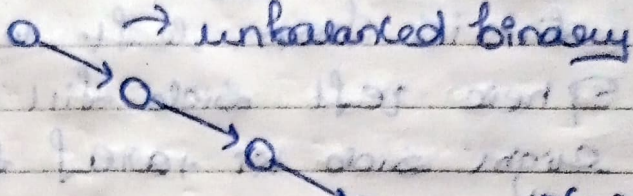
Trees (& k)
memoization DP

→ In tree add, remove, find all are happens in O(log N) times.

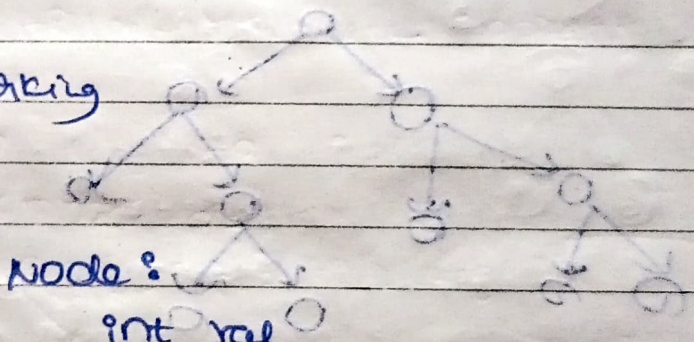
Limitation



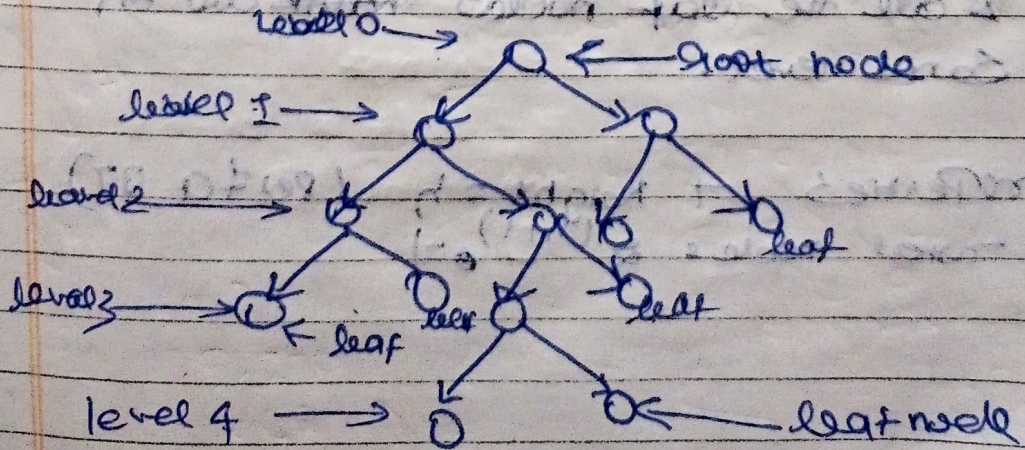
or
inefficient for ~~bottleneck~~ sorted data & find this

→ where is it used?
- file system
- Databases
- algo/Networking

for BT →



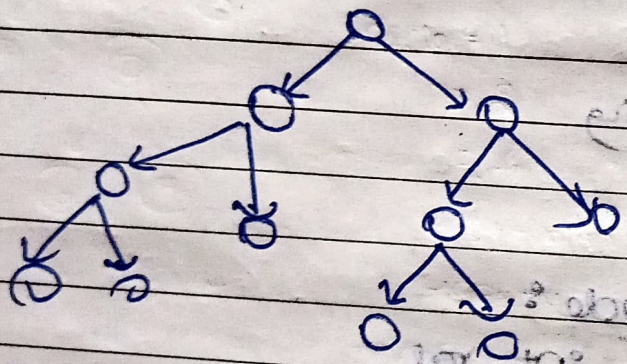
Node:
int val
Node left
Node right



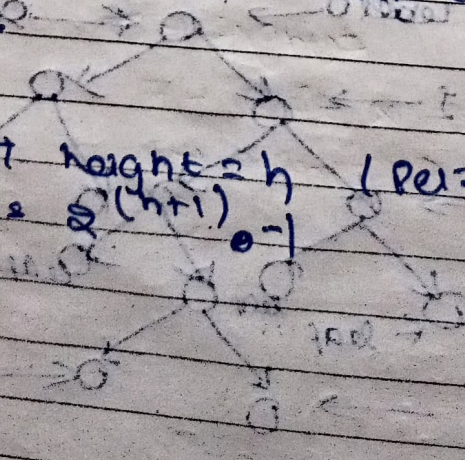
Type of binary tree

1) complete binary tree
→ all levels are full if last level are not full they should fill what fill is from left to right
* Phele left side fill hone chiye tab right side ki taraf badhna.

2) Full binary tree
→ each node have either 0 or 2 children



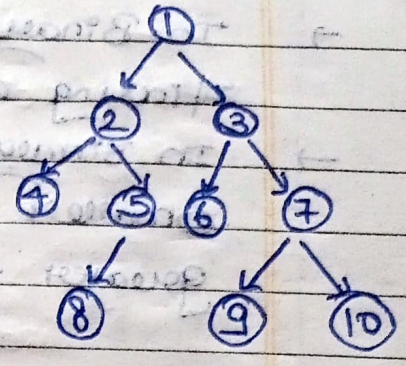
3) Perfect Binary Tree
→ all the internal node having two child & all the leaf nodes are on same level.



↑ Properties → height = h (perfect BT)
total node = $2^{(h+1)} - 1$

→ Inorder Traversal (left root right)

• 4 2 8 5 1 6 3 9 7 10



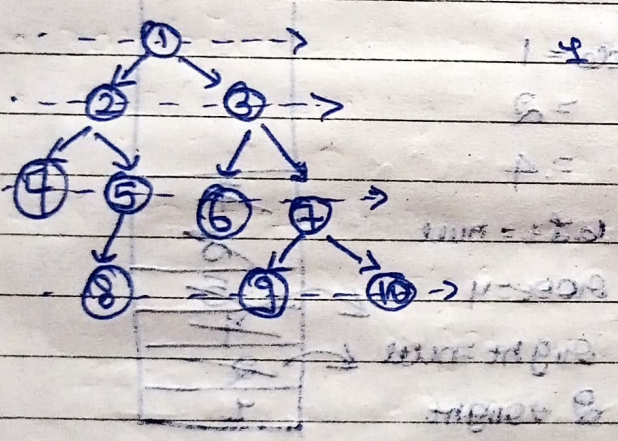
→ PreOrder Traversal (root left right)

1 2 4 5 8 3 6 7 9 10

→ Post order Traversal (left right root)

4 8 5 2 6 9 10 7 3 1

BFS (Traverse level wise)



1 2 3 4 5 6 7 8 9 10

inorder

```
void in(node){
  if(node==null) return;
  in(node->left)
  cout<< node->data<<endl;
  in(node->right)
}
```

Preorder

```
void Pree(node){
  if(node==null) return;
  cout<<node->data<<endl
  Pree(node->left)
  pree(node->right)
```

Post order

```
void post(node){
  if(node==null) return;
  post(node->left)
  post(node->right)
  cout<<node->data
```

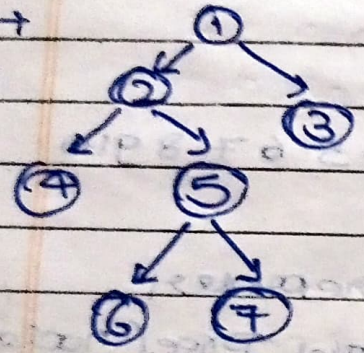
Binary Search Tree

→ height difference b/w two leaf node is not more than 1.

→ In Binary Search Tree insertion, deletion, finding all are happen in O(log N) times.

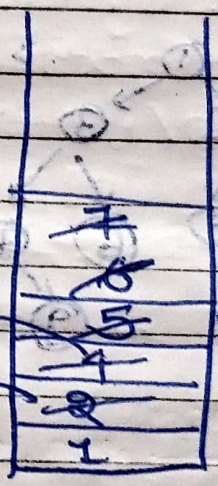
→ In Binary Search Tree left side is smaller than root node & right side is greater than root node.

Inorder Iterative



Left root right

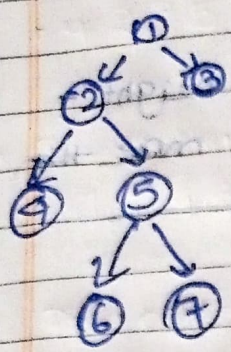
node = 1
= 2
= 4
left = null
root = 4
right = null
2 → right



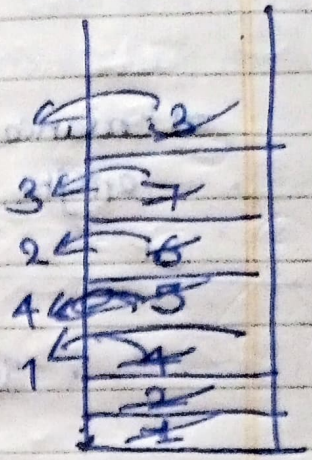
Print → 4, 2, 6, 5, 7, 1, 3

6 → (left, right = null)
5 → right 7
7 → (left x right x
1 → right → 3
3

Post order: (left, right, root)



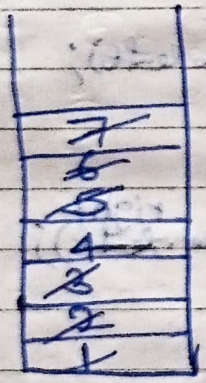
node = 1
= 2
= 4
left, right = null
right
5
6
7



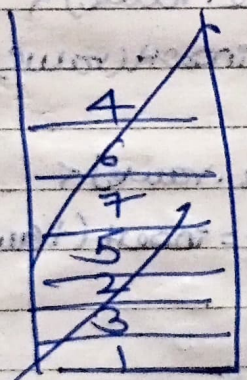
print
4, 6, 7, 5, 2, 3, 1

while(true) {
if(temp1 = Null)

By taking two stack



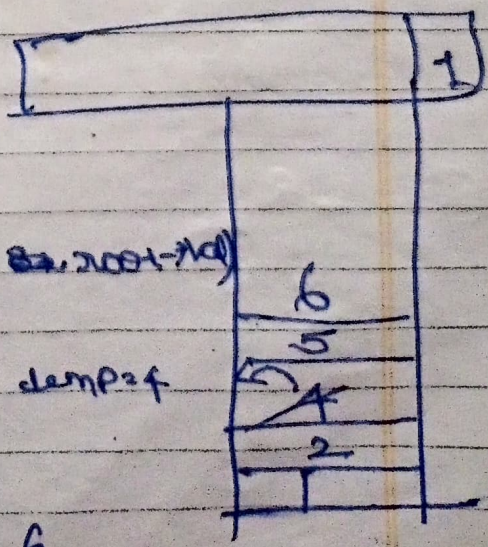
St1



St2

4, 6, 7, 5, 2, 3, 1

st.Push(root)
while(!stEmpty())
root = st.top()
st.pop()
ans.insert(ans.begin(), root->val)



ans = [4, 6

in c++.

Check for Balanced Binary Tree

→ balanced mean left subtree height - right subtree height is not more than 1

insert in BST

→
```
Private Node insert(int val, Node node){
  if(node == null){
    node = new Node(val);
    return node;
  }
  if(value < node.value){
    node.left = insert(value, node.left);
  }
  if(value > node.value){
    node.right = insert(value, node.right);
  }
```

Question Solution
Q24
Logic

count prime

→ vector<bool> prime(n+1, true) // n+1 tak sabko prime maan liya

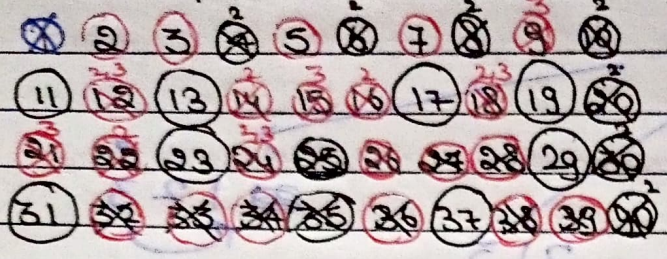
→ by normal it give TLE

0,1 → not prime

• Sieve of Eratosthenes :- n=40

→ mark each no as prime

→ jis prime no ke table mai jo aata yaa उसे हटा do



for(int i=2 ; i<n ; i++)
if(prime[i]) { cnt++; }
for(int j=2*i ; j<n ; j=j+i)
means us table mai jitne aate ho sabko not prime mark kardo

search a 2D array

row length = no. of col

0	1	3	5	7
1	10	11	16	20
2	23	30	34	60

1	3	5	7	10	11	16	20	23	30	34	60	
0	1	2	3	4	5	6	7	8	9	10	11	

end = row × col - 1

int element = matrix[mid/col][mid%col]

↓

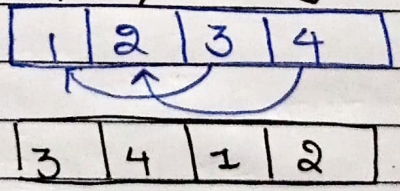
matrix ko convert into array

reverse string

s=0 , e= n-1 element

swap[s(st), s(e)]; st++, e--;

rotate array



K=2

2 value ko aage badhane hai

0 1 2 3 4

11, 12, 13, 14, 15

K=3 → 3 value ko change karna hai

arr[(i+k)%n] = num[i];

agar same array mai rakhte to koi no. repeat kar jaye isliye new array mai rakhnge

Step 1 If same array> arr[(i+k)%n] = arr[i]

for(int i=0)
arr[2%n] = arr[i]
arr[2] = arr[0]
{13, 12, 13, 14, 15} So it is wrong

move all zero to end

```
int j=0
for(int i=0; i<n; i++)
  if (num[i]!=0)
      swap(i,j)
      j++;
  }
}
```

nums = [0,1,0,3,12]
op = [1,3,12,0,0]
step 1 0 not go in if condition
step 2 1. if condition swap(i,j)
now j=1 j++ (1,0,0,3,12)
step 3 0(2) not go in if condition

Student attendence I

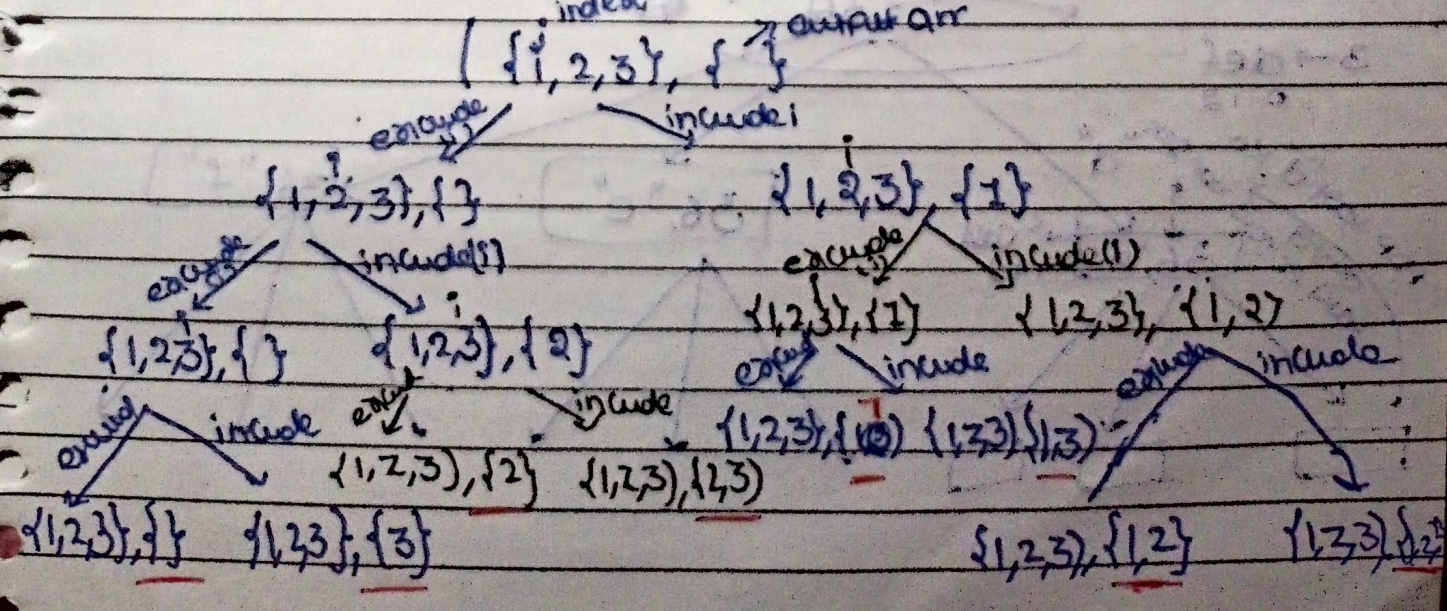
→ The student was absent(A) for strictly fewer than 2 day total
→ never late('L') for 3 or more consicutive days.
→ return true if the student is eligible for an attendence award, or false otherwise

```
countA=0 countL=0
for(int i=0; i<=n; i++){
  if (S[i]=='L'){
    countL++;
  } else {countL=0}
  if(S[i]=='A') countA++;
  if(countA>=2 || countL>=3)
      return false;
}
return true;
```

Subset Problem

i/p → arr[3] = {1,2,3}

Power set = {{}, {1}, {2}, {3}, {1,2}, {2,3}, {1,3}, {1,2,3}}



I ko 1 se start kiya do option nai include or exclude. agar include kiya to new array output array mai dal diye.

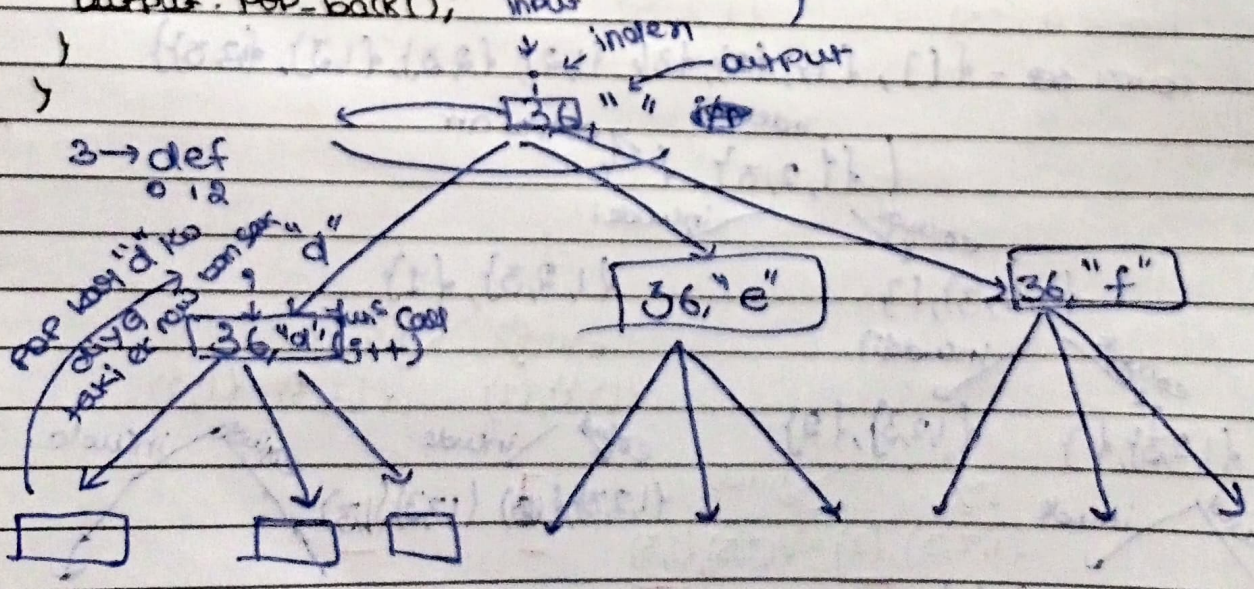
Subset Problem (Phone maze)

→ Recursive call is same as subset

Code

```
void String(String digit, String output,
            int index, vector<string> &ans,
            string mapping[])
// base case
if(index >= digit.length())
    ans.push_back(output);
    return; }
int num = digit[index] - 0
// for getting exact string
digit to normal mapping
string value = mapping[num];
for(int i=0; i< value.length(); i++){
output.push_back(value[i]);
solve(digit, output, index+1, ans, mapping)
output.pop_back(); }
}
```

```
// main code
vector<string> letterCombination
(string digit){
vector<string> ans
if(digits.length()==0)
    return ans;
string output = "";
string mapping[10] = {"", "",
"abc", "def", "ghi", "jkl",
"mno", "pqrs", "tuv", "wxyz"}
solve(digit, output, index, ans
    mapping);
return ans;
}
```



Remove duplicate from sorted array

```
{
int P=0
for(int i=0; i<nums.size(); i++){
  if(nums[P] != nums[i]){
    nums[P+1]= nums[i];
    P++;
  }
  return P+1;
}
```

Permutation II

```
void solve(vector<int> nums, index, k, vector<vector<int>>ans){
  // base case
  if(index>=k-1)
    ans.push_back(nums);
    return; }
for(int j= index ;j<k;j++)
  if(index != j && nums[index]==num[j] continue;
    swap(num[index], num[j]);
    solve( nums, index+1, k, ans);
  }
}
```

index=0, j=0 [1,1,2]
num[index]=1 nums[j]=1
continue ho jayega
loop start j++
index=0, j=1
num[index] == num[j] =1
continue
loop start with j++
index=0, j=2
num[index]=1, num[j]=2
[1,1,2] → swap[index, j];
result [2,1,1]
solve (nums, index+1, ans)
for [2,1,1] → same all of above
→ result [1,2,1]
index+1 → now for 2 index
→ result [1,2,2]

reverse bit (32 bits integer) (char)

n = 00000010100101000001111010011100 → 43261596

reverse bit one by one

Create new variable result = 0

00000000000000000000000000000 → 32 bits

ek variable banenge jo ki bataega n & 1 ka zero ya (ya jab 1 aayega)

add Binary

```
string addBinary(string a, string b){
    int carry = 0
    string res;
    int n = a.size(), m = b.size()
    int i = n-1, j = m-1
    while(i >= 0 || j >= 0)
        int sum = carry;
        if(i >= 0){
            sum = a[i--] - '0';
        if(j >= 0){
            sum += b[j--] - '0';
        res.in
        carry = sum > 1 ? 1 : 0;
        res.insert(res.begin(), char(sum % 2 + '0'));
        //convert sum from int to char and then insert into
        front of result
    }
    if(carry) // if still equal to 1:
        res.insert(res.begin(), '1');
    return res;
}
```