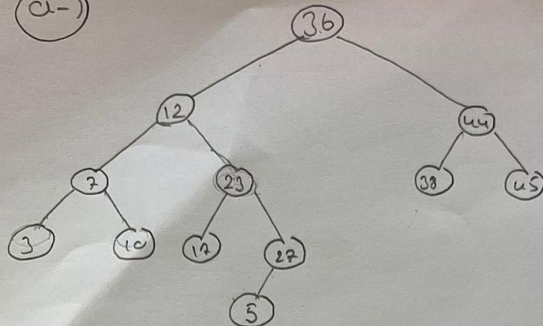


1

Mehmet Alperen Şahin

110 510 242

a-)

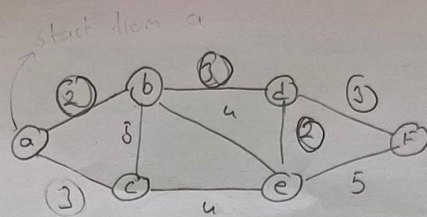


b-)

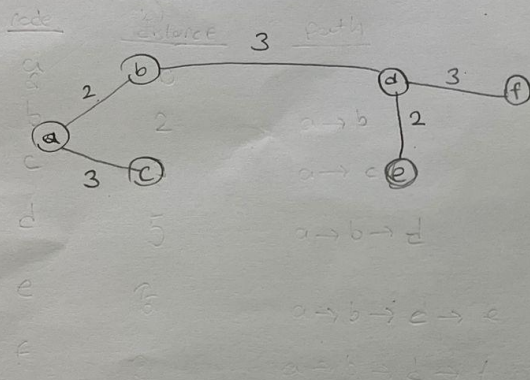
- 1- We define a class Node to represent a node in the binary search tree
- 2- The insertNode method recursively adds a new node to the binary search tree by comparing values and linking the appropriate position
- 3- The inorder Traversal method performs an in-order traversal of the tree, collecting node values in ascending order
- 4- The balanceBST method constructs a balanced binary search tree from a sorted list of values by recursively dividing the list and creating nodes
- 5- The balanceAndAddNode method combines previous steps. It adds a new node with value 42 to the tree, performs an in-order traversal, and builds a new balanced tree

c-) AVL Trees have a worst-case logarithmic height insertion with a time complexity of  $O(\log n)$  updating node heights and performing rotations for rebalancing also take  $O(\log n)$  time, making AVL trees suitable for maintaining balanced binary search trees

②



=>



$$\text{edge total weight} = 2 + 3 + 3 + 2 = 10$$

I started from e with the min. edge connected with d {e,d}, then the other min. edge same with ed is {a,b} = 2, then the other min edge value is 3, the edge with value of is {a,c}, {b,d} and {d,f}.  
 then i checked the other min value which is 4. we can not connect with edges 4 because it must be non cycled.

$$\text{edge weight total} = 2 + 2 + 3 + 3 = 10$$



③

```

a-) class Node:
    int data
    Node next

    function Node(data):
        this.data = data
        this.next = null

class stack:
    Node head
    int size
    int capacity

    function stack(capacity):
        this.head = null
        this.size = 0
        this.capacity = capacity

    function full() → boolean:
        return size equals capacity

    procedure push(data):
        if full():
            print "Can not push element"
            return
        newNode = Node(data)
        if head equals null:
            head = newNode
        else:
            newNode.next = head
            head = newNode
        size++
    
```

```

function pop() → int:
    if head equals null:
        print "cannot pop"
        return -1
    poppedData = head.data
    head = head.next
    size--
    return poppedData

function top() → int:
    if head equals null:
        print "stack empty"
        return -1
    return head.data
    
```

b-)

push() = this operations do not depend on the size of the stack so it is  $O(1)$

pop() = same with push() so is  $O(1)$

top() = same with push() so is  $O(1)$

full(): same with push so is  $O(1)$



all four methods have a time complexity of  $O(1)$  indicating that their execution time does not increase with the size of

4)

a-) Step 1: Node [E, 0] permanent-node  
 Node1 [E, F, 1] temporary-node  
 Node2 [E, C, 9] temporary-node  
 Node3 [E, D, 3] temporary-node  
 Node4 [E, B, 6] temporary-node

Step 2: Nodex [E, F, 1] permanent-node  
 Nodey [E, F, D, 3] temporary-node  
 Node z [E, F, C, 4] temporary-node

$$\min(\text{Node}(\text{Node}[x, y, z]) = \min \text{node}(E, F, D, 3)$$

Step 3:

node [E, F, D, 3] permanent-node  
 node [E, F, D, B, 8] temporary-node  
 node [E, F, D, B, 11] temporary-node

→ min is node [E, F, D, B, 8]

Step 4:

node [E, F, D, B, 8] permanent-node  
 node [E, F, D, B, A, 11] temporary-node  
 node [E, F, D, B, E, 14] temporary-node

→ min is node [E, F, D, B, A, 11]

result is  $\Rightarrow E-F-D-B-A \rightarrow 11$

b-) the time complexity of Dijkstra's algorithm in terms of the number of vertices  $V$  is typically  $O((V+E) \log V)$



⑤

Mehmet Alperen Sahin

110 510 242

Decision variables

$x$  = number of sofas to produce

$y$  = number of tables to produce

objective function:

$$\text{Maximize } z = 9x + 7y$$

Constraints:

$$5x + 4y \leq 300 \text{ (carpentry hours constraint)}$$

$$x + 2y \leq 90 \text{ (painting hours constraint)}$$

$$x \geq 0$$

$$y \geq 0$$

in the updated model, the painting hours constraint has been correct to  $x + 2y \leq 90$ , indicating that the total number of painting hours required for producing sofas and tables should not exceed the available 90 hours.