

Week 9 Q&A

- 12 Oct 2022 10-12

MOCK PE

- Date: 22 Oct 2022
- Time: Session 1: 2-3pm
Session 2: 3-4pm
- Venues: see seating plan in Luminus
- Coverage: up to Hashing
- Check MOCK PE folder in Luminus

Mock PE

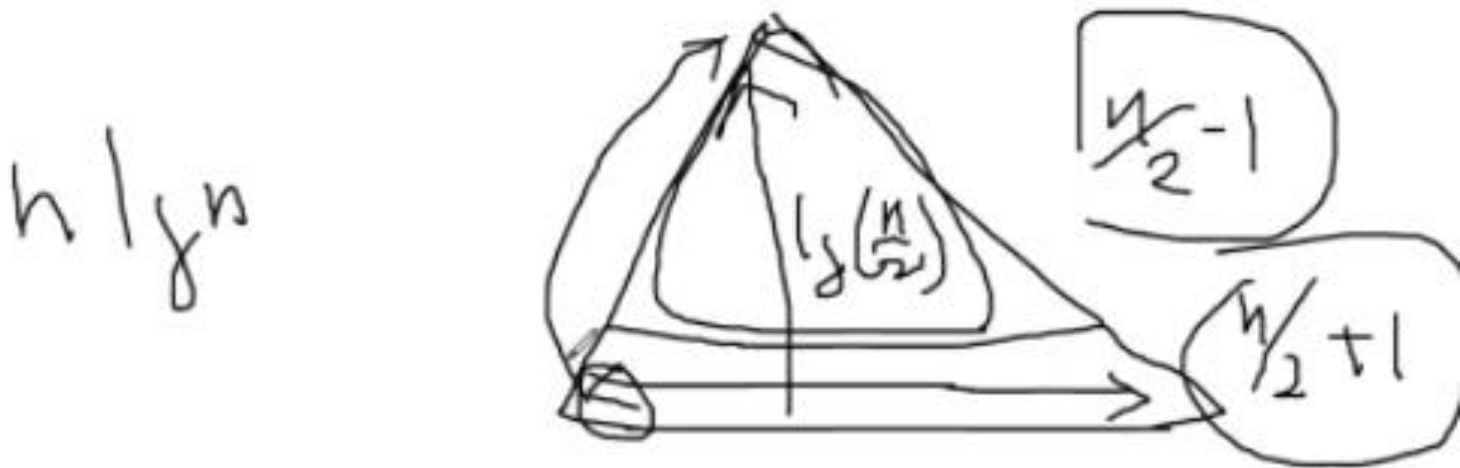
- What you need to do?
 - Make sure you check the Plab account given to you is the same one you sign for.
 - Write your name on the plab account slip.
 - Enter your Name, Plab account at the top of your program.

Mock PE

- What you do not need to do?
 - Transfer files
 - Submit to Codecrunch
- What you must not do?
 - Create new java files
 - Change the name of the java file given

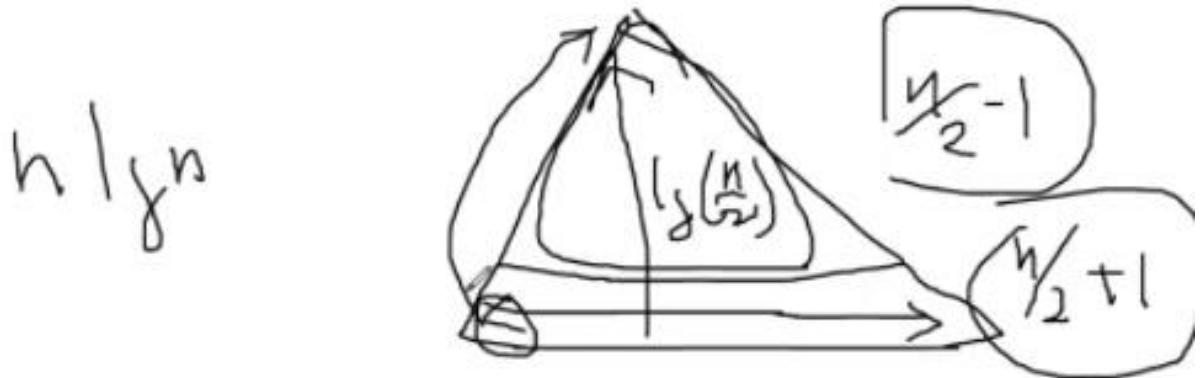
- Hi prof, May I know for a full tree, the number of nodes of the bottom level is $n/2 + 1$ instead of $(n+1)/2$?

Heap Construction



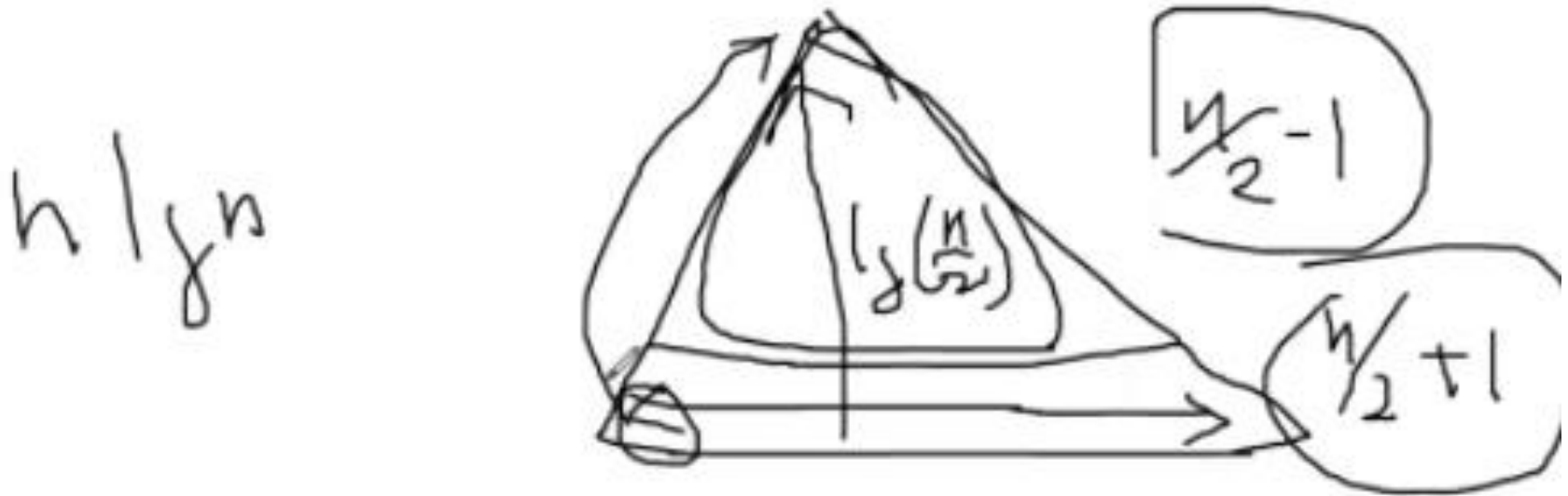
- I don't really understand why building a heap of n elements by inserting it 1 at a time is $O(n \log n)$. I understand that inserting 1 element is $O(\log n)$, so inserting n elements can be $n \cdot \log n$. But doesn't the height of the tree varies with insertion ($h=0$ when 1 element is inserted, $h=1$ when 1 subsequent element is inserted, $h=2$ when 2 subsequent elements are inserted, $h=3$ when 4 elements are subsequently inserted etc)? So the height is not always $\log n$?

Heap Construction



- Also, how can we calculate the number of elements in the highest level as shown in the picture below?

Heap Construction




- I have a few questions for the Heap part of this week's lecture:
- In slide 37, why is the retrieval time for top k pages $O(k \log N)$ if heap can be represented as an array with random access $O(1)$? Or is it because we need to traverse to the k th element in $\log(N)$ time, but then how does traversal work in heap?
- Can you explain why building a heap is $O(N)$ again?
- If constructing a heap is $O(N)$, how is heapsort $O(N \log N)$ if you need to build one heap for every element?

- Given an array representing a MinHeap, how do you convert it to a MaxHeap, in-place and in linear time. That is, in $O(n)$ time.

6. Suppose we relax the property of AVL trees so that the height difference between the left and right sub-tree is at most 2. Given *any* such AVL tree with this relaxed property, how many rotations do we have to perform in the worst case after an insertion to maintain the property?
- A. 0
 - B. 1
 - C. 2
 - D. $O(\log N)$ where N is the number of nodes in the AVL tree.
7. Let T be a complete binary tree with N nodes and height h . For a node in T at level i , the *weight* of a node is defined as $h - i$. What is the sum of the weights over all nodes in T ?
- A. $O(1)$
 - B. $O(\log N)$
 - C. $O(N)$
 - D. $O(N \log N)$

12) Which of the following is true of a priority queue?

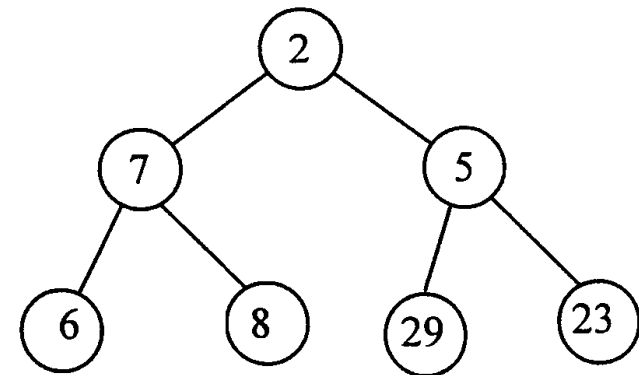
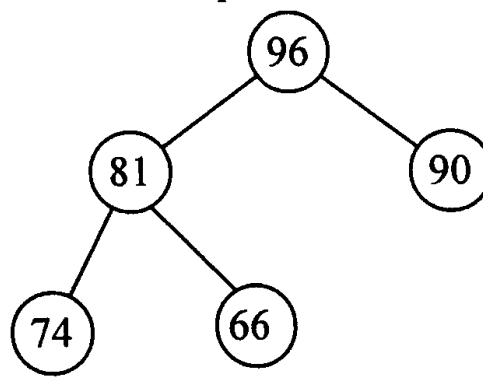
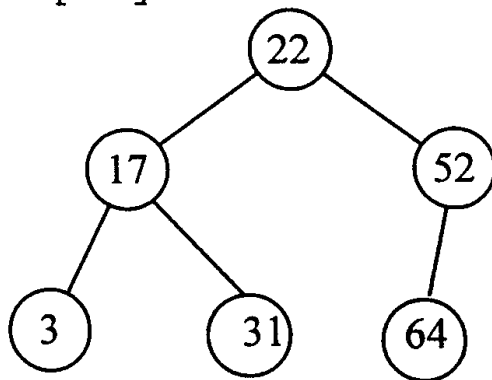
- a) If elements are inserted in increasing order of priority (i.e, lower priority element inserted first), and all elements are inserted before any are removed, it work like a queue.
- b) If elements are inserted in increasing order of priority (i.e, lowest priority element inserted first), and all elements are inserted before any are removed, it works like a stack. 
- c) If all elements are inserted before any are removed, it works like a queue.
- d) If elements are inserted in decreasing order of priority(i.e, highest priority element inserted first), and all elements are inserted before any are removed, it works like a stack.
- e) If elements are inserted in increasing order of priority, then it works like a queue whether or not all insertions precede any removals.

14 Which of the following is true about the heapsort?

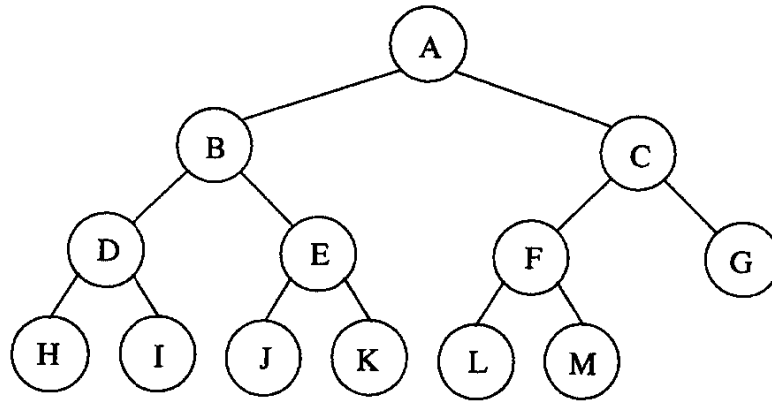
- a. heapsort does not require a second array
- b. heapsort is more efficient than the mergesort in the worst case
- c. heapsort is more efficient than the mergesort in the average case
- d. heapsort is better than the quicksort in the average case
- e. heapsort is stable

Question 7 (Heap)

For each tree as follows, indicate whether it is a maxheap. If it is not, write the resulting tree after each loop of `heapify` function until it becomes a maxheap.



13. (6 points)



The above shows a minimum heap. The values stored in the heap are not shown. You can assume that the values in this heap do not contain duplicates. The nodes in the heap are labeled with letter A to M.

- (a) Write down all possible nodes that could contain the second minimum value in the heap.

Answer:

- (b) Write down all possible nodes that could contain the third minimum value in the heap.

Answer:

- (c) Write down all possible nodes that could contain the maximum value in the heap.

15. (14 points) The teaching staff for CS1100 has a system called Automatic Judge, or AJ, that automatically grades students assignments on a first come, first serve basis. AJ maintains a queue – each new submission is entered into the queue. When AJ finishes grading one submission, it removes the next submission from the queue to grade.

Students have complained of long waiting time in the queue. To reduce the number of complaints, the teaching staff decided to assign priorities to submissions. Students who complain more will have higher priority for their submissions. When AJ finishes grading one submission, it always chooses the submission with highest priority to grade. If two submissions have the same priority, then the one with earliest submission time will be chosen.

- (a) Suppose the priority values of the submissions can *only* be 1, 2, or 3, with 3 being the highest priority. Describe an efficient data structure you can use to replace the queue. You may quote any of the data structures taught in the class without going into details.

^ -----

What is the running time for inserting a new submission?

Answer:

-
- (b) Suppose the priority values of the submissions are real numbers between 0 and 1, with 1 being the highest priority. Describe an efficient data structure you can use to replace the queue. You may quote any data structures taught in class without giving details.

Answer:

16. (16 points) In this question, we consider the following problem:

Given a set of N integers, find the k smallest integers from the set.

(a) For each of the algorithm given below, give the tightest running time of the algorithm in terms of N and k .

i. Run Selection Sort on the integers, but stop after k iterations.

Answer:

ii. Run Mergesort on the integers, then return the first k items after sorting.

Answer:

iii. Build a minimum heap from the integers, then delete k times.

Answer:

(b) Give a $O(k + N \log k)$ algorithm to solve the given problem.

Question 4 [Heap and Graph 8 marks].

- (a) A *ternary heap* is like a binary heap, but instead of two children, each node can have three children (the left child, the middle child, and the right child). Consider the implementation of a ternary heap using an array A . Assuming that the root is stored in array position 1, write down the expression for calculating the position of

- (i) the parent of a node $A[i]$.

Answer:

- (ii) the right child of a node $A[i]$.

Answer:

Question 6 (15 marks)

A double-ended priority queues allows access to both the minimum and maximum elements. In other words, all of the following are supported: `findMin`, `deleteMin`, `findMax`, and `deleteMax`. Do the following:

- a) [3 marks] Describe a data structure that will support such double-ended priority queues.

- b) [9 marks] Describe the algorithms for `insert`, `findMin` and `deleteMax`.

- c) [3 marks] What is the Big-O running time for each of your three algorithms in (b)?

`insert` `findMin` `deleteMax`

All heaps in the following exercises are MAX-heaps.

[6 marks] Below is an unsorted array of integers. Show the contents of the array after running `heapify()`.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|----------------|---|---|----|---|---|----|---|----|----|----|
| Original Array | 1 | 8 | 13 | 5 | 9 | 17 | 6 | 14 | 20 | 25 |
| After heapify | | | | | | | | | | |

[3 marks] The array below represents a heap. Show the contents of the array after executing a `heapDelete()` operation.

| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|------------------|----|----|----|----|----|----|----|----|---|----|
| Original heap | 90 | 70 | 65 | 38 | 11 | 60 | 61 | 15 | 1 | 10 |
| After heapDelete | | | | | | | | | | |

[3 marks] Explain how to implement a stack using a priority Queue. Is this implementation more efficient (in terms of time complexity) compared to a LinkedList-based implementation of the stack? Explain.

Question 6 (10 marks)

Given the following values: 34, 67, 12, 90, 37, 82, 22

- a) Create a **maxHeap** by inserting the elements one at a time. In the table, show the heap after each element is inserted and heap property re-established (3 marks)
- b) Using heap construction to create a **minHeap**. When you heapify an internal node, it is considered as one step. (3 marks)
- c) Which heap (maxHeap (a) or minHeap (b)) should be used if we want to create a list sorted in descending order? Show it using the following table (4 marks)

| | | | | | | |
|--|--|--|--|--|--|--|
| | | | | | | |
|--|--|--|--|--|--|--|

Question 7.

7 (A) Let H be a heap represented by an array with the key values:

29, 18, 17, 4, 3, 5, 1, 2

Show the contents of the array after inserting key value 20 into the heap and then deleting the maximum key value from the heap. (5 marks)

| | | | | | | |
|--|--|--|--|--|--|--|
| | | | | | | |
| | | | | | | |

6. Which of the following statement(s) is/are true regarding insertion in BST?

- I) Insertion always increase the tree height
- II) Insertion always occurs at leaf level
- III) It is possible for insertion to traverse the whole tree

- a) I only
- b) I and II only
- c) II and III only
- d) I and II only
- e) I, II and III

7. Which of the following statement(s) is/are true regarding deletion in BST?

- I) Deletion always results in removal of a leaf node
- II) During the recursive deletion function, the target value to be removed may change.
- III) Deletion may cause several values to change place in the BST

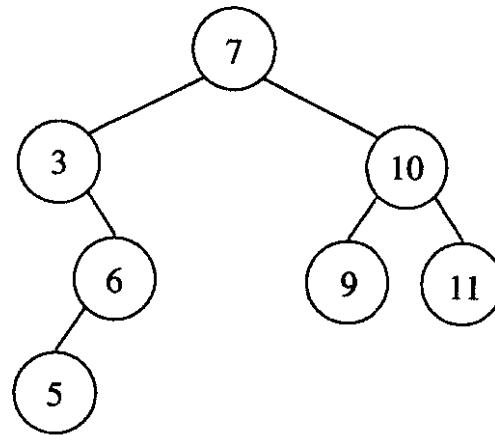
- a) I only
- b) I and II only
- c) I and III only
- d) II and III only
- e) I, II, III

9. If $\{A, F, D, G, P, K, H, X\}$ is the pre-order sequence of a binary tree and the in-order sequence of the same binary tree is $\{D, F, P, G, A, K, X, H\}$, which of the following is the post-order of the same tree?

- a) $\{X, P, H, G, D, K, F, A\}$
- b) $\{P, X, D, G, H, F, K, A\}$
- c) $\{D, P, G, F, H, X, K, A\}$
- d) $\{D, P, G, F, X, H, K, A\}$
- e) NONE of the above

6. Suppose we relax the property of AVL trees so that the height difference between the left and right sub-tree is at most 2. Given *any* such AVL tree with this relaxed property, how many rotations do we have to perform in the worst case after an insertion to maintain the property?
- A. 0
 - B. 1
 - C. 2
 - D. $O(\log N)$ where N is the number of nodes in the AVL tree.
7. Let T be a complete binary tree with N nodes and height h . For a node in T at level i , the *weight* of a node is defined as $h - i$. What is the sum of the weights over all nodes in T ?
- A. $O(1)$
 - B. $O(\log N)$
 - C. $O(N)$
 - D. $O(N \log N)$
- (b) Which of the following operations CANNOT be done in $O(h)$ time on a binary search tree of height h ?
- A. Calculating the height of the tree.
 - B. Finding an element.
 - C. Finding the third largest element.
 - D. Deleting the root.

12. (6 points)

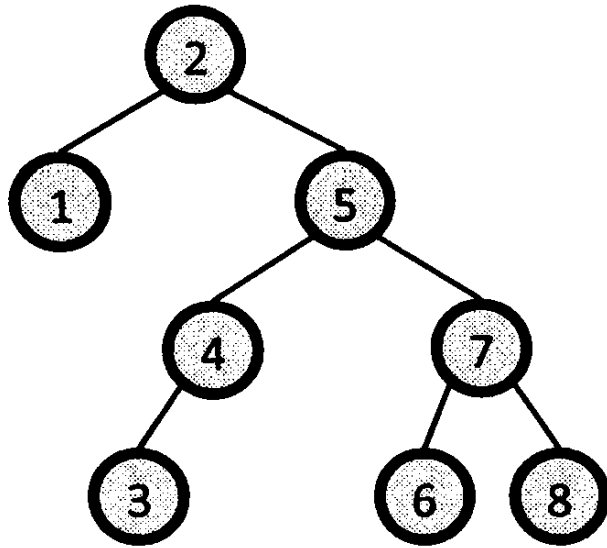


- (a) The figure above shows the result of an insertion (without rebalancing) into a originally balanced AVL tree. Which node has just been inserted?

Answer:

- (b) The AVL tree property has been violated by the insertion in the figure above. Rebalance the tree, and draw the resulting tree in the space below.

B. [6 marks] Write an algorithm to print all the leaf nodes of any given binary search tree in reverse order, i.e. decreasing order. For example, the output for the below binary search tree are: 8, 6, 3, 1. What is the complexity of your algorithm?



Question 6: (10 marks)

Part I) (7 marks)

Write in pseudo code a recursive algorithm to count the number of leaf nodes in a binary tree.

- 4a. We can uniquely construct a binary tree if the inorder traversal and either preorder or postorder traversal of that tree are given.

Given inorder traversal is J Z C A P W N K
and postorder traversal is Z J C N W P K A, construct the tree below.

(3 marks)

- 4b. We can also uniquely construct a binary tree when preorder and postorder traversals are given together with the condition that all the nodes in the tree have either zero or 2 children.

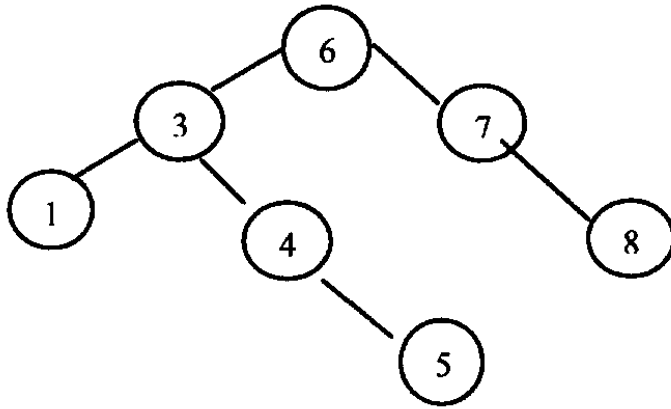
Construct the tree if preorder traversal is A C J S Z D K P B F E
and the postorder traversal is S Z J D C P F E B K A

all the nodes in this tree has either zero or two children.

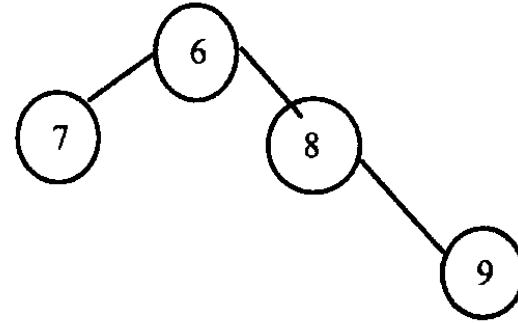
(5 marks)

- 3) We can save a binary tree in a file using the following sequence(s) in order to be able to restore the binary tree to its original shape later.
- a) in-order sequence
 - b) pre-order sequence
 - c) in-order sequence and post-order sequence ←
 - d) pre-order sequence and post-order sequence
- 4) What is the minimum size of a complete binary tree with height h ?
- a) $2^{h-1} - 1$
 - b) 2^h
 - c) 2^{h-1}
 - d) $2^h - 1$
- 5) What is the complexity of an algorithm for testing whether a binary tree with n nodes (may have duplicate values) is a binary search tree in the best case?
- a) $O(n)$
 - b) $O(n \log n)$
 - c) $O(n^2)$
 - d) $O(\log n)$

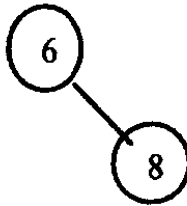
6) Consider the 3 trees below:



Tree T1



Tree T2



Tree T3

- a) Only T1 is an AVL trees
- b) Only T2 is not an AVL tree
- c) Only T3 is an AVL tree
- d) All the 3 trees are AVL trees

Question 4 (21 marks)

a) [4 marks] Show the expression tree which represents the arithmetic expression:

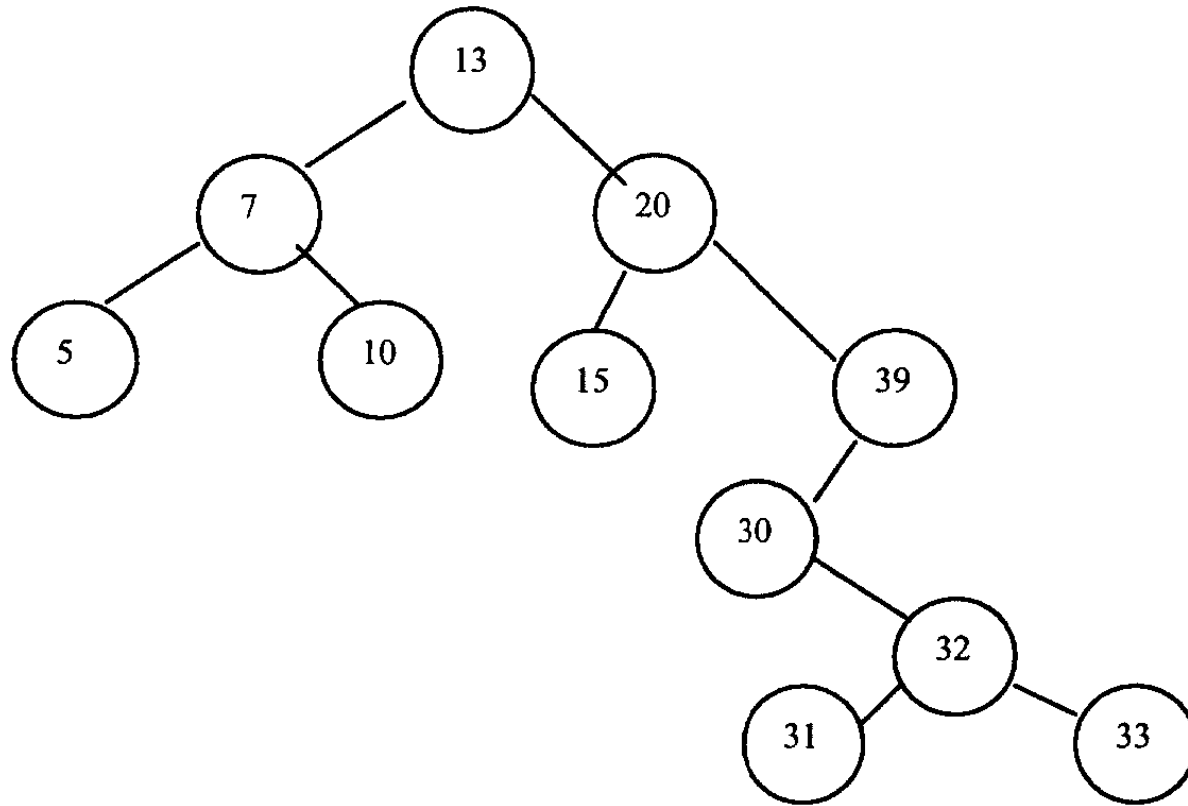
$$A * B^N - (C / D - E)$$

C. [3 marks] What are the minimum and maximum numbers of internal nodes in a binary tree with 20 nodes?

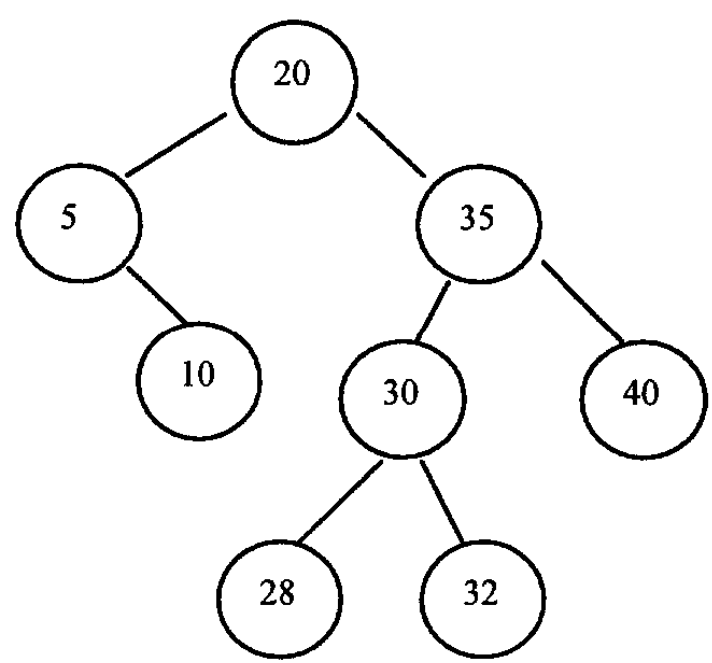
III) A. [4 marks] Construct the original binary search tree from its post-order traversal sequence:

{9, 8, 7, 5, 13, 15, 22, 20, 10}.

- b) **[4 marks]** Show the resulting binary search tree after deleting 20 from the binary search tree below:



c) **[8 marks]** Show the resulting AVL tree after inserting 33 into the AVL tree below:



Question 4: (15 marks)

- a). Insert the following values in the order given into an empty AVL tree. You may continue on the same tree until you need to balance it. Indicate whether the violation is due to insert-inside or insert-outside, then show the final balanced tree. (7 marks)

The values: 48, 32, 65, 37, 24, 41, 78, 36, 52, 94, 70, 58, 73, 35

The tree before balancing:

- b) Delete the values 48, 35 and then 41 from the final balanced tree above. When you delete a node with two children, replace the node with its inorder predecessor and delete the inorder predecessor. Show the final balanced tree. (8 marks)

The tree after 48, 35 and 41 are deleted in that order:

6a) Describe an algorithm to check if a given sequence of numbers is a pre-order sequence of a BST without constructing the BST. For example, {12, 6, 3, 5, 7, 9, 8, 16, 19, 17} is a pre-order sequence, while {12, 6, 3, 7, 5, 9, 8, 16, 19, 17} is not. **(5 marks)**

- a) If we have a full tree of height h , what is the height of its left and right sub-trees?
- b) The function `checkFullTree()` checks if the current tree is a FULL tree. It returns true if it is a full tree and returns false otherwise. The function calls a sub-function `checkFullSubTree_()`. Your task is to implement `checkFullSubTree_()` **recursively** in C++ language. The function will check the input `subTree` if it is a full tree of height h .

Question 5 [Binary Search Trees 8 marks].

(a) A binary tree can be represented using the following structures.

```
public class TreeNode {  
    public TreeNode leftChild;  
    public TreeNode rightChild;  
    public int value;  
}
```

A binary search tree (BST) can be built using the above class, by maintaining the properties that all nodes whose values are less than the root are stored in the left subtree, and all nodes whose values are larger than the root are stored in the right subtree. A variant of BST, called Flipped BST (FBST), has a slightly different property – nodes whose values are *larger* than the root are stored into the *left* subtree instead, and nodes whose values are *smaller* than the root are stored in the *right* subtree.

Write the following Java method called `flip` that takes in a root to a BST and converts it into FBST.

Given roots of two BST T and T' of size N and M respectively, containing unique integers, write the algorithm for a merge operation which will merge the two BST into an AVL tree (and return the root of the new tree) in $O(N+M)$ time. [*5 marks]

- 1.) In-order traversal of T and T' and output the sequence into arrays A and A' respectively.
 A and A' contains the elements of T and T' in ascending order. --- $O(N+M)$
- 2.) Use merging function of merge sort to merge A and A' into an array B . --- $O(N+M)$

3.) Now convert B into AVL tree as follows

```
Vertex ConvertToAVL(B,left,right) {  
    if (left <= right)  
        Create Vertex V for B[mid]  
        V.left = ConvertToAVL(B,left,mid-1) & link left child to V  
        V.right = ConvertToAVL(B,mid+1,right) & link right child to V  
        return V  
    else  
        return null  
}
```

//This is pre-order creation of AVL tree.

Call $T'' = \text{ConvertToAVL}(B, 0, N+M-1)$ to perform the conversion

This takes time $O(N+M)$ since each node is processed only once.

4.) After step 3, the tree generated must be balanced. --- $O(N+M)$

Thus total time taken is $O(N+M)$

- Given an array A of N unsorted and non-repeated integers, output the M smaller and the M larger values than the median of the N integers, including the median. They must be output in ascending order. Assume that $M \geq 1$, N is odd and $N \leq 1,000,000,000$, and $2 \times M$ is much smaller than N ($2M \ll N$). Your algorithm must run in expected time $O(N \log M)$ or better.
- For example, given $[132, 13, 151, 15, 1, 41, 6]$ the median is 15. If $M=2$, then 6, 13, 15, 41, 132 will be output.
- Solve the problem in $O(N \log M)$ time or better using any data structure you have learned in CS2040 [10 marks]

- 1.) Quick select $M = \text{median-}m \rightarrow O(N)$ time
- 2.) Quick select $M' = \text{median+}m \rightarrow O(N)$ time
- 3.) Scan through the input again putting all numbers between M and M' into another list. $\rightarrow O(N)$
- 4.) Sort that list $\rightarrow O(M \log M)$ time

Total time is $O(\text{Max}(M \log M, N))$.

Quick select is the find kth smallest algorithm we discussed in the recursion lecture.

$O(N \log M)$ solution:

Use two AVL trees T and T' . T to store the M numbers smaller than the median and T' to store the M numbers larger than the median.

1. Use quick select algorithm to find the median (the $(N/2)$ th integer) in expected $O(N)$ time.
2. Go through the A from index $i = 0$ to $N-1$
 - If $A[i] < K$
 - If $|T| < M$: insert $A[i]$ into $T \rightarrow O(\log M)$
 - If $|T| \geq M \ \&\& \ A[i] > \text{findMin}(T)$: delete min and insert $A[i] \rightarrow O(\log M)$
since size of T is maintained at $|M|$
 - If $A[i] > K$
 - If $|T'| < M$: insert $A[i]$ into $T' \rightarrow O(\log M)$
 - If $|T'| \geq M \ \&\& \ A[i] < \text{findMax}(T')$: delete max and insert $A[i] \rightarrow O(\log M)$
since size of T' is maintained at $|M|$
3. perform inorder traversal on T and T' and output the values including $K. \rightarrow O(M)$

Total time taken is expected $O(N + N \log M + M) = O(N \log M)$ since $M \ll N$.

- Another Solution: Use a min heap H and a max heap H' . H' will store the M values larger than the median. H will store the M values smaller than the median.

1. Use quick select algorithm on A to find the $(N/2)$ th smallest element (the median). Let K store the median. \rightarrow expected $O(N)$ time

2. Now go through A from $i = 0$ to $N-1$

If $A[i] < K$

if $|H| < M$: Insert $A[i]$ into $H \rightarrow O(\log M)$

if $|H| \geq M$ & if min value in $H < A[i]$

extractMin(H) and insert $A[i]$ into $H \rightarrow O(\log M)$ since $|H|$ is maintained at M .

If $A[i] > K$

if $|H'| < M$: Insert $A[i]$ into $H' \rightarrow O(\lg M)$

if $|H'| \geq M$ & if max value in $H' > A[i]$

extractMax(H') and insert $A[i]$ into $H' \rightarrow O(\log M)$ since $|H'|$ is maintained at M .

Total time taken is $O(N \log M)$

3. perform heap sort on H and H' and output the values including K . $\rightarrow O(M \log M)$

Total time taken is $O(N + N \log M + M \log M) = O(N \log M)$ since $M \ll N$.


```

public TreeNode constructTree(int [] inOrder, int [] preOrder, int n) {
    int inorderIndex=0;
    //Base case, when there is only 1 item, it's trivial
    if(n == 1) return new TreeNode(inOrder[0]);
    if(n == 0) return NULL;

    //Root is always the first element of the preOrder array
    int element = preOrder[0];
    TreeNode root = new TreeNode(element);

    //Determine the location of the root element in the inOrder array
    for (int i=0; i<n; i++)
        if (inOrder[i] == preOrder[0]) {
            inorderIndex = i;
            break;
        }
    //codes to create new arrays to store the pre and in_order of left subtree
    root.left = constructTree(inorderLeft, preorderLeft, inorderIndex);
    // codes to create new arrays to store the pre and in order of right subtree
    root->right = constructTree(inorderRight, preorderRight, n - (inorderIndex+1));

    return root;
}

```