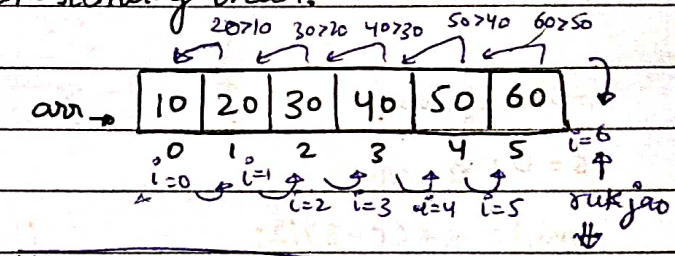
RECURSION CLASS-3

Date 13./.10./.23...

Q check the given array is sorted or not, in ascending order.

f(arr, size, index) — index = 1

(II) if arr[index] > arr[index-1]
- True → f(arr, size, index+1)
- false → return false



if (index ≥ size) → return true → Base Case

(I) main f(array, size, index)
arr, 6, (1) → for comparing it with previous ones.

Code:-

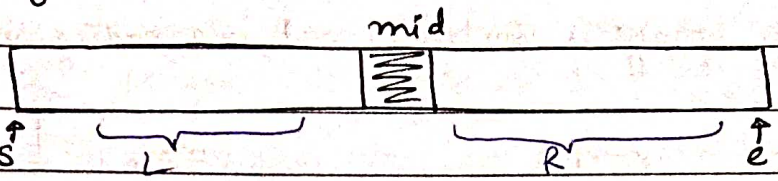
```
bool isSorted ( int arr[], int size, int index) {
    // Base Case
    if( index ≥ size ) return true;
    // Processing
    if ( arr[index] > arr[index-1]) {
        // Aage check krna padega ab
        // Ab recursion sambhalega
        bool aagekaAns = isSorted (arr, size, index+1);
        return aagekaAns;
    } else {
        return false; // Sorted nahi h
    }
} main
```

arr → 10 20 30 (0 1 2)
Size = 3
Index = 1
main → f(arr, size, index)

1 ≥ 3	2 ≥ 3 → F	3 ≥ 3 → T
if(index ≥ size) X return true;	if(index ≥ size) X return true;	if(index ≥ size) return true;
if(arr[index] > arr[index-1]) 20 > 10 → T	if(arr[index] > arr[index-1]) 30 > 20 → T	if(arr[index] > arr[index-1])
return f(arr, size, index+1);	return f(arr, size, index+1);	return f(arr, size, index+1);
else return false;	else return false;	else return false;
True	True	True

Date ..../..../.....

Binary Search using Recursion:-



n = size of array

s = 0; e = n-1;

mid = s + (e-s)/2;

Base case

if (s > e)
  return -1;

while (s > e) //Base case

if (arr[mid] == target) //found wala case
  return mid;

if (arr[mid] < target) → Right me jao
  return f(arr, mid+1, e)
  (s, e)

else → Left me jao
  return f(arr, 0, mid-1)
  (s, e)

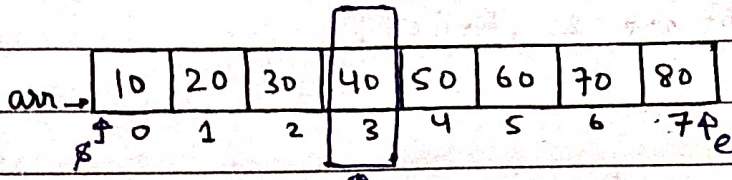
Code:-

```
int binarySearch (int arr[], int s, int e, int target){
    // Base Case
    if ( s>e) return -1;
    // Processing → 1 case khud solve Kro
    int mid = s + (e-s)/2;
    if (arr[mid] == target)
        return mid;
    // Baaki recursion sambhal lega
    if (arr[mid] < target) {
        // right me jao
        // s = mid+1 karke jyange so we can write
        return binarySearch (arr, mid+1, e, target);
    }
    else {
        // Left me jao
        // e = mid-1 karke so we can write
        return binarySearch (arr, s, mid-1, target);
    }
}
```

Spiral

Date ..../..../.....

Dry Run:-



call (arr, 0, 7, 80) → 6 returned

f(arr, s, e, target)

s > e →

//B.C 0 > 7 → F x

//Processing mid = (0+7)/2 = 3

40 == 80 → F

//R.R If(arr[mid] < target)

40 < 80 → True

return f(arr, mid+1, e, target)

return 6

f(arr, 4, 7, 80)

//B.C s > e → 4 > 7 F x

//Process mid = (4+7)/2 = 5

60 == 70 → F

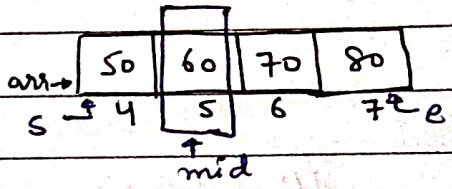
//RR If(arr[mid] < target).

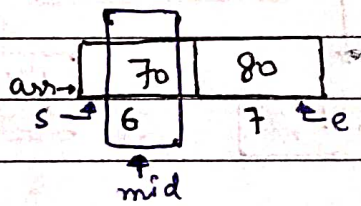
60 < 70 → T

// right me jao, s = mid+1

return f(arr, mid+1, e, target)

return 6





f(arr, 6, 7, 80)

//B.C → s > e → 6 > 7 → F x

//Processing mid = (6+7)/2 = 6

if(arr[mid] == target)

70 == 70 → True

return ⑥ → mid ki value.

Subsequences of String :- [Include - Exclude Pattern]

From the given string, we can include or exclude any character in our output string. But ordering of characters must be same as it's in input string.

Ex:- i/p → "abc" ⇒ subsequences

✓ ✗ ✗ → a
✗ ✓ ✗ → b
✗ ✗ ✓ → c
✓ ✓ ✗ → ab
✗ ✓ ✓ → bc
✓ ✗ ✓ → ac
✓ ✓ ✓ → abc
✗ ✗ ✗ → ""

i/p → "xy" ⇒ subsequences

✓ ✗ → x
✗ ✓ → y
✓ ✓ → xy
✗ ✗ → ""

∴ n = length of String → Total subsequences = 2^n

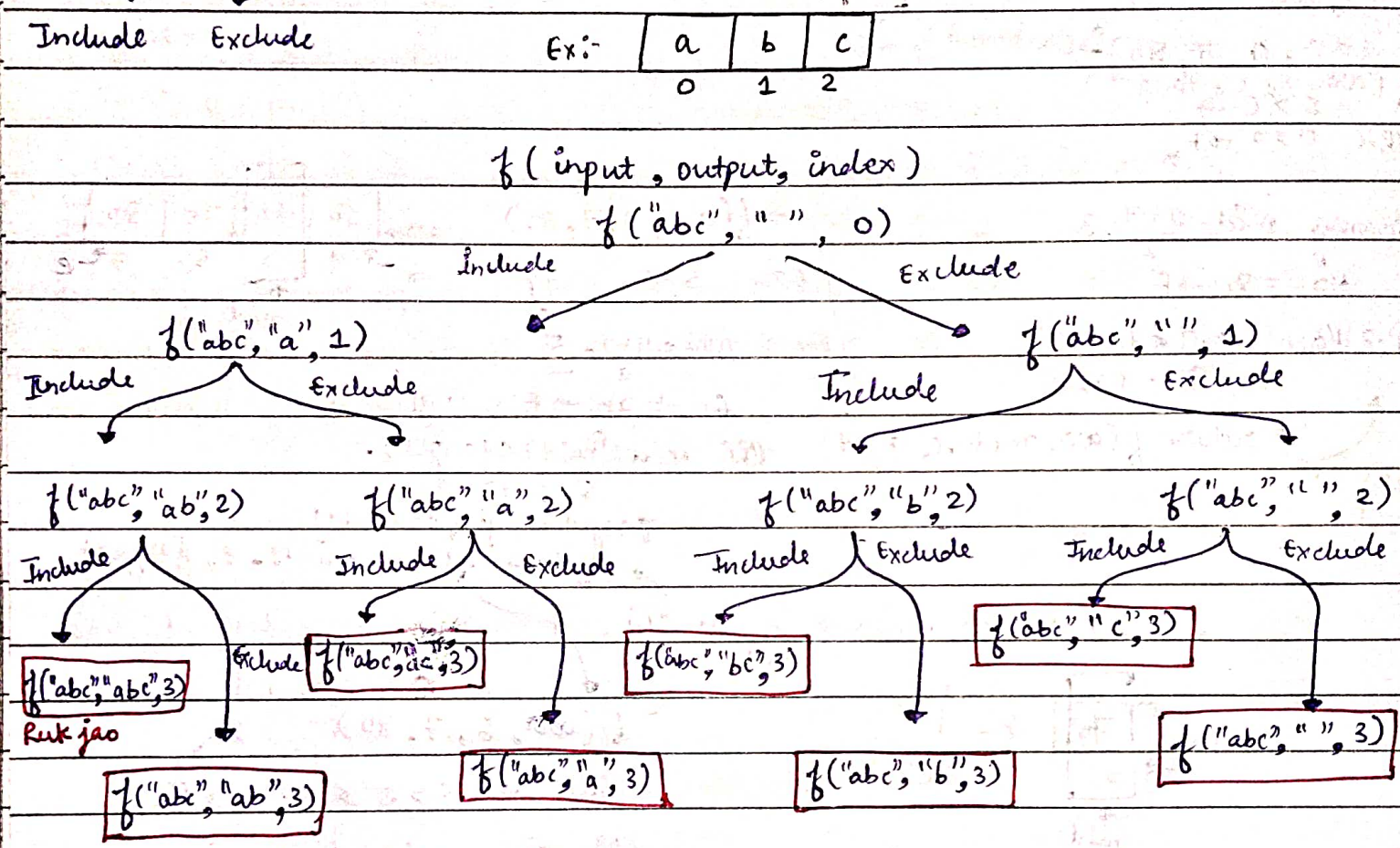
Date ..../..../.....

for every character, there are 2 choices.

'x'

Include Exclude

Ex:- | a | b | c |
0 1 2



Code:-

```
void findSubSequences ( string input, string output, int index ){
    //Base Case
    if (index ≥ input.length()) {
        //ans jo hai, wo output string me built ho chuka hai
        //print krdo
        cout << output<<endl;
        return;
    }
    // Processing
    char ch = input[index];
```

Date ..../..../.....

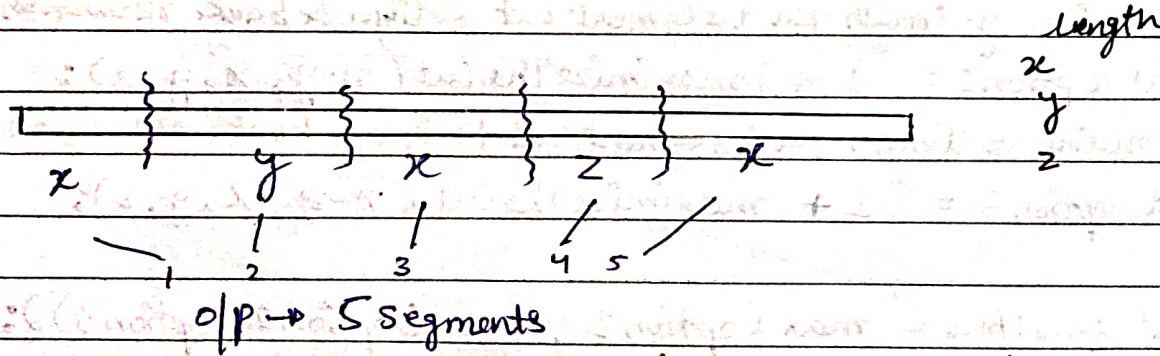
```
// include
output.push_back(ch);
findSubSequences(input, output, index+1);
// exclude
output.pop_back();
findSubSequences(input, output, index+1);
}
```

Exploring all Possible ways Pattern

Q Cut into segments / Maximize The Cut Segments (GFG)

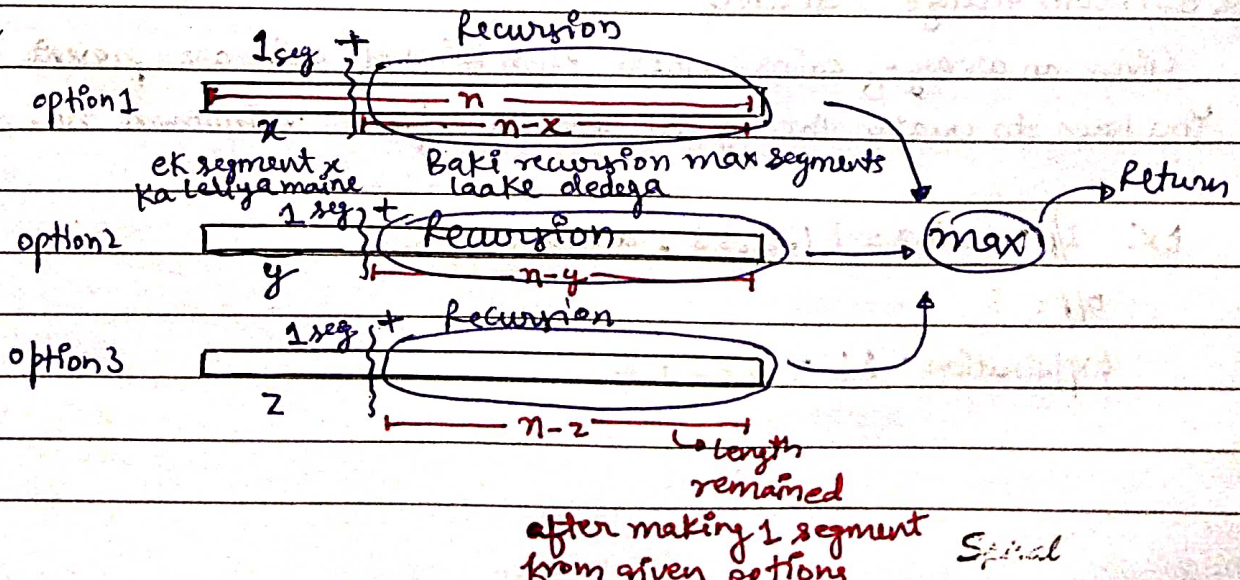
i/p given a rod of n length. We can break it into 'x', 'y' & 'z' segments.
x, y, z are integers.

o/p → return the maximum no. of possible segments of x, y & z lengths.



Keep in mind while applying this approach

→ Exploring All possible ways pattern (Ek case tum solve karo baki recursion sambhal lega)



Spiral

Date ..../..../.....

Code:-

```
int maximizeTheCuts(int n, int x, int y, int z){
    //Base Case
    if(n==0){
    // Jab length hi nahi hai rod ki to ans kya return krange go
    return 0;       // Length of rod = 0 ke liye no. of segments = 0
    }

    if(n<0){
    // Jab <0 wali condition aye to max me wo consider hina ho
     return INT_MIN;
    }

    // maine x length ka 1 segment cut krliya and baaki recursion dekhlega
    int option1 = 1 + maximizeTheCut(n-x, x, y, z);
    // maine y length ka 1 segment cut krliya & baaki recursion dekhlega
    int option2 = 1 + maximizeTheCut(n-y, x, y, z);
    // maine z length ka 1 segment cut krliya & baaki recursion dekhlega
    int option3 = 1 + maximizeTheCut(n-z, x, y, z);

    int finalAns = max(option1, max(option2, option3));
    return finalAns;
}
```

Q 322. Coin Change (Leetcode)

Given an array of coins where each kind of coin are present in infinite number. You have to create the given amount from minimum no. of coins.

Ex:- I/p: coins = [1,2,5], amount = 11

o/p: 3

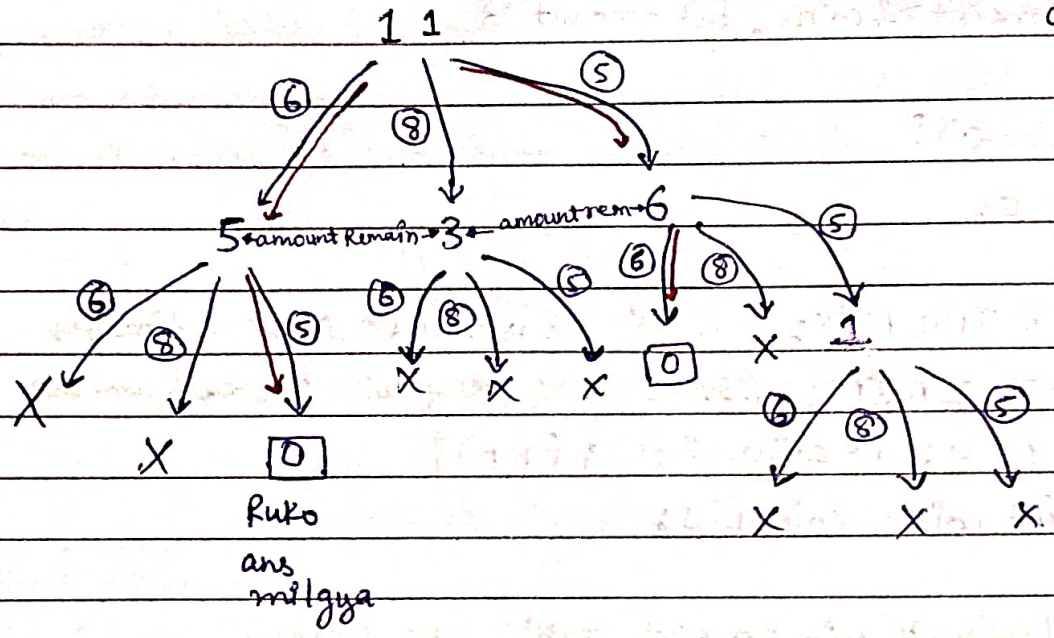
Explanation: 11 = 5+5+1

Date ..../..../.....

∴ Recursive Tree

Amount = 11

coins → [6, 8, 5]



Coin > amount value

↓

no recursive call

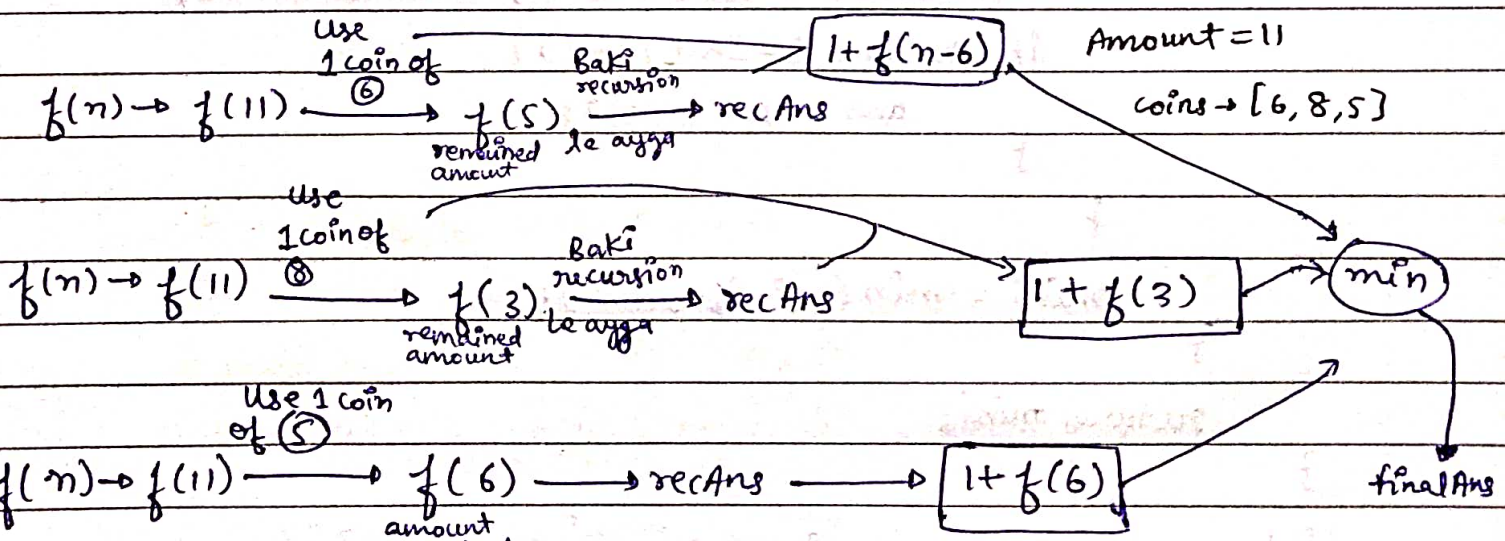
Base Case

```
if (n==0)
  return 0;
```

5 + 6 → 11 2 coins

6 + 5 → 11 2 coins

→ min → 2 coins



Dat: ../.....

Code:-

```
int solve (vector<int>&coins, int amount){
    //Base Case
    if(amount == 0){
        return 0;
    }
    int mini = INT_MAX;  // It's the variable to store finalAns
    int ans = INT_MAX;  //Store ans of every coin to create amount
    for(int i=0; i<coins.size(); i++){
        int coin = coins[i];

        // current coin ko sirf tabhi use krange
        // Jab uskivalue <= amount hogi.
        if(coin<=amount){
            int recAns = solve(coins, amount-coin);
            if(recAns != INT_MAX){
                ans = 1 + recAns;
            }
        }
        mini = min(mini, recAns);
    }
    return mini;
}
int coinChange(vector<int>&coins, int amount){
    int ans = solve(coins, amount);

    if(ans == INT_MAX)
      return -1;
    else
      return ans;
}
```

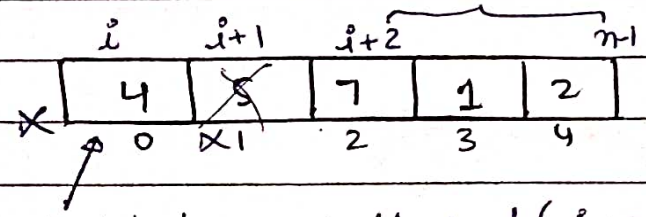
Spiral

Date ..../..../....

Max sum of non-adjacent elements

Q. 198. House Robber (leetcode)

Given an array of houses. we can only rob adjacent houses. We have to return the maximum amount which can be rob.



chori Krta hu ⟶ 4 + f(i+2, n-1) ⟶ final Ans

max ⟶

chori nahi Kri ⟶ 0 + f(i+1, n-1)

T.C

i ≥ size ⟶ rukjao

Code:-

```
int solve (vector<int> &nums, int size, int index){
    //Base Case
    if (index >= size){
        return 0;
    }

    // chori karlo ⟶ ith index pr
    int option1 = nums[index] + solve(nums, size, index+2);
    // chori mat karo ⟶ ith index pr
    int option2 = 0 + solve (nums, size, index+1);

    int finalAns = max (option1, option2);
    return finalAns;
}

int rob (vector<int>&nums) {
    int size = nums.size(); int index = 0;
    int ans = solve (nums, size, index);
    return ans;
}
```