# Array to Binary Search Trees Quick

Asked 6 years ago Modified 2 years, 6 months ago Viewed 26k times

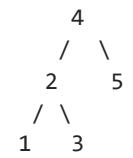
10

2

Given an array of integers, is there a way to convert it into Binary Search Tree (unbalanced) quickly? I have tried inserting it one by one for each element, but this means that I have to traverse from the beginning for each insertion. It works perfectly, but I think the worst case is O(N^2) for being unbalanced, e.g. the array is sorted. Given a large N, I think it is going to take some time.

Back to my question, is there a way to do this faster than the algorithm that I stated?

For example, given array [4,5,2,3,1], is there a fast way to create this?




algorithm binary-search-tree

Share Edit Follow

edited Apr 23, 2016 at 6:34

asked Apr 23, 2016 at 5:16

jamesalone
**291** 1 5 18

## 4 Answers

Sorted by: Highest score (default)

Yes, there is easy way to construct a balanced Binary Search Tree from array of integers in O(nlogn).

Algorithms is as follows:

5

Algorithms is as follows:

1. Sort the array of integers. This takes O(nlog(n)) time
2. Construct a BST from sorted array in O(n) time. Just keep making the middle element of array as root and perform this operation recursively on left+right half of array.

**EDIT:**

1. Firstly, you can't do this better than O(nlog(n)) because then you would have essentially sorted the unsorted array (using comparison) in complexity better than O(nlogn). This is impossible.
2. If you don't have to do sorting initially, you could construct binary tree by using binary tree insertion algorithm for each element of the array.

Refer to any standard implementation of Self-balancing BST. While scanning the array, at ith iteration, you have BST for arr[1...i]. Now, you add arr[i+1] to BST (using insertion algorithm).

Share Edit Follow

edited Apr 23, 2016 at 6:50

answered Apr 23, 2016 at 5:44

Rishit Sanmukhani
**2,099** 13 25

Okay, I get the idea. Just wondering though, does this mean there is no faster way to do it without sorting? I've updated the question, to show some example of what I mean. – jamesalone Apr 23, 2016 at 6:32

5

Already a good explanations is available. Below is the code for constructing BST from a given Array.

```
public static void main(String args[])
{
    Node root=null;
    int arr[]=new int[]{99,35,19,0,11,40,5};
    int length=arr.length;
    Arrays.sort(arr);
    root=constructBST(arr,0,length-1,root);
}
```

```
}
public static Node constructBST(int[]arr,int start,int end,Node root)
{
    if(start>end)
        return null;
    int mid=(start+end)/2;

    if(root==null)
        root=new Node(arr[mid]);

    root.left=constructBST(arr,start,mid-1, root.left);
    root.right=constructBST(arr,mid+1,end, root.right);

    return root;

}
```

After this just do inorder traverse to pri

Share Edit Follow

answered May 16, 2018 at 6:35

Abhay
**129** 1 5

I dont think that we need to pass a Node to the constructBST function. It still works without node. – Toan Tran Mar 15, 2020 at 15:07

3

Given an array of integers, is there a way to convert it into Binary Search Tree (unbalanced) quickly?

Sure. Sort the array in O(n logn), select the middle element of the array to be the root and insert all element before the middle element in the left to the root and those after the middle in the right (O(n) time). The total complexity is `O(n logn)`. For example, if you have the array:

```
3, 5, 2, 1, 4
```

you sort it to `1, 2, 3, 4, 5`. The middle element is 3 so you create the tree

```
      3
     / \
    2   4
   /     \
  1       5
```

You could have two pointers to the middle element and you start to move the first to the left and the other to the right, and just insert the elements pointed by the pointers to the left and right subtree respectively.

The problem is that the height of the tree is `n/2` which mean that the search operation is `O(n)` which is slow. To get better performance you should use self-balancing binary search tree instead, like red-black tree or AVL tree

Share Edit Follow

edited Apr 23, 2016 at 5:48

answered Apr 23, 2016 at 5:43

sve
**4,236** 1 18 29

May be use some radix sort since it is array of integers? – Giorgi Nakeuri Apr 23, 2016 at 5:52

0

Okay, I am not sure how optimal this solution is but here is what I wrote

## Algo

1. sort the input array
2. create a recursive function which takes that sorted array
3. Inside the recurisve function if the length of arr is one or if it is an empty array, return arr
4. Else calculate the mid point `(parseInt(arr.length/2))`
5. return an array `[midpoint, recuriveFunc(arr[leftToMidPoint]),recuriveFunc(arr[rightToMidpoint])]`

## Code implementation in Javascript

```
const arr =  [1,2,3,4,5,6,7]


const recursiveSetValues = (arr) => {
  if (arr.length < 2) return arr
  const midPoint = parseInt(arr.length/2)
  return [arr[midPoint], ...recursiveSetValues(arr.slice(0, midPoint)),
...recursiveSetValues(arr.slice(midPoint+1, arr.length))]
}


const sortArray = arr.sort((a,b) => a-b)
console.log(recursiveSetValues(sortArray))
```

Run code snippet | Copy snippet to answer | Expand snippet

Share Edit Follow

answered Nov 4, 2019 at 1:00

iRohitBhatia

**3,019** 10 48 100