FINAL EXAMINATION

|  |  |  |  |
| --- | --- | --- | --- |
| TERM | COURSE NAME | COURSE CODE | VERSION |
| Fall 2023 | Data Structures and Algorithms | |  |  | | --- | --- | |  | [DSA 456-V1C (11180)](javascript:submitAction_win0(document.win0,'CLASS_TITLE$1');) | | A |

|  |  |
| --- | --- |
| Name | Neron Parmar |
| Student Number | 171690217 |
| Section | V1C |

DATE: Dec 13, 2023

TIME ALLOWED: 2 hours (120 minutes)

TOTAL MARKS: 100 marks

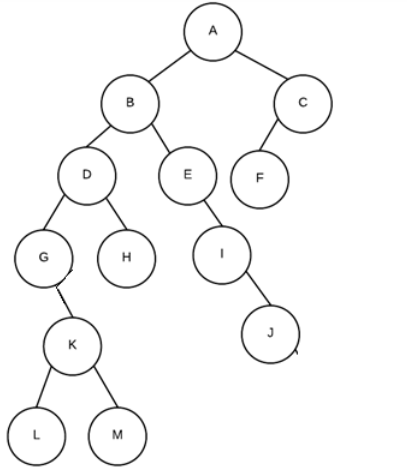
PROFESSOR(S): Elham Ahmadi SPECIAL INSTRUCTIONS:

As a Seneca student, you must conduct yourself in an honest and trustworthy manner in all aspects of your academic career. A dishonest attempt to obtain an academic advantage is considered an offense, and will not be tolerated by the College.

SENECA’S ACADEMIC HONESTY POLICY

**Question 1 (10 marks):  Tree Definitions**

 Given the following tree:



1. List the sibling(s) of D?

ANSWER 1: The sibling of D is E.

1. List all leaf nodes in the tree:

ANSWER 2: L, M, H, J, F are the leaf nodes.

1. What is the height of the tree?

ASNWER 3:

1. What is the height of subtree G?

ASNWER 4: 3 is the height of the subtree of G.

1. What is the depth of E?

ANSWER 5: the depth of the E is 2.

1. What is the height balance at D?

ANSWER 6: no height balance to be measured in.

1. What is the path from A to M?

ASNWER 8: The path is A-B-D-G-K-M from A to M.

1. What are the leaf nodes of the sub-tree with root “D”?

ANSWER 8: the leaf node of subtrees with the root D is L, M and H.

# Question 2 (15 marks): Tree shape definitions



A

B

G

C

E

D

F

Explain is the above tree complete? (1 marks)

ANSWER 1:

For the above tree complete, it is not complete. A complete binary tree is a tree in which the each level except the last level is completely filled and all the nodes are as left as possible mostly.

Explain is the above tree binary tree? (1 marks)

ANSWER 2:

The tree is a binary tree as each node has at most two children.

Explain is the above tree height balanced? (1 marks)

ASNWER 3:

Seeing about the tree height balanced, the tree is not height balanced. A binary tree is height balanced when the height of the two subtrees of every node differs by no more than 1. However, in the given tree. The left subtree that is rooted at the node B has a height of 2 whereas on the right subtree rooted at the node G has the height of 1.

What is the pre-order traversal of the tree? (3 marks)

ASNWER 4:

The pre-order traversal of the tree is A, B, C, D, E, F, G.

What is the in-order traversal of the tree? (3 marks)

ANSWER 5:

The in-order traversal of the tree above is D, C, B, E, F, A, G.

What is the post-order traversal of the tree? (3 marks)

ANSWER 6:

The post-order traversal of the tree is D, C, F, E, B, G, A.

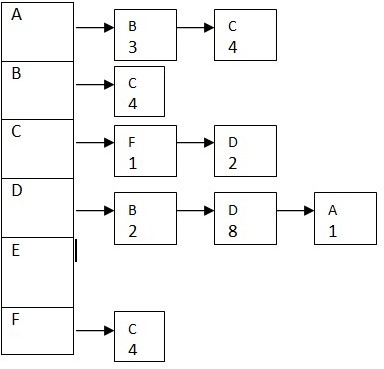
What is breadth-first traversal of the tree? (3 marks)

ASNWER 7:

The breath-first traversal of the tree is A, B, G, C, E, D, F.

## Question 3 (15 marks): Graphs

The following diagram of adjacency list represents a graph.



* 1. draw the above as a directed graph. Be sure to clearly show the “directions of edges”, and “edges weights” (10 marks)

A paper with lines and symbols

Description automatically generated

* 1. If the above graph was represented as an adjacency matrix instead of an adjacency list, what is the capacity of the data structure needed to represent the same graph? (5 marks)

# Question 4 (15 marks): Binary Heaps

Given the following array: {1, 3, 6, 5, 8, 7, 9, 2, 11, 4, 10}

1. Construct a max heap tree by inserting these number in the given order based on percolate up procedure. Show your full work step by step to get the full mark (8 marks). Note that whiteout step by step illustration and details you will get **Zero** mark.

**ANSWER a:**

Constructing the max heap tree on basis of percolate up procedure step by step. The steps are as follows:

At first we need to start with an empty heap and inserting the element one by one and then after that the percolate up procedure needs to be done. Array:{1, 3,6,5,8,7,9,2,11,4,10}

Step1: Inserting the element 1. So heap will be

Heap: [1]

Step 2: After insertion of 1 we will insert the element 3 so the heap will be

Heap: [3, 1]

Step 3: Now, to maintain the max heap property, the element 3 and 1 will be swapped and after swapping the heap will be

Heap: [1, 3]

Step 4: Now inserting the 6 after the swap we will get the heap like

Heap: [6, 1, 3]

After this insertion we will again have a swap of 6 and 3 to maintain the property

Step 5: now inserting the element 5 we will get the heap as follows

Heap : [6, 5, 3, 1]

Now after this we will again have a swap between 5 and 1

Step 6: Now by inserting the element 8. The heap obtained will be

Heap: [8,6,3,1,5]

So after this as done previously swapping the element 8 and 6.

Step 7: Now, inserting the 7 in we get the heap as

Heap: [8,7,3,1,5,6]

Now swapping the 7 and 6 elements.

Step8: so , inserting the element 9. The resulted heap will be

Heap: [9,7,8,1,5,6,3]

Now, swapping the 9 and 3 from their locations

Step 9: Insert element 2 in the heap

Heap: [9,7,8,2,56,3,1]

Similarly swapping the 2 and 8

Step 9: now inserting element 11.

Heap: [11, 7, 9, 2, 5, 6, 3, 1, 8]

And again swapping between elements 11 and 9.

Step 10: after that inserting the 4 we get heap as

Heap: [11, 7, 9, 2, 5, 6, 3, 1, 8, 4]

And later again swapping element 4 and 2.

Step 11: at last inserting the element 10 we get the heap

Heap: [11, 10, 9, 7, 5, 6, 3, 1, 8, 4, 2]

And after swaping 10 and 7.

The final max heap is we get is : [11, 10, 9, 7, 5, 6, 3, 1, 8, 4, 2]

1. Create a **maxheap** in place using the “heapify routine”. What is the final tree in array form. Show your work to get full marks (7 marks)

ANSWER b:

Now, creating a max heap in the place using the heapify routine is as follows :

Firstly, starting with the given array which is {11, 10, 9, 7, 5, 6, 3, 1, 8, 4, 2} the steps goes as follows,

Step 1: Start from the last non-leaf node that is index 4 and heapify down. And then swap 8 with 2: {11, 10, 9, 7, 5, 6, 3, 1, 2, 4, 8}

Step 2: After swapping 8 with 2, move to the next non-leaf node that is index 3 and heapify down. Swap 7 with 1 it will be {11, 10, 9, 1, 5, 6, 3, 7, 2, 4, 8} and again swap 9 with 1 so this will be {11, 10, 1, 7, 5, 6, 3, 9, 2, 4, 8}. Similarly, swapping element 11 with 1 so it will be {1, 10, 11, 7, 5, 6, 3, 9, 2, 4, 8}.

Step 3: Now, continuing this process for the remaining non-leaf nodes that are 10 and 11. At first, swapping 10 with 1 will be {1, 1, 11, 7, 5, 6, 3, 9, 2, 4, 8} and after this swapping 11 with 1 will be {10, 11, 1, 7, 5, 6, 3, 9, 2, 4, 8}

Thus after doing all this steps, the final max heap in array form is {10, 11, 1, 7, 5, 6, 3, 9, 2, 4, 8}

# Question 5 (15 marks):

Given the following array: {7, 3, 6, 5, 8, 17, 9, 2, 11, 4, 10}

1. Draw the binary search tree created by inserting each of the values in the array in the given order into a binary search tree (8 marks).

**ANSWER a:**

The binary search tree for the values in array is as follows:

First, we will insert the element 7:

7

After this we will insert the element 3 below the 7 on the left side

7

/

3

Now, as the 3 is inserted we insert the 6 and the 6 is greater than 3 so we place on the right hand side

7

/ \

3 6

After inserting the 6, we now insert 5 and this will again be placed on the right hand side of 7 and below 6.

7

/ \

3 6

\

5

Now, inserting the element 8 as the element 8 is greater than the element 6, 5 and 3 all the nodes we arrange it in the center of 6 and 5 and below 7.

7

/ \

3 8

/ \

6 5

Now, inserting the element 17 we place it below the 5 as it is greater

7

/ \

3 8

/ \

6 5

\

17

Now next step inserting 9 it is smaller than 17 but greater than the 8 so will be placed below 8.

7

/ \

3 8

/ \

6 9

\

17

Now, inserting 2 as it is smaller that the 3 we place it on the left side of the 7 in the tree below the 3.

7

/ \

3 8

/ \

2 6

\

9

\

17

Now we insert the element 11 after this and this will be place below the 17 on the left hand side as it is greater than the 9 and less than the 17

7

/ \

3 8

/ \ \

2 6 9

\

17

/

11

Now, inserting the element 4 this goes below 6 on the left side as it is smaller than the 6

7

/ \

3 8

/ \ \

2 6 9

/ \

4 17

/

11

Now, inserting the 10 in the tress it will go on the right hand side of the 7 below the 9 element node so the final bst tree will be:

7

/ \

3 8

/ \ \

2 6 9

/ / \

4 10 17

/

11

1. From constructed tree from part “a”, now remove node 17. How would be the final tree? (7 mark)

ANSWER b:

Now the removing of 17 node from the bst tree the final tree according to steps would be:

Step 1: Find the in-order successor of node 17, which is 11.

Step 2: After this, replace the value of node 17 with the value of its in-order successor (11).

Step 3: At last, delete the in-order successor (11).

Therefore, the final tree after removing node 17:

7

/ \

3 8

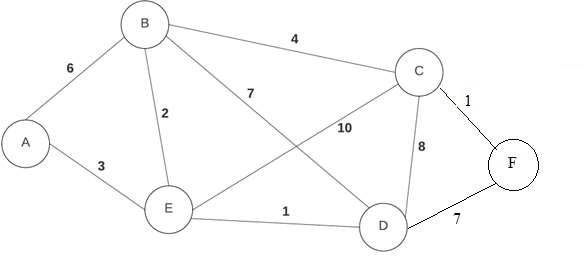
/ \ \

2 6 9

/ /

4 10

# Question 6 (10 marks): Graphs



Given the graph on the right, Use Dijkstra’s shortest path algorithm to find the shortest path from A to every other node. Fill in the two table to show how you got result. The first table is for rough work. To get part marks, cross out values as they are modified but leave them in the table so that it is clear to me where you may have made a mistake. If your rough work only contains final rough work table, you won't get part marks if it is wrong. Note that you will get **zero** mark without illustration and details.

rough work Final table that includes final values

|  |  |  |  |
| --- | --- | --- | --- |
| **Vertex** | **Shortest distance to A** | **Previous Vertex** | **Known** |
| A | 0 |  | true |
| B |  |  |  |
| C |  |  |  |
| D |  |  |  |
| E |  |  |  |
| F |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| **Vertex** | **Shortest distance to A** | **Previous Vertex** | **Known** |
| A |  |  |  |
| B | 5 | E |  |
| C |  |  |  |
| D |  |  |  |
| E |  |  |  |
| F |  |  |  |

**Final Result (2 marks)** From the above completed revised table, explain how the shortest path from A to F is defined? for example suppose your shortest path to D is A to B to C to D, put A-B-C-D for path entry.

# Programming: Please put all answers in the space provided. Question 7 (20 marks)

This question includes two parts “a”& “b”. Given the definition of a node and

binary search tree, you should complete the code of two functions in the next pages. Remember these functions must be defined based on concepts of BST. You can define extra functions if you needed.

class BST:

class Node:

# Node's init function

def init (self, value=None, left=None, right=None): self.value = value

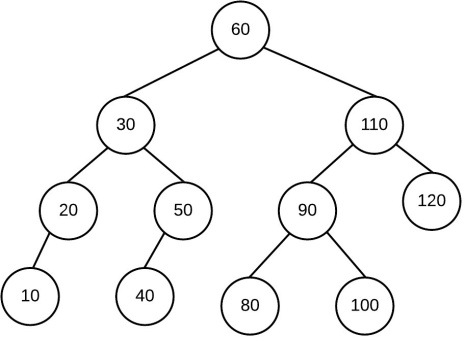
self.left = left self.right = right

# BST's init function def init (self):

self.root = None

Use this page if you needed any extra space for parts “a” and “b” answers.

1. Complete the “**Min\_Leafs\_Value()**” function (10 marks):

This function starts traversing the tree from root, and returns the smallest value of all leaf nodes values.

For given the tree to the right, function returns 10.

## def Min\_Leafs\_Value(self):

class Node:

def \_init\_(self, value=None, left=None, right=None):

self.value = value

self.left = left

self.right = right

class BST:

def \_init\_(self):

self.root = None

def Min\_Leafs\_Value(self):

# Check if the tree is empty

if not self.root:

return None # Or raise an exception

def find\_min\_leaf(node):

# If the node is a leaf

if not node.left and not node.right:

return node.value

# Recursively find the min value in both subtrees and return the min of them

min\_left = find\_min\_leaf(node.left) if node.left else float('inf')

min\_right = find\_min\_leaf(node.right) if node.right else float('inf')

return min(min\_left, min\_right)

return find\_min\_leaf(self.root)

# Example Usage

bst = BST()

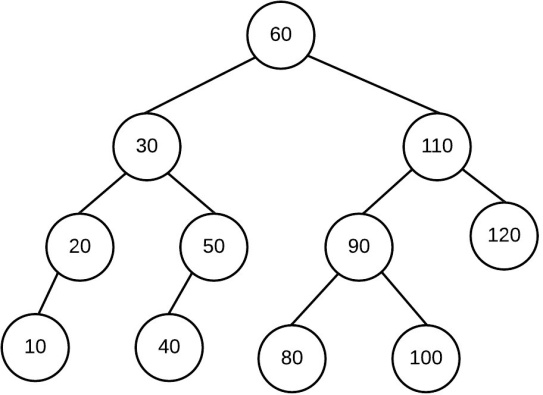
bst.root = Node(20)

bst.root.left = Node(15)

bst.root.right = Node(25)

bst.root.left.left = Node(10)

bst.root.left.right = Node(18)  
print(bst.Min\_Leafs\_Value()) # Should print 10

1. **def min\_of\_subtree(self, value) (10 marks)**

This function starts traversing the tree from “root” and returns the smallest value in the subtree with root node containing value. If the node does not exist, function returns -1

Example, given the tree on the right:

**min\_of\_subtree (15)** returns -1 as 15 does not exist

**min\_of\_subtree ( 110)** returns 80

**min\_of\_subtree ( 50)** returns 40

**min\_of\_subtree ( 30)** returns 10

def find\_min\_of\_subtree(self,value):

class Node:

def \_init\_(self, value=None, left=None, right=None):

self.value = value

self.left = left

self.right = right

class BST:

def \_init\_(self):

self.root = None

def find(self, value):

current = self.root

while current:

if current.value == value:

return current

elif current.value < value:

current = current.right

else:

current = current.left

return None

def min\_of\_subtree(self, value):

# Find the node with the given value

node = self.find(value)

# If the node is not found, return -1

if not node:

return -1

# Find the minimum value in the subtree

current = node

while current.left:

current = current.left

return current.value

# Example Usage

bst = BST()

# Add nodes to bst here

# For example:

bst.root = Node(100)

bst.root.left = Node(50)

bst.root.right = Node(110)

bst.root.left.left = Node(30)

bst.root.left.right = Node(60)

bst.root.left.left.left = Node(10)

bst.root.left.left.right = Node(40)

bst.root.right.left = Node(80)

print(bst.min\_of\_subtree(110)) # Should print 80

print(bst.min\_of\_subtree(50)) # Should print 40

print(bst.min\_of\_subtree(30)) # Should print 10

print(bst.min\_of\_subtree(15)) # Should print -1 as 15 does not ex