Bradley Mont

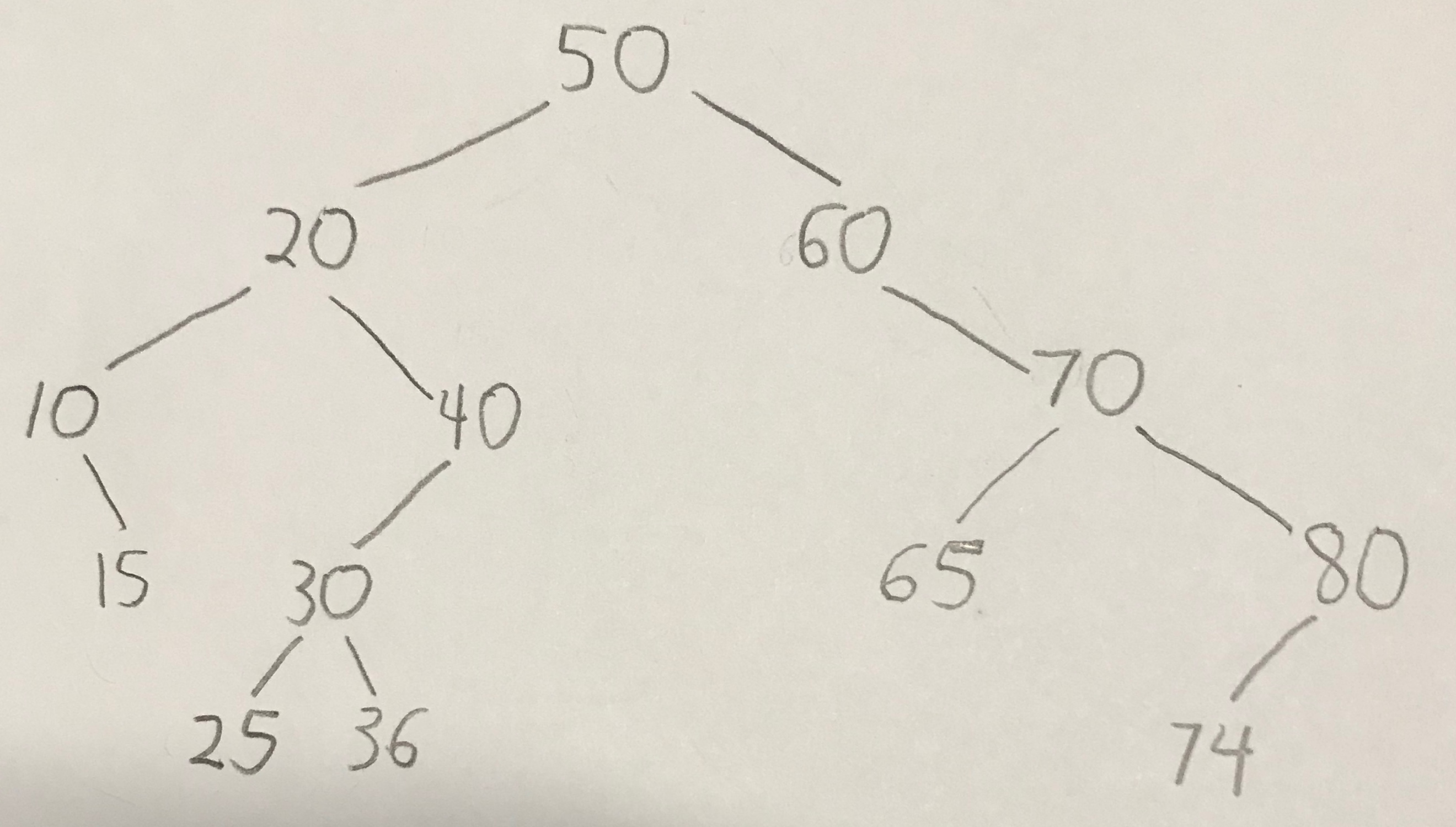
Com Sci 32

Professor Smallberg

Lecture 2, Discussion 2C

UID: 804-993-030

Homework 5

1a.

1b.

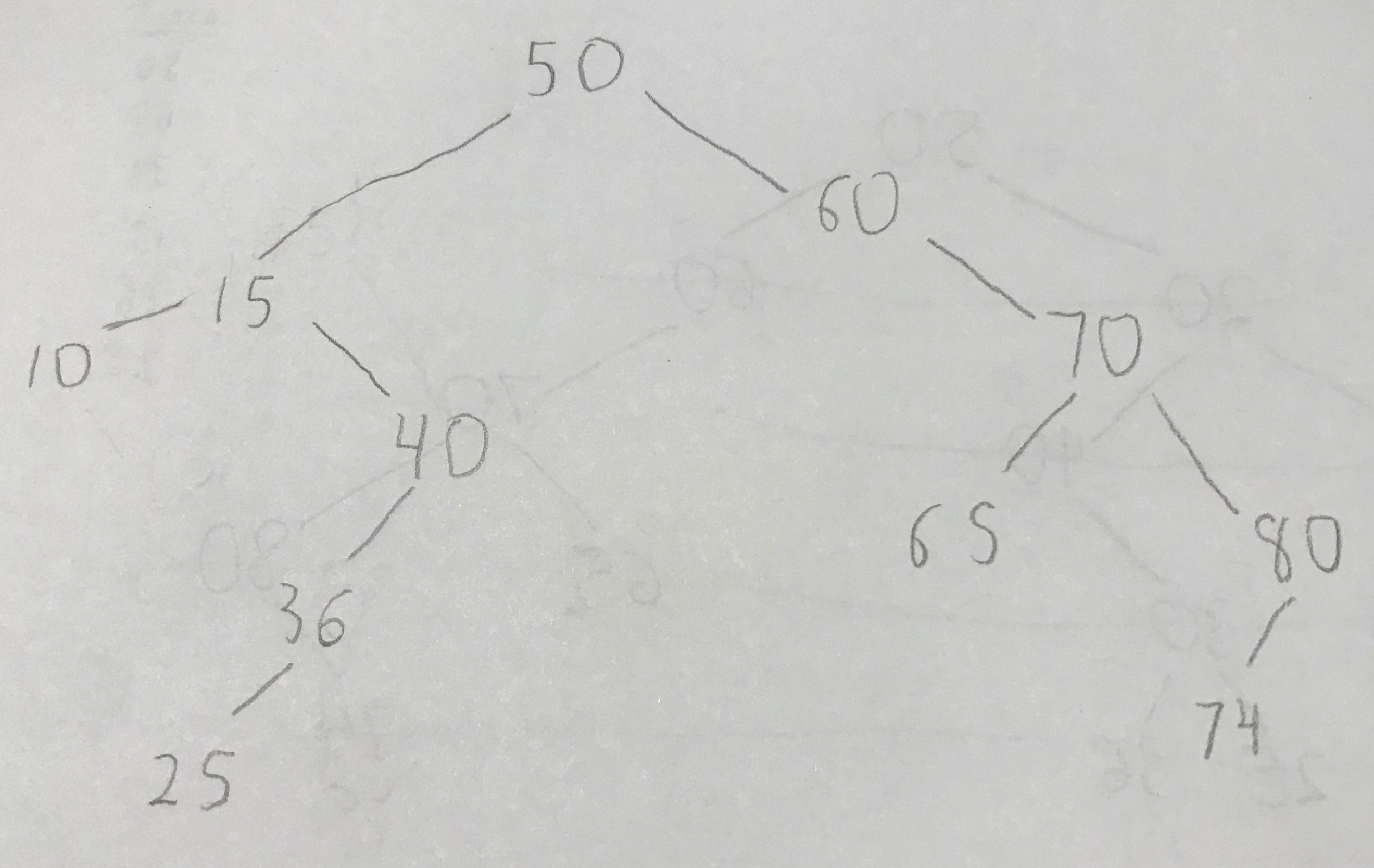
pre-order: 50 20 10 15 40 30 25 36 60 70 65 80 74

in-order: 10 15 20 25 30 36 40 50 60 65 70 74 80

post-order: 15 10 25 36 30 40 20 65 74 80 70 60 50

level-order: 50 20 60 10 40 70 15 30 65 80 25 36 74

1c.



2a.

struct Node

{

Node(int myVal)

{

value = myVal;

left = right = parent = nullptr;

}

int value;

Node \*left, \*right, \*parent;

};

2b.

Input: A value v to insert

If the tree is empty

Allocate a new node and put V into it (with left = right = parent = nullptr)

Point the root pointer to our new node

Start at the root of the tree

While we haven’t reached the end of the tree

If V is equal to the current node’s value

Exit the function without doing anything (the value already exists)

If V is less than the current node’s value

If there’s a left child

Go left (set the current node equal to its left pointer)

Else

Allocate a new node and put V into it. Set current node’s left pointer to the new node. Set the new node’s parent pointer to the current node. Exit the function.

If V is greater than the current node’s value

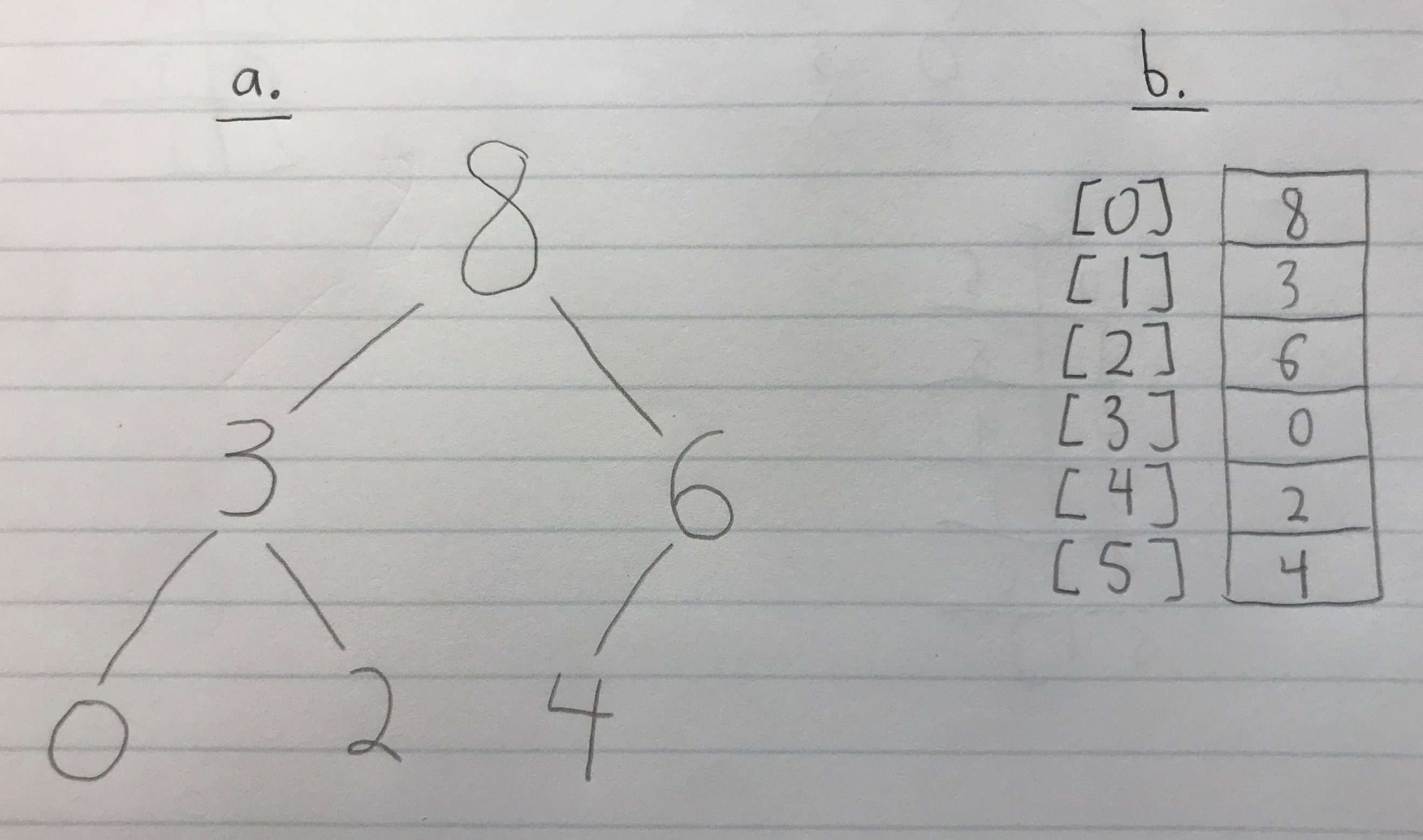
If there’s a right child

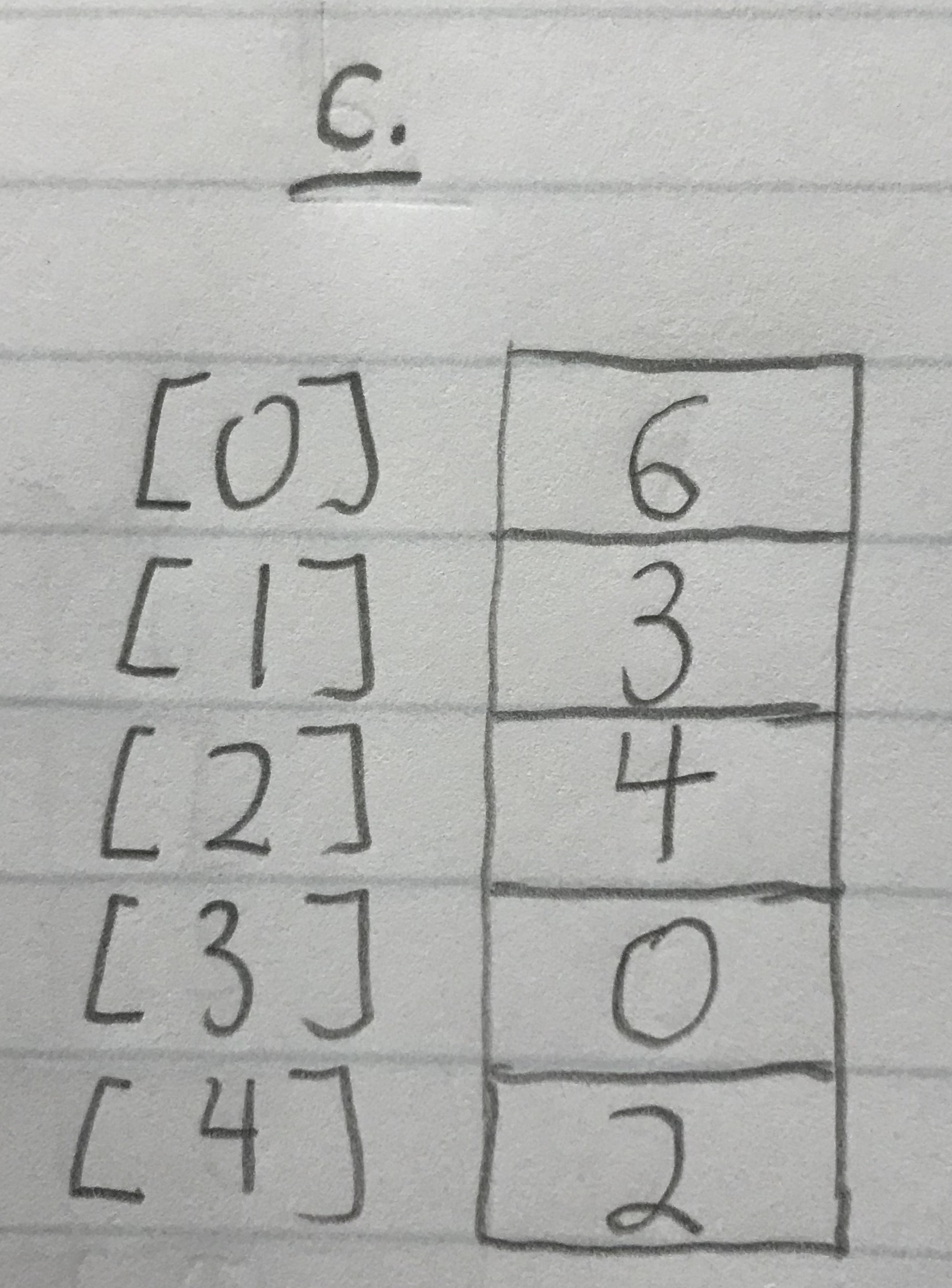
Go right (set the current node equal to its right pointer)

Else

Allocate a new node and put V into it. Set current node’s right pointer to the new node. Set the new node’s parent pointer to the current node. Exit the function.

3.





4.

a. O(C + S)

b. O(logC + S)

c. O(logC + logS)

d. O(logS)

e. O(1)

f. O(logC + S)

g. O(SlogS)

h. O(ClogS)