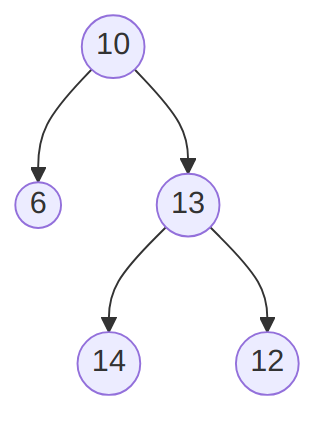
2025-09-30

# Trees

[Solutions for those exercises.](https:/princomp.github.io/solutions/data/trees)

## Exercises

1. Consider the following tree:

* 
* A binary tree that is not a binary search tree. ([text version](https:/princomp.github.io/diag/gra/bstree_example_4.txt), [image version](https:/princomp.github.io/diag/gra/bstree_example_4.png), [svg version](https:/princomp.github.io/diag/gra/bstree_example_4.svg))
  1. Explain why it is **not** a binary search tree.
  2. Pick one among *inorder*, *preorder* and *postorder* traversal, and give
     1. A brief description of how it proceeds,
     2. What it would produce for the given tree.

1. Consider the following implementation of “random” binary trees:

* public class RBTree<T>  
  {  
    
  private class Node  
   {  
   public T Data { get; set; }  
   public Node left;  
   public Node right;  
   public Node(  
   T dataP = default(T),  
   Node leftP = null,  
   Node rightP = null  
   )  
   {  
   Data = dataP;  
   left = leftP;  
   right = rightP;  
   }  
   }  
    
  private Node root;  
    
  public RBTree()  
   {  
   root = null;  
   }  
    
  public void Insert(T dataP)  
   {  
   root = Insert(dataP, root);  
   }  
    
  private Node Insert(T dataP, Node nodeP)  
   {  
   if (nodeP == null)  
   {  
   return new Node(dataP, null, null);  
   }  
   else  
   {  
   Random gen = new Random();  
   if(gen.NextDouble() > 0.5)  
   {  
   nodeP.left = Insert(dataP, nodeP.left);  
   }  
   else  
   {  
   nodeP.right = Insert(dataP, nodeP.right);  
   }  
   }  
   return nodeP;  
   }  
  }
* Note that the Insert(T dataP, Node nodeP) method uses the gen.NextDouble() > 0.5 test that will be randomly true half of the time, and false the other half.
  1. Explain the T dataP = default(T) part of the Node constructor.
  2. Write a ToString method for the Node class, remembering that only a node Data needs to be part of the string returned.
  3. Write a series of statements that would
     1. create a RBTree object,
     2. insert the values 1, 2, 3, and 4 in it (in this order).
  4. Make a drawing of a possible RBTree obtained by executing your code.
  5. Write a Find method that takes one argument dataP of type T and returns true if dataP is in the RBtree calling object, false otherwise.