Normal BST to Balanced BST

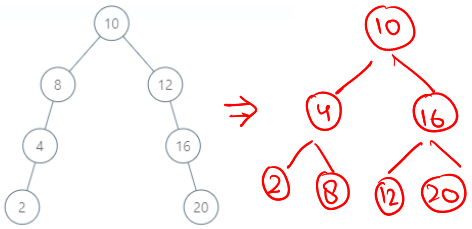
Problem statement Send feedback

You have been given a binary search tree of integers with 'N' nodes. Your task is to convert it into a balanced BST with the minimum height possible.
A binary search tree (BST) is a binary tree data structure that has the following properties.

- The left subtree of a node contains only nodes with data less than the node's data.
- The right subtree of a node contains only nodes with data greater than the node's data.
- Both the left and right subtrees must also be binary search trees.

A Balanced BST is defined as a BST, in which the height of two subtrees of every node differs no more than 1.
For Example:

For the given BST:



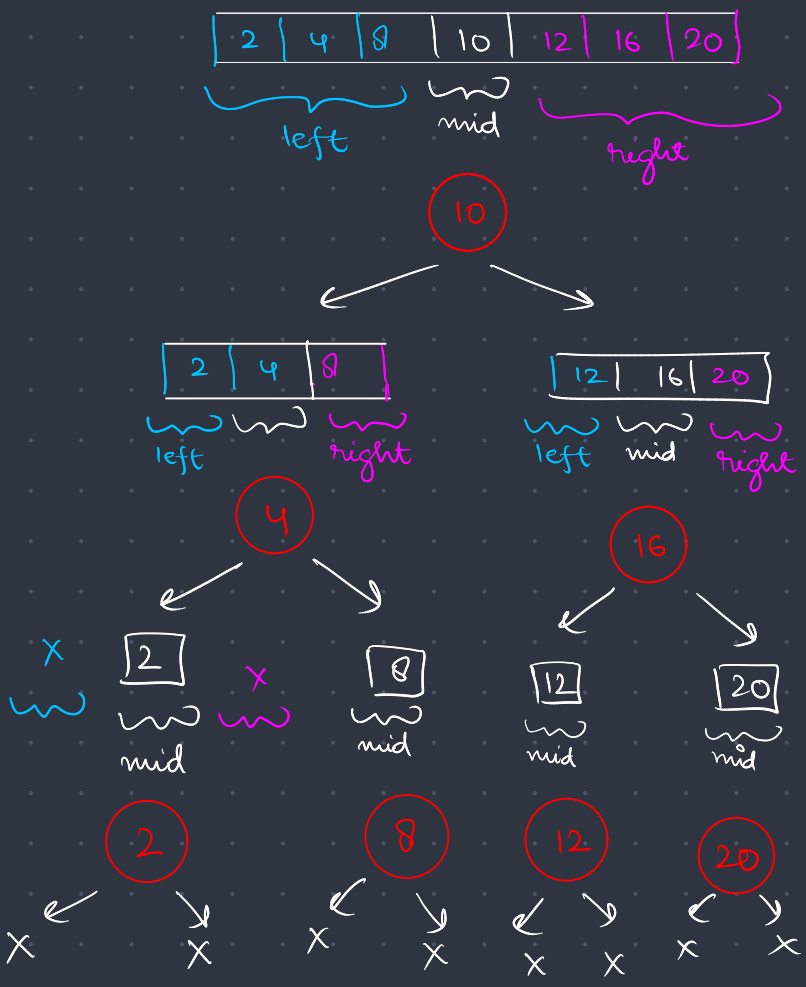
→ Approach :-

↳ First get the preorder traversal of the BST and store it inside array.

↳ Use mid index of the array.

↳ Make it root node

↳ The elements left to it will be root → left and the elements right to it will be contained inside root → right.



```
void inorder(TreeNode<int> *root, vector<int> &ino) {
  if (!root)
    return;

  inorder(root->left, ino);
  ino.push_back(root->data);
  inorder(root->right, ino);
}

TreeNode<int> *balance(vector<int> ino, int s, int e) {
  if (s > e)
    return NULL;
  int mid = (e - s) / 2 + s;

  TreeNode<int> *node = new TreeNode<int>(ino[mid]);

  TreeNode<int> *leftNode = balance(ino, s, mid - 1);
  TreeNode<int> *rightNode = balance(ino, mid + 1, e);

  node->left = leftNode;
  node->right = rightNode;

  return node;
}

TreeNode<int> *balancedBst(TreeNode<int> *root) {
  vector<int> ino;
  inorder(root, ino);
  int e = ino.size() - 1;
  int s = 0;

  return balance(ino, s, e);
}
```

Time Complexity : $O(N)$

Space Complexity : $O(N)$