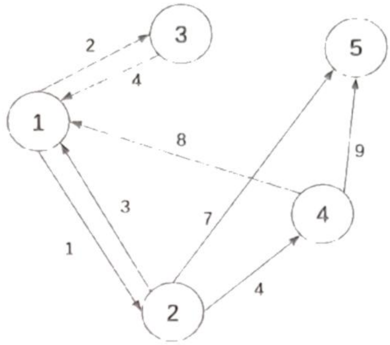

figure-2

$E \log V$

Application Questions

$E \log E$

> For each question, please provide your answer along with a brief yet clear explanation. Responses lacking explanations will receive 0 points. You can earn a maximum of 10 points for each question.

16. Consider the task of finding the Minimum Spanning Tree (MST) of an undirected graph **G**. In what scenarios does Prim's algorithm prove to be more time-efficient than Kruskal's algorithm? To earn full marks, please express the constraints formally.
17. Assume **T** is a binary tree. The preorder traversal of **T** is [1, 7, 2, 4, 3, 5], and the inorder traversal of **T** is [2, 7, 4, 1, 5, 3]. Please reconstruct **T** based on these traversal results. If multiple valid solutions exist, please provide any of them.
18. Assuming **G** is a directed graph, how can we efficiently determine if **G** contains a cycle? Please propose a method for checking the presence of a cycle in **G** that uses no more than O(**E**) extra space and achieves a time complexity of less than or equal to O(**E**). Additionally, demonstrate that your approach adheres to the specified time and space complexity constraints.
19. Assuming **G** is an undirected graph, how can we determine if **G** is connected? Please devise a method to check the connectivity of **G** while adhering to both time and space complexities of less than or equal to O(**E**). Demonstrate that your approach satisfies the specified time and space constraints.
20. Assume that **G** is a Directed Acyclic Graph (DAG). Prove that it is always possible to find a topological order for **G**. Ensure that your proof is general and applicable to all cases, rather than being specific to a provided example.

Bonus Points Follow-Up Questions

> For each question, please provide your answer along with a brief yet concise explanation. Answers submitted without explanations will not be eligible for bonus points. You can earn a maximum of 10 bonus points for each question.

21. Following question 17, if we are provided with the preorder traversal sequence **P** and the inorder traversal sequence **I** of a binary tree **T**, how many different possible configurations of **T** are there? Please provide a proof for your answer.
22. In our presentation slides, we utilize a matrix representation of a directed graph to illustrate Dijkstra's Algorithm, which exhibits a time complexity of $O(V^2)$. However, this approach becomes inefficient when the degree of each vertex is low. Therefore, we need to identify an improved

method for implementing Dijkstra's Algorithm in cases where the degree of each vertex is low. Additionally, please provide the time and space complexities of your new implementation.

23. Assuming **G** is a directed graph, please devise a method for identifying all strongly connected components that utilizes no more than O(E) additional space and achieves a time complexity of O(E) or less.

Challenging Bonus Question

> For each question, please provide your answer along with a brief and concise explanation. Answers submitted without explanations will not receive any bonus points. You can earn a maximum of 20 bonus points.

24. Let's consider the development of a dictionary application that necessitates an efficient data structure for word storage. With a total of N words to be stored, each word having a maximum length of M characters, our objective is to create a data structure that satisfies the following criteria:

    1. The data structure must store all words using space less than or equal to O(M * N).

    2. It should support the insertion of a new word in O(M) time complexity. This operation is denoted as *insert()*.

    3. The data structure should facilitate word searching with a worst-case time complexity of O(M) and utilize O(1) extra space. The *search()* method should return a Boolean value of **true** if the word is currently stored; otherwise, it should return **false**.

    4. Furthermore, the data structure should allow the deletion of an existing word in O(M) worst-case time complexity while using O(1) extra space. This operation is labeled as *delete()*.

    - To achieve full marks, it is imperative to demonstrate that your approach aligns with the specified time and space complexity constraints.

Abbreviations and Notations

1. **G** = (**V**, **E**) denotes a graph **G** with vertices **V** and edges **E**. All graphs are simple graphs unless otherwise specified.
2. SCC(**G**) denotes the strongly connected components of the graph **G**.
3. |**S**| indicates the size of a sequence **S**.
4. BST is the abbreviation for Binary Search Tree.
5. MST is the abbreviation for Minimum Spanning Tree.
6. DAG is the abbreviation for Directed Acyclic Graph.