Binary Search Tree (BST)
We have Studied about binary search on the array. For applying it own array should be sorted.

i/p + {10, 20, 30, 40, 50, 60} target → {10}

mid = 0 + 5 = 2

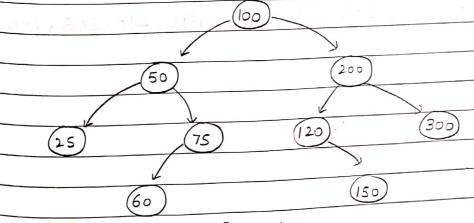
arr[mid] = = target (x) arr[mid] > target (v)

10,20

mid = 0 + 1 = 0

avr [mid] = = target ()

Time complexity = O(logn)



BST can be defined as root → data > (left subtree data) root → data < (right subtree data)



Vote →	In BST, if there are duplicate values then
	it would be specified in the questions.
	Excess for the Tree (BST)
301 /	100 > 50, 25, 75, 60 { Valid for root
201	100 < 200,120,300,150 node
	50 725
	50 < 75,60 00 408 2014 10 10 10 10 10 10 10 10 10 10 10 10 10
	25 } leaf nodes
	60
	200 > 150, 120 200 < 300
	120 < 150
	150 and 300 are leaf nodes.
	ser sole stay nodes.
Note -	The property which we discussed should
	De vocation of the hood of the
	we assume it to be true.
<i>€</i> -x->	Create a BST with following data.
	100.150.200.135.4
	ا رکورورورورورورورورورورورورورورورورورورو
	(100)
	(150)
	140 (200)
	175) (210)
	(160) (190) (205)
- 11	

Scarined with Cam



	Just insert data according to the definition
	of BST.
	the state of the s
Exca	Create BST with following data
	10,20,5,11,17,2,4,8,6,25,15
	(10)
	(5) (20)
	(2) (8) (1) (25)
	46 (17)
	(15)
	The above is how we create a binary search
	tree. Here we follow -1 as stopping criteria
	Just like binary trees.
	*OCCE
	Node * insert IntoBST (Node * root, int data)
	1
	// Ist node case (Empty tree)
	if (root == NULL) {
	root = new Node (data);
	return root j
	3
	l'insert Porto left
	If (root → data > data) root → left = insert IntoBST (root ~ left, data);
	root - left = 11150a = 1700 data);

Scarineu Willi Call



	Minsort into right
	else {
	root - right = insent Into BST (root - right
	3 - Adata phinalla Han (2 data);
	3 return rooti
	A STATE OF THE STA
	//called in main function
	void take Input (Node* & root) {
	int data;
	cin >> data
	while (data! = -1) {
	root = insert Into BST (root, data))
	cin>>data;
	3
	3
Vote -	The codes of income
3	The codes of inorder, preorder & postorder traversal is same as that of binary trees.
4:	as that of binary trees.
	(10)
ed obs	(5) (20)
	(2) (8) (1) (25)
	46
	T 1 : 2 / 5
	Inorder: 2 4 5 6 8 10 11 15 17 20 25
	Preorder: 10 5 2 4 8 6 20 11 17 15 25 Postorder: 4 2 6 8 5 15
	Postorder: 4 2 6 8 5 15 17 11 25 20 10
5 1 10 1	E. I.:

Scarnieu with Cam



Scarineu with Cari

Note 7	It is important to note inorder traversal of the
- Jane	Binary Search Tree is sorted.
	time of this table to be with a taken to be a
12.2	Searching in Binary Search Tree
V V	
	(5) (20)
	(2) (8) (1) (25)
	· (double of interference in the same in t
	(4) (6) (17) Target = 17
	(15)
	root → data = = tanget J found root → data > tanget J left
_ 2)	root - data > tanget] left
3)	root -data < target I right
	0 0
	10→20 → 11 → (17)
	4 found
	. Hat we prove of addition 40 con descriptions
	Time complexity in average case = O(h) Height of BST in average case is logn.
	Height of BST in average case is logn.
	January Company of the Company of th
	Height of BST in worst case is O(n)
	Height of BST in worst case is O(n) 4 Skewed tree
	(10)
	In Skewed BST
	(4) 1 2 (4) 1 2 (5) 1 (5
	ciondinate de la constante de
	(1) Annual Santa day and a saile day of the contract of the co

97	Code
	bool find Node (Node * root , int target) [
	// Base case
	If (root = = NULL)
	retwin false
	// Found
	If (root -data = = target)
	//Search in left
	else if (root + data > target)
\ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \	retwin find Node (root - left, target)
-	
	//Search in right
	else 1
	2 retwin find Node (root-right, target)
	2
	J
Vota	Have oxdess of x
STOLES	Here order of recursive calls is not of concern. It can be in any order.
	many order.
	Maximum & minimum in BST
*	Approach-1 => Find inorder and then 1st value
	Is minimum and last value is maximum.
*	Abbreach - 2 = Pin 10 0
	Approach - 2 = go to left until we get left node such that left node does not exist further to get minimum, value of
	further to get minimum does not exist
	go till right to get marine value. Similarly
	further to get minimum value. Similarly go till night to get maximum value.
	Scarneu Willi Ca

Scarineu With Call

Code int find Min (Node * root) { Node * temp = root; if (temp = = NULL) return - 1) While (temp - left (= NULL) temp = temp -left > retwin temp - datas int find Max (Node * root) { Node * Lemp = root i mor men if (temb = = NULL) return -1; while (temp + right != NULL) temp = temp -right return temp - datai Inorder predecsor/successor predecessor means inorder traversal mai root->left mai "MAX" value hota hai. successor means inorder traversal 20) mai root->right mai "MIN" value hota hai,

(17)



Inorder: 2 4 5 6 8 10 11 15 17
20 25
Bulling of the low would herit n
Inorder predeccesor of 11: 10
Inorder successor of 11: 15
Here we don't have to store the inorder
traversal.
Alate divascie proi
There are very high chance of getting a
There are very high chance of getting a question of variation of deletion in BST.
Inorder predeccesor means left subtree's
maximum value.
Inorder successor means right subtree's
minimum value
(JOUN = taring edmod) slide
But the above 2 Statements are not
always true. Like It does not have anything
in the left subtree but it has inorder
predecessor as 10. If we need to find for 11, then we need to stove the inorder traversal
then we need to stove the in the
and then we can find it.
GIVE TITLE
Deletion in BST
(10)
(5) (20)
(2) (8) (11) (25)
(2) (1) (25)
(4) (b) (15)
Scarnied with Ca

A STATE OF THE PARTY OF	Page
_	target = 25
	THE PART OF THE PA
_	Steb-I = Seauch for 25 in the BST
	Case-1
	Retwin NULL
	Market Harman X and Say X and January and American
	Case-2
	Return root right
į	Wine 13 y many X as so O Jalo on
	# HALL & STORY
	Case-3 Retwin root -left
	X
	\mathcal{O} \mathcal{X} \mathcal{I}
	Case-y
	Suppose we have to delete 5 in the BST.
- 1	THE TOTAL PROPERTY OF THE PROP

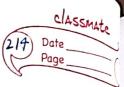
We have to delete in such a way that after deletion, tree is BST.

In this case instead of 5, place inorder predeccesor 2 delete that node. (inorder predecesor node)

Code

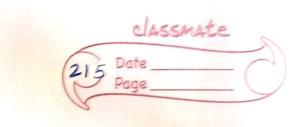
Node * delete Node In BST (Node * root, int target) {

Scarineu with Can1



```
if (700t = = NULL)
            return root j
   if (root - data = = target) {
      // Case 1: Leaf node
      if (root-left == NULL && root-right == NULL)
                    return NULL)
      // Case 2 : Left child not exist
     else if (root + left = = NULL 12 root + right != NULL
          Node * child = root -right;
         return child;
   // Case 3: Right child not exist
else if (root - left!= NULL & & root - right == NULL
           Node * child = root - left;
          return child;
  // Case 4: Both child exist
  else {
       int inoyder Pre = find Max (root -left);
       root - data = inorder Pre;
       root + left = delete Node InBST (root+left,
       inoyder Pre);
       return root
3 // Search in rught subtree
else if (root +data < target) {
     root - right = delete Node In BST (root - right)
    target);
```

Scarineu with cam



```
else { // search in left subtree

root +left = delete Node In BST (root+left,

target);

return root;
}
```

Scanned with Cam