**Summary:**

The same data can be represented using different binary search trees. Some binary search trees may be closer to a linked list and some closer to a perfect tree. Any type of binary search tree can display the information in ascending order when traversing in in-order format. A binary search tree can be placed into an array using the formulas listed below (starting from index 0 or index 1). Such an array would not be in sorted order. Using the same formula, the unsorted array can be converted back to a binary search tree.

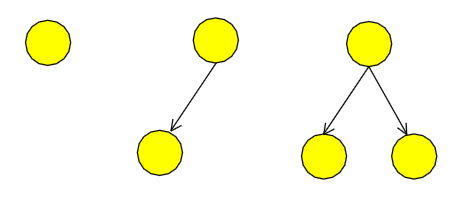
To convert a sorted array into a binary search tree, the formula below cannot be used. We have to use recursion instead.

**Converting a tree into an array which will be unsorted**

How can we represent an arbitrary binary tree in an array?

If we use an array to implement a tree, we have to set a limit on the number of nodes we will permit in the tree. Our strategy is to find the maximum height of the tree (H), and make the array big enough to hold any binary tree of this height (or less). We'll need an array of size (2^H)-1.

Every node has at most two children. Given N nodes, what is the depth of a binary tree?



Depth 0: N=1, N=(2^(0+1))-1

Depth 1: N=3, (2^(1+1)) -1

Depth d: N=(2^(d+1))-1 (Perfect tree)

Two options:

Option 1: Start with index 0

Children of a[i]

Left=a[2i+1]

Right=a[2i+2]

Option 2: Start with index 1

Children of a[i]

Left=a[2i]

Right=a[2i+1]

Example: Depth=2, Number of nodes = 2^(2+1)-1=7

7

3 20

9 | 22

The empty slots could be filled with some number like -999 to indicate there is nothing

Start at 0

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| I=0  2I+1=1 L  2I+2=2 R |  | I=2  2I+1=5 L  2I+2=6 R |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
| 7 | 3 | 20 |  |  | 9 | 22 |

Start at 1

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | I=1  2I=2 L  2I+1=3 R |  | I=3  2I=6 L  2I+1=7 R |  |  |  |  |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|  | 7 | 3 | 20 |  |  | 9 | 22 |

How do you convert an unordered array (could be based on BST) into a tree?

**Converting a sorted array into a binary tree.**

Pseudo-code: recursive

Step 1. Create a function that takes in the integer array, an integer start, and an integer end. Start = 0, end = integer array size - 1.

Step 2. Create an integer middle which equals (start + end)/2.

Step 3. Test to see if start > end.

Step 4. If so, return null.

Step 5. Else, make the value at the middle index the root of your tree.

Step 6. Set the left node of the root equal to the function with (array, start, middle - 1).

Step 7. Set the right node of the root equal to the function with (array, middle + 1, end).

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Index** | **0** | **1** | **2** | **3** | **4** | **5** |
| a[] | 1 | 2 | 3 | 4 | 5 | 6 |

int arr[] = {1, 2, 3, 4, 5, 6};

|  |  |  |  |
| --- | --- | --- | --- |
| 1st insertion  **1,2,3,4,5,6**  S(a,0,5)  mid=2  root=Node(2)**3**  left=S(a,0,1)  right=S(a,3,5) | 2nd insertion  **1,2**  S(a,0,1)  Mid=0  root=node(0) 1  left=S(a,0,-1)  right=S(a,1,1)  return 1 | **Start>end**  S(a,0,-1)  Return null | |
| 3rd insertion  **2**  S(a,1,1)  Mid=1  root=node(1) 2  left=S(a,1,0)  right=S(a,2,1)  return 2 | **Start>end**  S(a,1,0)  Return null |
| **Start>end**  S(a,2,1)  Return null |
| 4th insertion  **4,5,6**  S(a,3,5)  mid=4  root=Node(4)**5**  left=S(a,3,3)  right=S(a,5,5)  return 5 | 5th insertion  **4**  S(a,3,3)  Mid=3  root=node(3) 4  left=S(a,3,2)  right=S(a,4,3)  return 4 | **Start>end**  S(a,3,2)  Return null |
| **Start>end**  S(a,4,3)  return |
| 6th insertion  **6**  S(a,5,5)  Mid=5  root=node(3) 6  left=S(a,5,4)  right=S(a,6,5)  return 6 | **Start>end**  S(a,5,4)  Return null |
| **Start>end**  S(a,6,5)  Return null |

3

1 5

2 4 6