

COL106: Data Structures and Algorithms

Lecture 29

Keerti Choudhary
Department of Computer Science, IIT Delhi

Today's Lecture

1. Discussion of Minor-II

Banking System

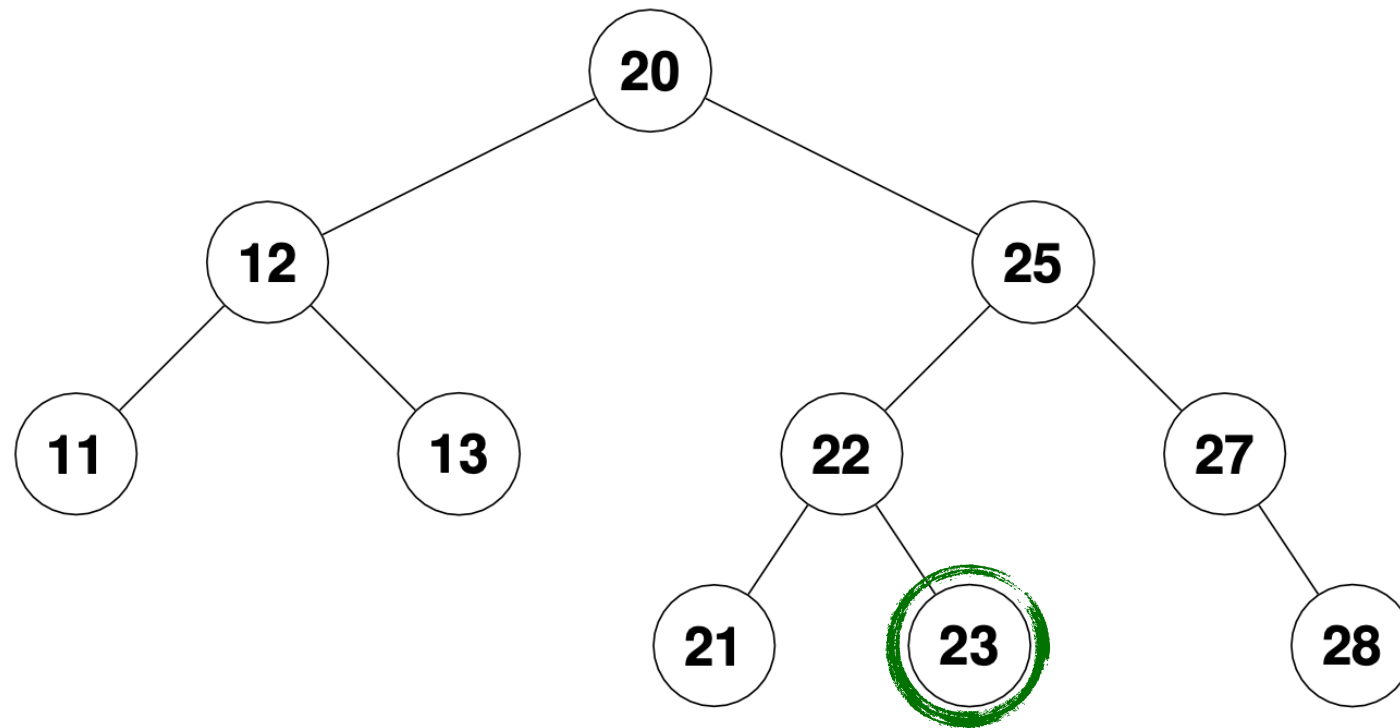
					
Registration id:	11	22	33	44	55
Balance:	600 rs.	800 rs.	200 rs.	500 rs.	800 rs.

Goal: Design a data-structure with following tasks:

1. **Add new user** with registration number r and opening balance b .
2. **Increment balance** of user r by an amount $+\delta$.
3. **Return registration id** of user with k^{th} -highest balance.

Space used must be $O(n)$, and all updates/queries must be handled in $O(\log n)$ time.

Simpler Question: Find k -th smallest element in BST



Question: How can you find 7th smallest element?

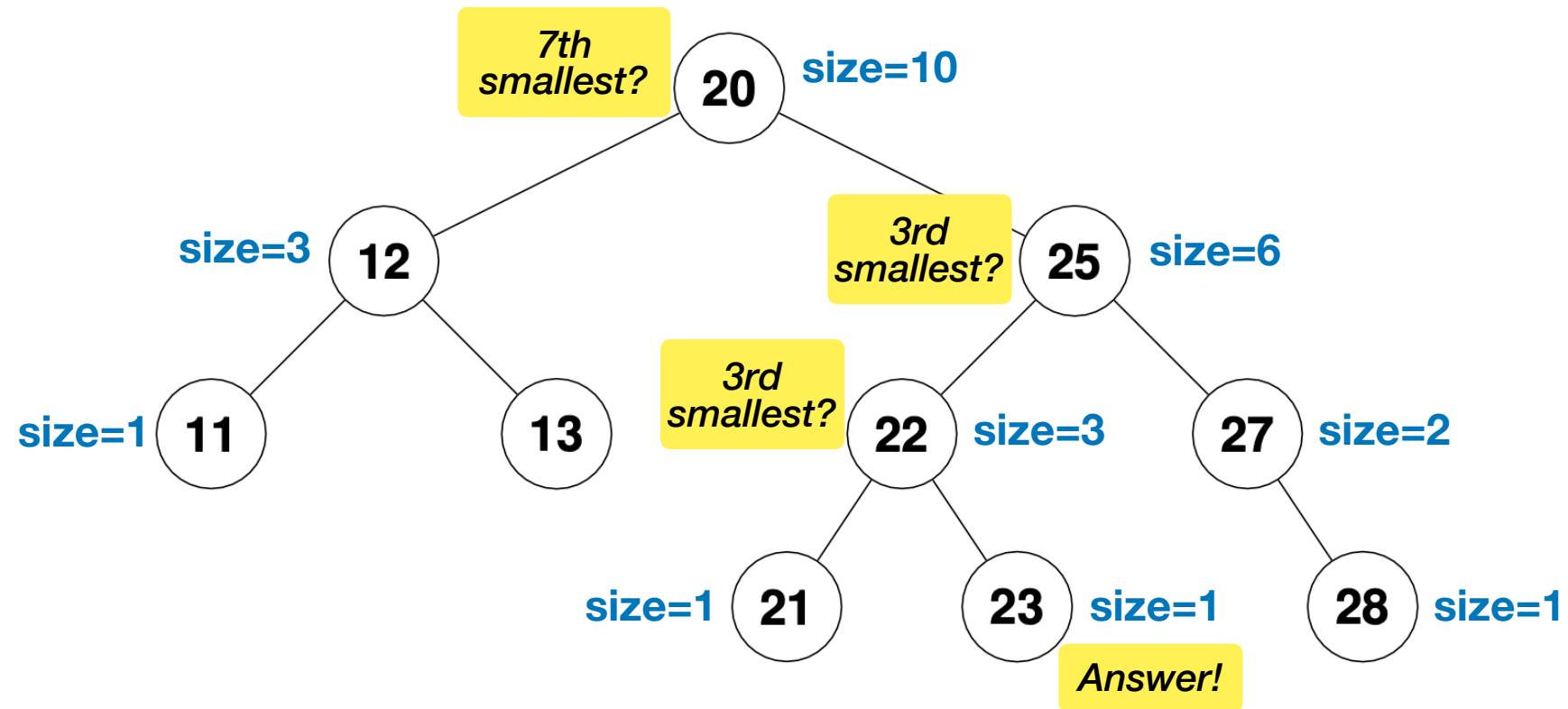
Trivial way: Perform in-order traversal and report 7th element.

Can we achieve $O(\log n)$ time?

(instead of $O(n)$)



Simpler Question: Find k -th smallest element in BST



Question: How can you find 7^{th} smallest element?

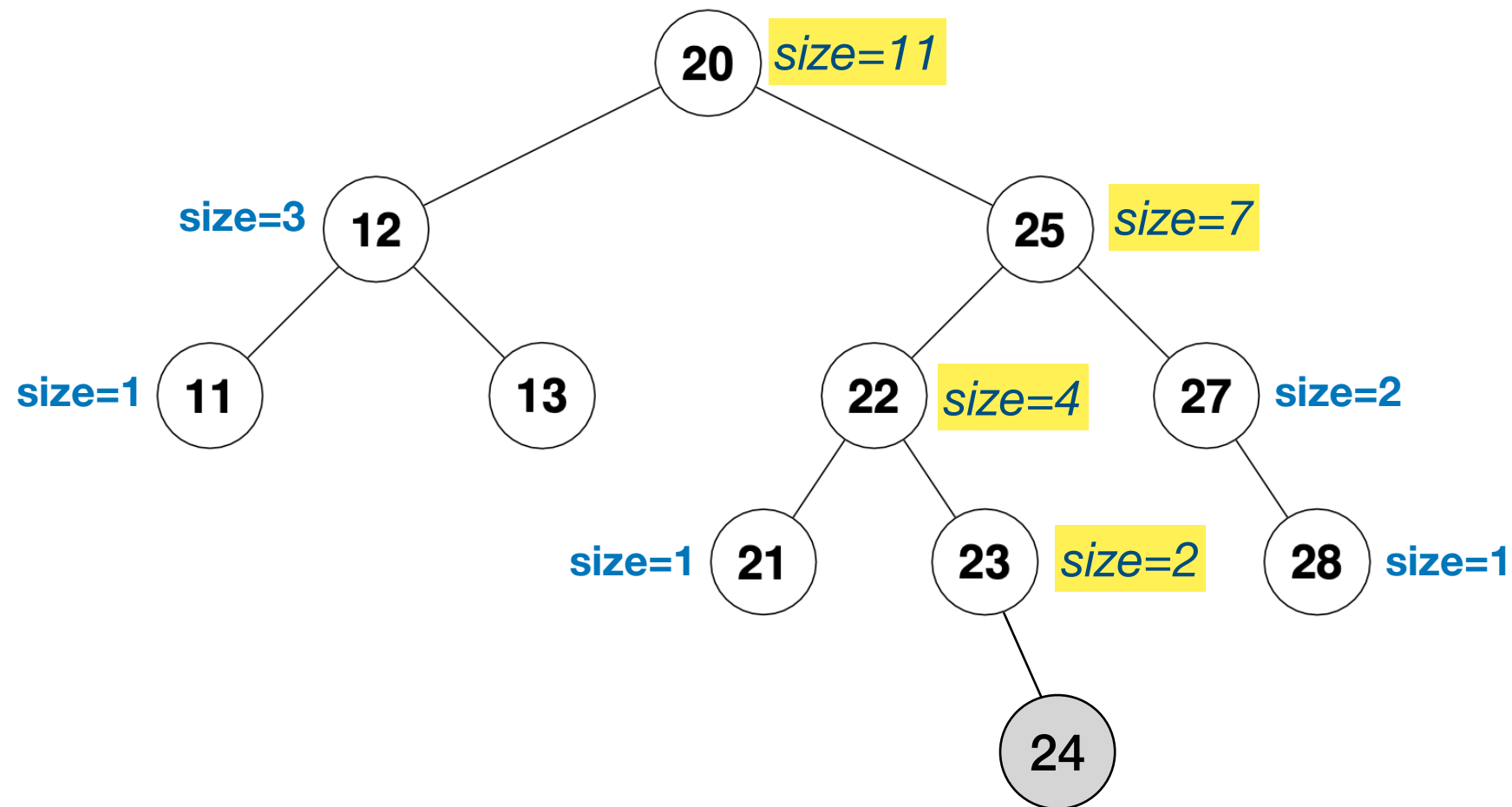
Improved solution:

Store in the variable “**size**” the number of nodes in subtree

*Time is
order of
height
of BST!*



Simpler Question: Find k -th smallest element in BST



Question: How can you update size after insertion of new element, say 24?

Answer: Increase $x.size$ by 1 for ALL nodes x lying on path from root to newly inserted leaf 24.

Second Question: Find k -th smallest element in AVL tree

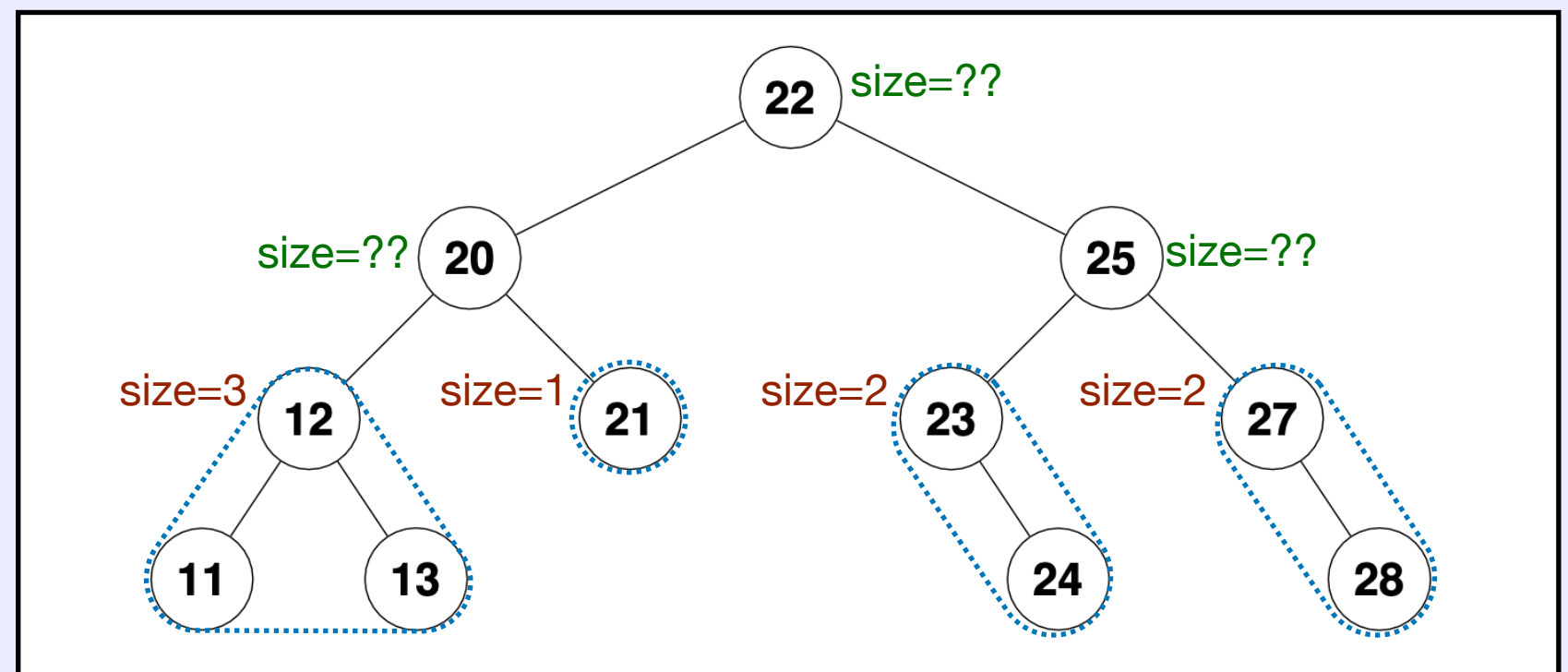
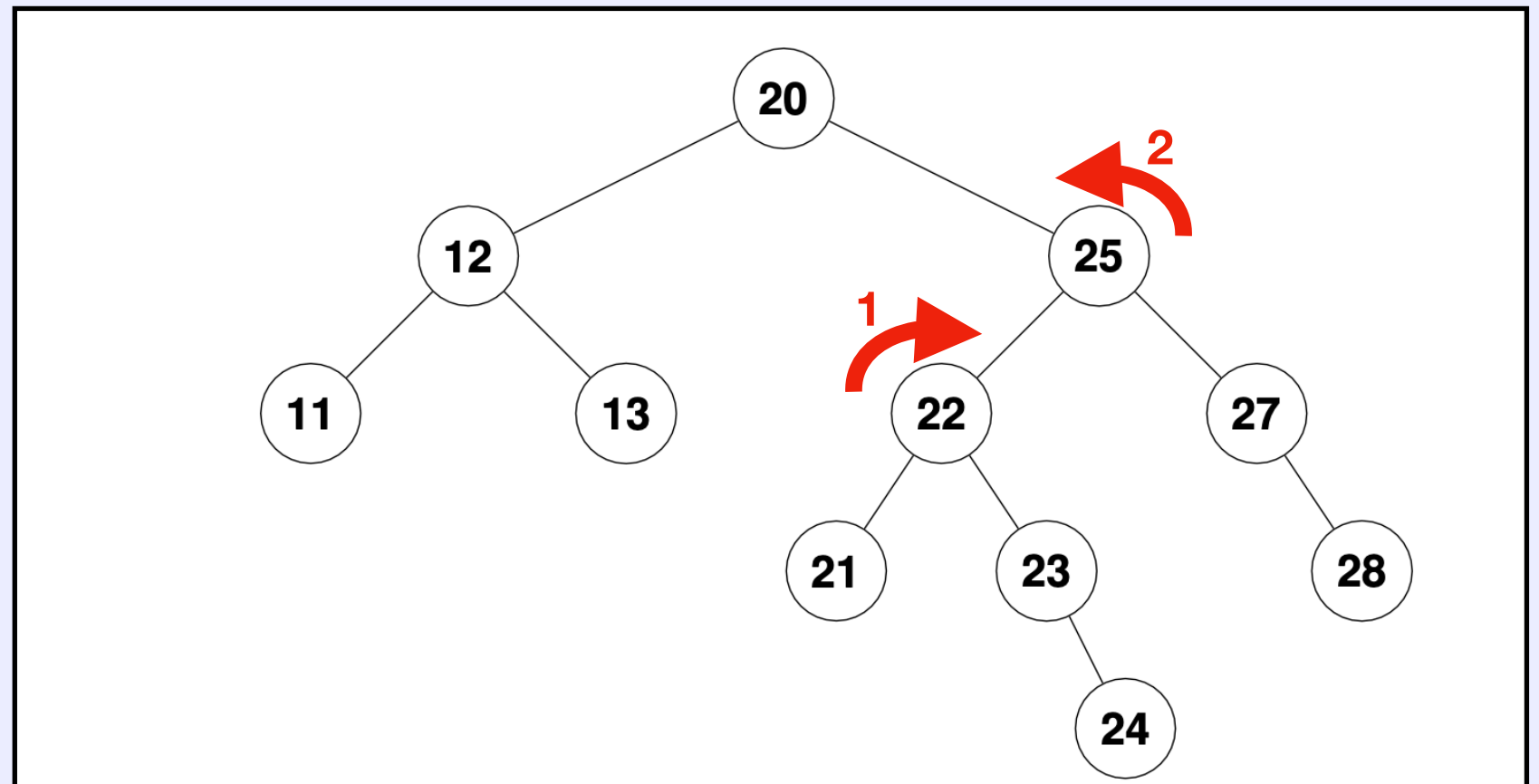
AVL tree
=
BST + rotations

Algorithm to update size:

1. Find all nodes undergoing rotations.
2. For each affected node x , update $x.size$ to the value $1 + x.left.size + x.right.size$.

Remark: Nodes must be processed in bottom-up manner.

Time to update size after single insertion/deletion is $O(\log n)$.



Main Question: Solve Banking System problem

					
Registration id:	11	22	33	44	55
Balance:	600 rs.	800 rs.	200 rs.	500 rs.	800 rs.

Goal: Design a data-structure with following tasks:

1. **Add new user** with registration number r and opening balance b .
2. **Increment balance** of user r by an amount $+\delta$.
3. **Return registration id** of user with k^{th} -highest balance.

Space used must be $O(n)$, and all updates/queries must be handled in $O(\log n)$ time.

Main Question: Solve Banking System problem



Registration id:

11

22

33

44

55

Balance:

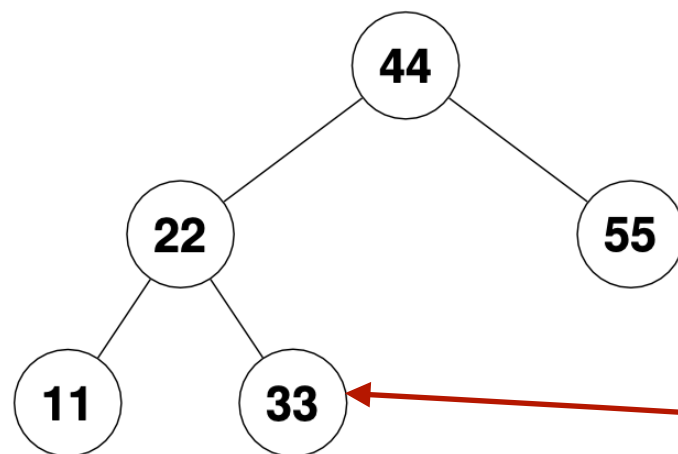
600 rs.

800 rs.

200 rs.

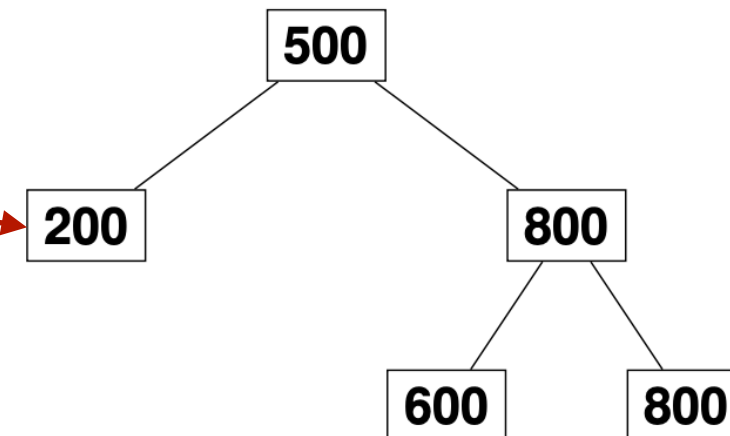
500 rs.

800 rs.



AVL tree 1 storing Registration Ids

*Pointers b/w
"r" and "b"*



(modified) AVL tree 2 storing Balances

Question 4: Coding Fest

Problem: Design an algorithm to return top k scorers in a coding fest.

Algorithm

1. Compute a max-heap that stores the pairs (s_i, i) . This takes $O(n)$ time.
2. Initialize A as an empty-set.
3. Repeat k times: Delete the max entry of the heap, and add the corresponding index to A .
4. Return set A .

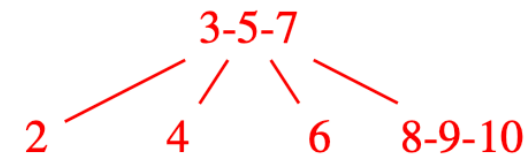
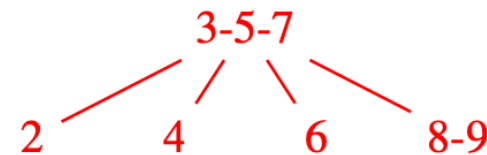
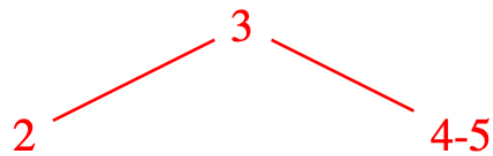
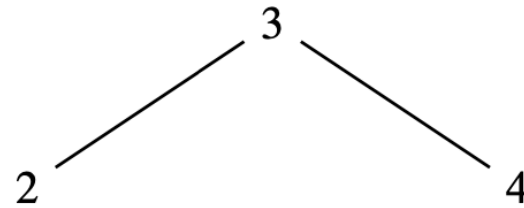
Question 3: Bipartite Graph

Problem: Show that a graph is bipartite iff it does not contain odd length cycles.

Solution: Covered in lectures

Question 2(e): 234 Tree

- (e) Consider the 234 tree below where each node is a 2-node. What is the minimum number of insertions needed to increase the height of the tree? Explain.



The answer is 7.

One can see that on insertion of keys 5-10, we get a root to leaf path in which all nodes are 4-node. Now on making next insertion to corresponding leaf node (say a key 11) the height of 234 tree will increase.

Question 2(d): Merging Balanced BSTs

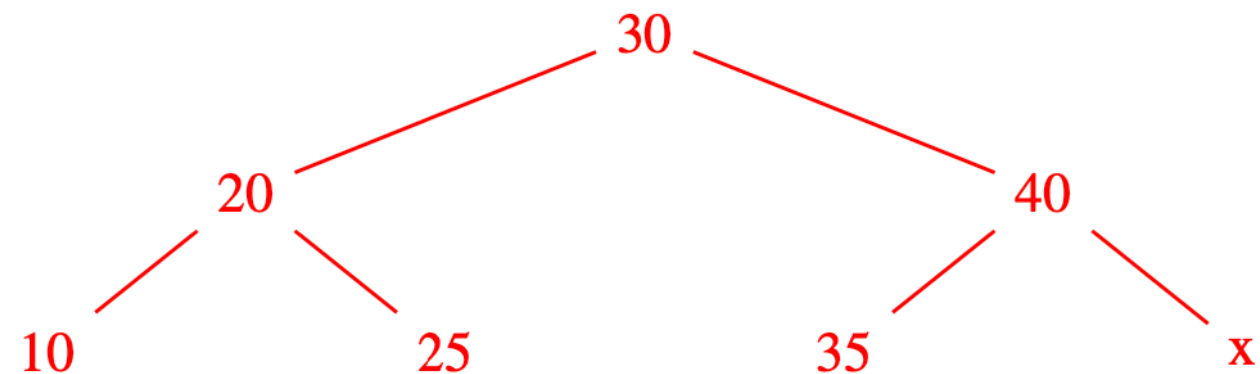
(d) Prove or disprove: Any algorithm to merge two balanced BSTs T_1 and T_2 of size n into a single **balanced** BST takes $\Omega(n \log n)$ time.

False. We can perform this task in linear time as follows:

- (a) Convert the BSTs into sorted arrays, say A_1 and A_2 , by doing inorder traversal.
- (b) Merge A_1 and A_2 in $O(n)$ time, and let B denote the resultant array.
- (c) Convert B to a balanced BST using solution of question 2(a).

Question 2(c): BST and Postorder Traversal

- (c) Suppose the preorder traversal of a BST visits the nodes in the order: 30 20 10 25 40 35. Draw the corresponding BST and compute its postorder traversal.



The postorder traversal visits the nodes in the order: 10 25 20 35 40 30.

Question 2(b): Connectivity

(b) Let $G = (V, E)$ be an undirected graph, and A be a function on $V \times V$ defined as below:

$$A(x, y) = \begin{cases} 1 & \text{if there is a path from } x \text{ to } y \text{ in } G; \\ 0 & \text{otherwise.} \end{cases}$$

Design an $O(|V| + |E|)$ time algorithm to compute $\sum_{x, y \in V} A(x, y)$.

Algorithm:

(a) Use BFS traversal to compute all the components of graph G , let these be $C_1, \dots, C_k$.

This step takes $O(m + n)$ time.

(b) Output $\sum_{i=1}^k |C_i| \times (|C_i|)$, where $|C_i|$ denotes the number of vertices in C_i .

Question 2(a): Sorted array to balanced BST

```
// @ param A:          sorted array
// @ param start, end:  starting and ending index of A
// @ return:           root of the balanced BST

BSTNode buildBST(Integer[] A, Integer start, Integer end) {

    if(start > end) return null;

    Integer mid = start + (end-start)/2 ;

    BSTNode root = new BSTNode();

    root.val = A[mid] ;

    root.left = buildBST(A, start , mid-1 );

    root.right = buildBST(A, mid+1 , end );

    return root;
}
```