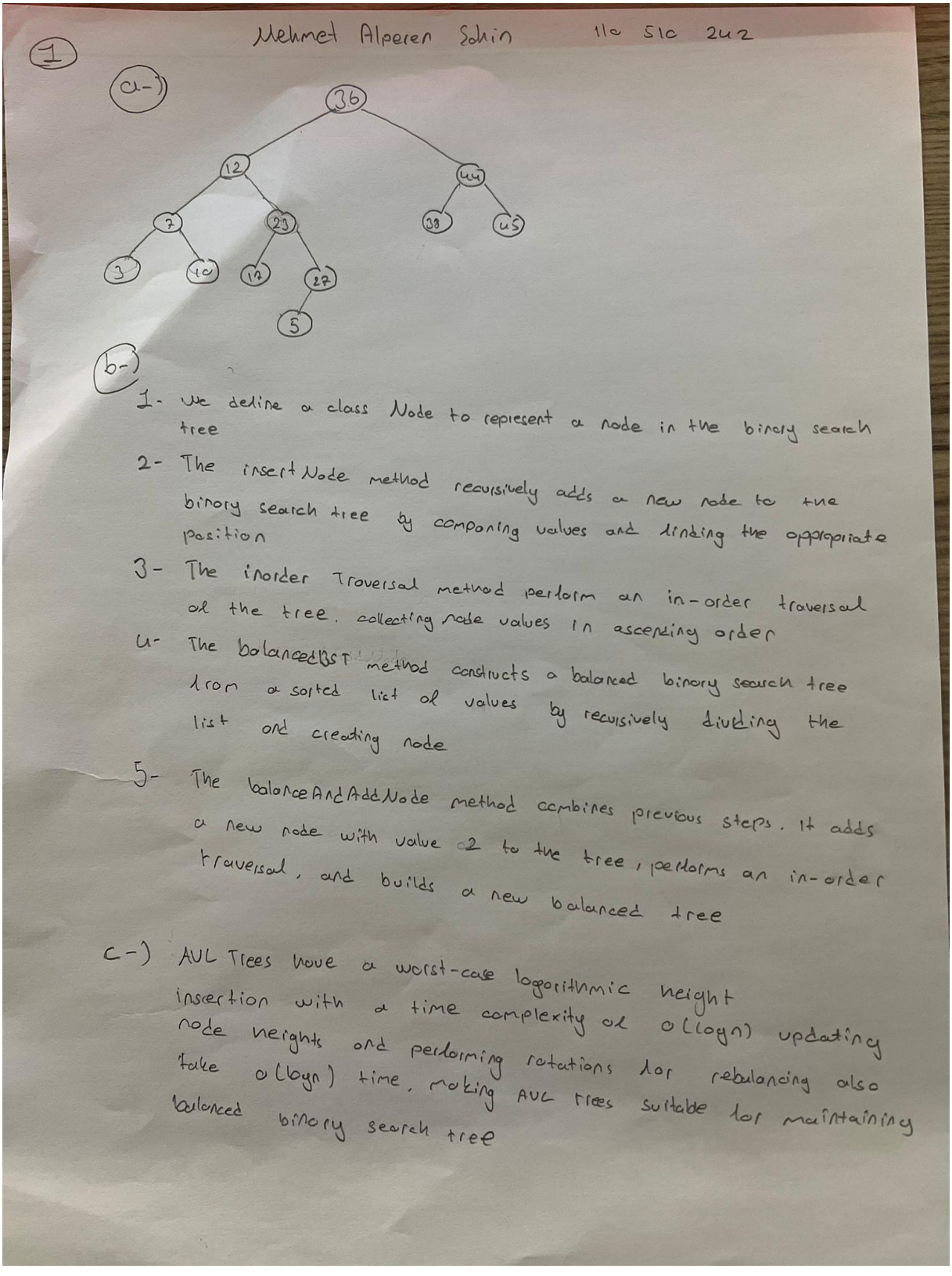
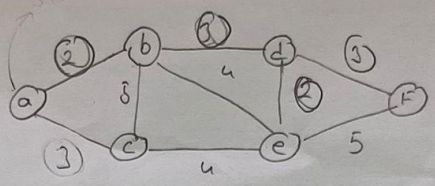
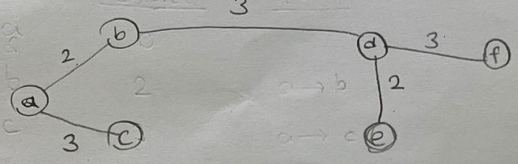
Mehmet Alperen Şahin 110510242



(2)



=>



edge total weight = 2+3+3+3+2
= 13

I started from e with the min. edge connected with d {e,d}, than the other min. edge same with e,d 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} connected these edges also. then i checked the other min value which is 4. we can not connect with edges 4 because it must be non cycled.

Mehmet Alperen Sahin 110 510 242

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 "cannot 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

Mehmet Alperen Sahin 110 516 242

④

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

Step 2:
NodeX[E,F,1] permenent node
NodeY[E,F,D,3] tempory-node
node 2[E,F,C 4] tempory-node

min(Node(node[x,y,z]) = min node(E,F,D,3)

Step 3:
node[E,F,D,3] permenent-node
node[E,F,D,B,8] temporray-node
node[E,F,D,B,11] tempory-node

↳ min is node[E,F,D,B,8]

Step 4:
node[E,F,D,B,8] permenent-node
node[E,F,D,B,A,11] tempory-node
node[E,F,D,B,E,14] tempory-node

↳ min is node[E,F,D,B,A,11]

result is => E-F-D-B-A → 11

---

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

Mehmet Alperen Sahin 110 510 242

(5)

Decision variables

x = number of sofas to produce

y = number of tables to produce

objective function:

Maximize $Z = 9x + 7y$

Constraints:

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

$x + 2y \leq 90$ (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.