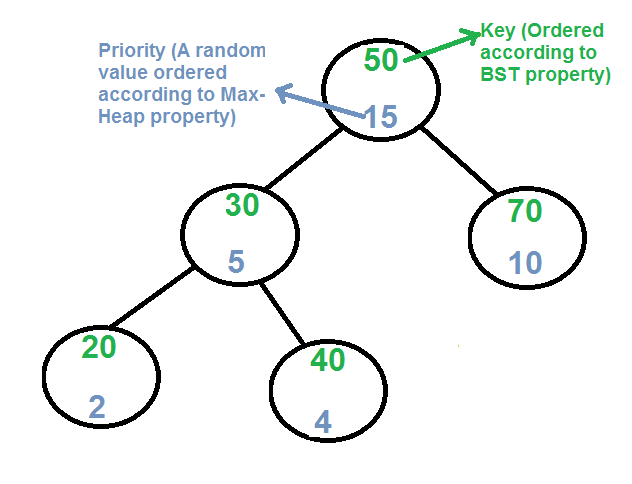
# Treap (A Randomized Binary Search Tree)

Like [Red-Black](http://www.geeksforgeeks.org/red-black-tree-set-1-introduction-2/) and [AVL](http://www.geeksforgeeks.org/avl-tree-set-1-insertion/) Trees, Treap is a Balanced Binary Search Tree, but not guaranteed to have height as O(Log n). The idea is to use Randomization and Binary Heap property to maintain balance with high probability. The expected time complexity of search, insert and delete is O(Log n).

[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/treap.png)

Every node of Treap maintains two values.  
1) **Key** Follows standard BST ordering (left is smaller and right is greater)  
2) **Priority** Randomly assigned value that follows Max-Heap property.

**Basic Operation on Treap:**  
Like other self-balancing Binary Search Trees, Treap uses rotations to maintain Max-Heap property during insertion and deletion.

T1, T2 and T3 are subtrees of the tree rooted with y (on left side)

or x (on right side)

y x

/ \ Right Rotation / \

x T3 – – – – – – – > T1 y

/ \ < - - - - - - - / \

T1 T2 Left Rotation T2 T3

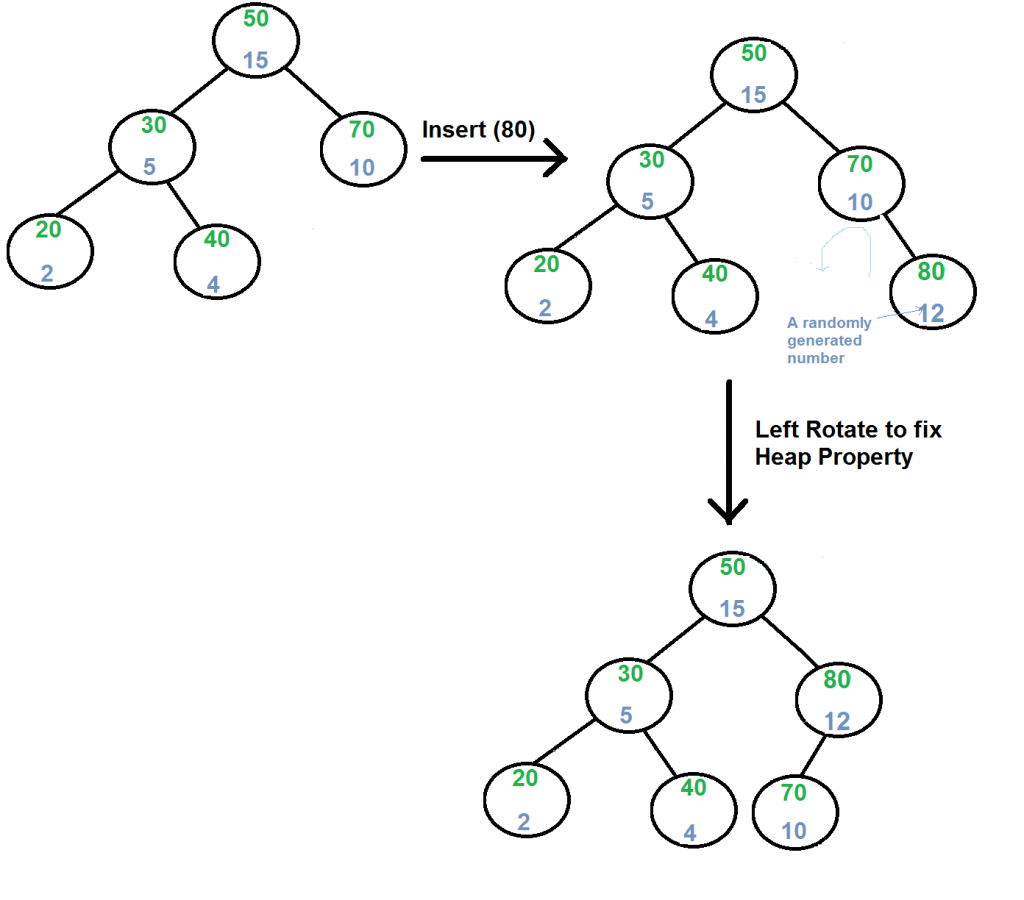
Keys in both of the above trees follow the following order

keys(T1) < key(x) < keys(T2) < key(y) < keys(T3)

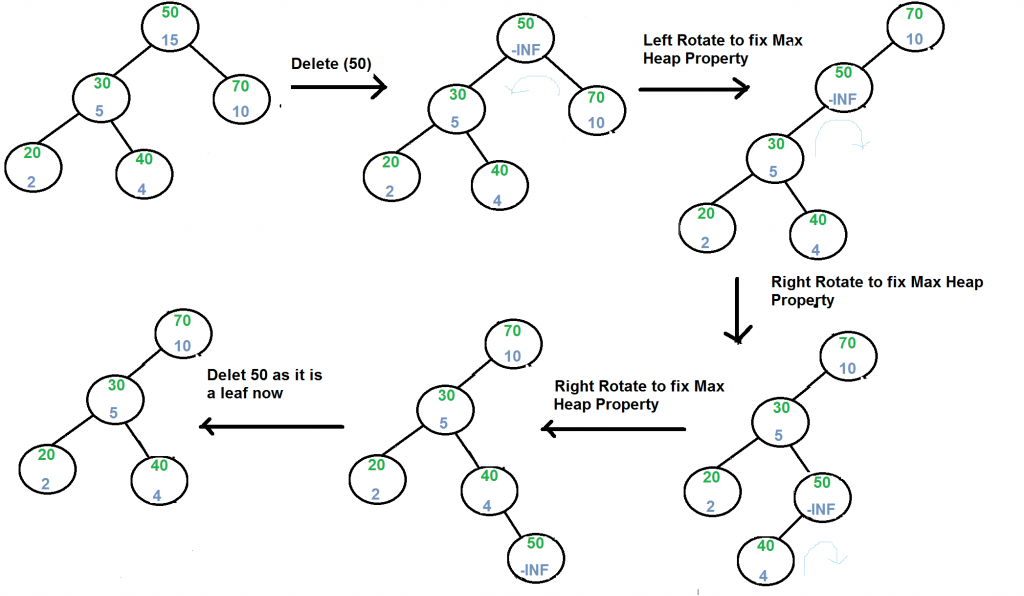
So BST property is not violated anywhere.

**search(x)**  
Perform standard [BST Search](http://geeksquiz.com/binary-search-tree-set-1-search-and-insertion/) to find x.

**Insert(x):**  
1) Create new node with key equals to x and value equals to a random value.  
2) Perform standard [BST insert](http://geeksquiz.com/binary-search-tree-set-1-search-and-insertion/).  
3) Use rotations to make sure that inserted node's priority follows max heap property.

[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/treapInsert1.png)

**Delete(x):**  
1) If node to be deleted is a leaf, delete it.  
2) Else replace node's priority with minus infinite ( -INF ), and do appropriate rotations to bring the node down to a leaf.

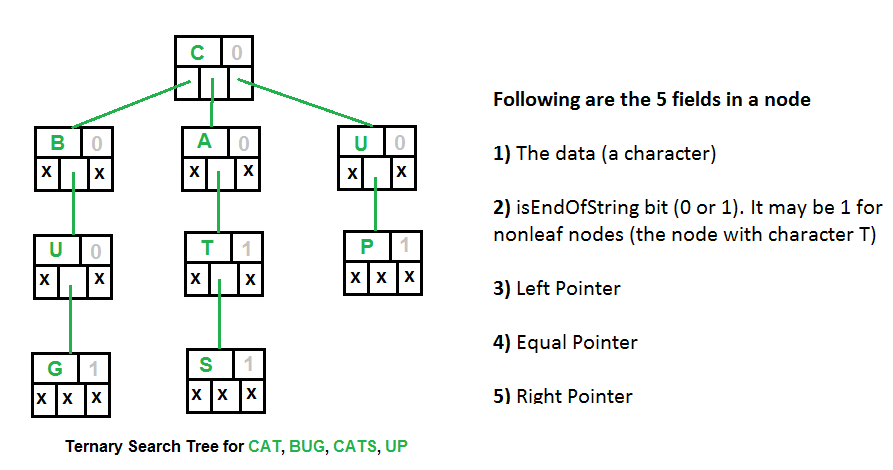
[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/treapDelete.png)

Refer [Implementation of Treap Search, Insert and Delete](http://www.geeksforgeeks.org/treap-set-2-implementation-of-search-insert-and-delete/) for more details.

# Ternary Search Tree

A ternary search tree is a special trie data structure where the child nodes of a standard trie are ordered as a binary search tree.

**Representation of ternary search trees:**  
Unlike trie(standard) data structure where each node contains 26 pointers for its children, each node in a ternary search tree contains only 3 pointers:  
1. The left pointer points to the node whose value is less than the value in the current node.  
2. The equal pointer points to the node whose value is equal to the value in the current node.  
3. The right pointer points to the node whose value is greater than the value in the current node.

Apart from above three pointers, each node has a field to indicate data(character in case of dictionary) and another field to mark end of a string.  
So, more or less it is similar to BST which stores data based on some order. However, data in a ternary search tree is distributed over the nodes. e.g. It needs 4 nodes to store the word “Geek”.  
Below figure shows how exactly the words in a ternary search tree are stored?  
[](http://d1hyf4ir1gqw6c.cloudfront.net/wp-content/uploads/Ternary-Search-Tree.png)

One of the advantage of using ternary search trees over tries is that ternary search trees are a more space efficient (involve only three pointers per node as compared to 26 in standard tries). Further, ternary search trees can be used any time a hashtable would be used to store strings.

Tries are suitable when there is a proper distribution of words over the alphabets so that spaces are utilized most efficiently. Otherwise ternary search trees are better. Ternary search trees are efficient to use(in terms of space) when the strings to be stored share a common prefix.

**Applications of ternary search trees:**  
**1.**Ternary search trees are efficient for queries like “Given a word, find the next word in dictionary(near-neighbor lookups)” or “Find all telephone numbers starting with 9342 or “typing few starting characters in a web browser displays all website names with this prefix”(Auto complete feature)”.

**2.**Used in spell checks: Ternary search trees can be used as a dictionary to store all the words. Once the word is typed in an editor, the word can be parallely searched in the ternary search tree to check for correct spelling.