Some properties of binary trees:

Property 1: The maximum number of nodes on any level i is 2^i where $i \geq 0$.

Property 2: The maximum number of nodes possible in a binary tree of height h is $2^h - 1$

Property 3: The minimum number of nodes possible in a binary tree of height h is equal to h.

Property 4: If a binary tree contains n nodes, then its maximum height possible is n and minimum height possible is $\lceil \log_2(n+1) \rceil$.

Property 5: In a non empty binary tree, if n is the total number of nodes and e is the total number of edges then $e = n-1$.

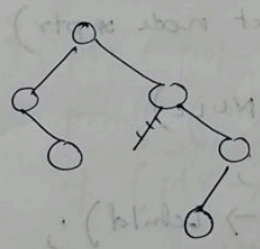
Property 6: For any non-empty binary tree, if n_0 is the number of nodes with no child and n_2 is the number of nodes with two children, then $n_0 = n_2 + 1$.

Property 7: A strictly binary tree with n nonleaf nodes has n+1 leaf nodes.

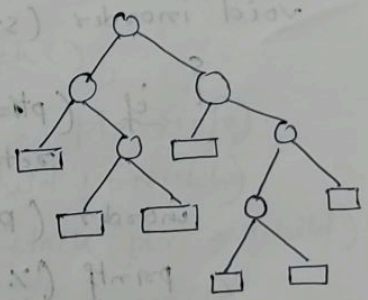
Property 8: A strictly binary tree with n leaf nodes always has 2n-1 nodes.

Property 9: In an extended binary tree, if E is the external path length, I is the internal path length and n is the number of internal nodes, then

$E = I + 2n$



Binary tree.



Extended Binary Tr

Property 10: If height of a complete binary tree is h $h \geq 1$, then the minimum number of nodes possible is 2^{h-1} and maximum number of nodes possib is $2^h - 1$.

Recursive functions for three traversals:

```
void preorder (struct node *ptr)
{ if (ptr == NULL)
        return;
  printf ("%d", ptr -> info);
  preorder (ptr -> lchild);
  preorder (ptr -> rchild);
}

void inorder (struct node *ptr)
{ if (ptr == NULL)
        return;
  inorder (ptr -> lchild);
  printf ("%d", ptr -> info);
  inorder (ptr -> rchild);
}

void postorder (struct node *ptr)
{ if (ptr == NULL)
        return;
  postorder (ptr -> lchild);
  postorder (ptr -> rchild);
  printf ("%d", ptr -> info);
}
```

9mp

Non-recursive function of Preorder traversal

```
void nrec-pre (struct node * root)
{ struct node * ptr = root;
  if (ptr == NULL)
    { printf ("Tree is empty \n");
      return;
    }
  push_stack (ptr);
  while (! stack_empty ())
    { ptr = pop_stack ();
      printf ("%d", ptr->info);
      if (ptr -> rchild != NULL)
          push_stack (ptr -> rchild);
      if (ptr -> lchild != NULL)
          push_stack (ptr -> lchild);
    }
  printf ("\n");
}
```

3rd The non-recursive function of Inorder traversal :

```
void nrec_in (struct node * root)
{
    struct node *ptr = root;
    if (ptr == NULL)
    { printf ("Tree is empty \n");
      return;
    }
    while (1)
    {   while (ptr -> lchild != NULL)
        { push_stack (ptr);
          ptr = ptr -> lchild;
        }
        while (ptr -> rchild = NULL)
        { printf ("%d", ptr -> info);
          if (stack_empty ())
              return;
          ptr = pop_stack ();
        }
        printf ("%d", ptr -> info);
        ptr = ptr -> rchild;
    }
    printf ("\n");
}
```

The non-recursive function of Postorder Traversal:

```
void nrec_post (struct node *root)
{
  struct node *q, *ptr = root;
  if (ptr == NULL)
  {
    printf ("Tree is empty\n");
    return;
  }
  q = root;
  while (1)
  { while (ptr->lchild != NULL)
    {
      push_stack (ptr);
      ptr = ptr->lchild;
    }
    while (ptr->rchild == NULL || ptr->rchild
                                  == q)
    {
      printf ("%d", ptr->info);
      q = ptr;
      if (stack_empty ())
        return;
      ptr = pop_stack ();
    }
    push_stack (ptr);
    ptr = ptr->rchild;
  }
  printf ("\n");
}
```