ITC
Enriching Value

classmate

Music

edu GAMES inside

LEARN WHILE YOU PLAY

include <bits/stdc++.h>

100%

▷ • abs(a-b)
max(a,b)
Swap(a,b) → Directly

▷ • # include <vector> → • vector<int> arr(n);
ans.Push_back(arr[i]);
ans.Pop_back();
vector<vector<int>> arr2; → n = arr.Size(
c = arr[0].Size

include <set> → set<int> ans; → ans.insert(a[i])

include <limits.h> → INT_MAX
INT_MIN

include <unordered_map> → Unordered_map<int, int> hash; → hash[a[i]]++;

include <algorithm> → • reverse(arr.begin(), arr.end())
• sort(arr.begin(), arr.end())
dec order → • sort(arr.rbegin(), arr.rend())

include <numeric> → • accumulate(arr.begin(), arr.end())

• unordered<...
• Pair<...
• if(m.find()!=m.end())
• auto
• for(auto...

▷ • for (auto i: arr) {
cout << i;
}

• for (auto i: hash) {
cout << i.first << i.second;
}

hash[i] = 2;
(auto as a single value → int, char)

• vector<Pair<int, int>> & mem;
for (int auto & [m, n] ...)
mem.Push_back({ , , 3});
first second

• Pair<string, int> max = {"...", 0};
mx["...abb"] = 2;
• if (m.find("a") != m.end()) {
cout << "found"
}

only for vector
(unordered map)

★ arr.insert(arr.begin()+1, 2)
arr.erase(arr.begin()+1)

• 2 Pointer Approach
• Basic
• Index Approach
→ I use only vector to Solve this module, if Array is given I copy everything in vector using Push_back.

arr[0] != arr[arr[0]]

▶ DEADLY COMBO → Sort + 2 Pointer

• # include <math>

• int index = lower_bound(arr.begin(), arr.end(), Opt[i]) - arr.begin();
int element = arr[index]
→ sqrt(n)

1001

Rishabh 100%
Best for

▷ Linear Search
- count no. of 1's in an array of (0,1)
- Maximum no. of Array
- Extreme point in Array
- Reverse the array

▷ Find unique Element
- Union of 2 array
- Intersection of 2 array
- Union with Duplicate
- Pair sum
- Triplet Sum
- Sort 0's and 1's

▷ Column-wise sum
- Linear Search using 2D array
- Find max in 2-D array
- Transpose of a Matrix

▷ Sort colors (0,1,2) → 3 pointers
- Move all -ve no. to left side
- Find duplicate no. → in range [1, n]
- Missing element with duplicate → in range [1, n]
- find first repeating element → hash
- Common elements in 3 sorted array
- Wave print a Matrix
- Spiral ~~Print~~
- Left rotate of array by d-element
- Majority element in array
- Buy and Stock Level-1

▷ Add 2 no. of 2 array/vector.
▷ Factorial of a large number

```
# include <algorithm>
# include <vector>
```

▷ int max = * max_element (arr.begin(), arr.end()) ;
▷ int min = * min_element (arr.begin(), arr.end()) ;

▷ Unordered_map< char, int > hash;

```
for(auto ch : s){
    hash[ch]++;
}
```

▷ For fill data through unordered_map :-

```
if(hash.find(s[i]) == hash.end()){
    hash[s[i]] = i
}
```

▷ For fill strings.

```
for(auto i = hash.begin(); i != hash.end(); ...)
    arr.push_back(i->second);
}
```

100%

▷ • # include < string > ; ⟶ • string str ;
getline (cin, str)

▷ • str . size ()

- str . push_back (' ');
  str . pop_back ();

- a . compare (b) == 0 ✓

- Sentence . find (target) → Valid index ✓
  → string :: npos ✗

- str . substr (0, 5)
  name . replace (0, 5, word)
  str . erase (0, 4)
  ↓ Starting index ↘ how many characters

- str = str + str2 ;
  str . append (str2) ;

- int a, int b
- string, string b
- pair <int, int> p, pair <int, int> b

static bool comp (------) E

3

▷ sort (nums . begin (), nums . end (), comp);

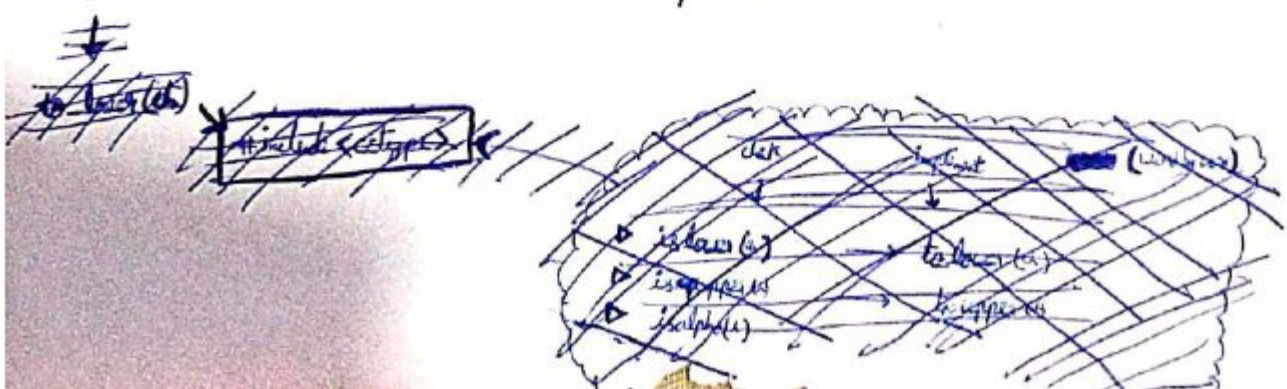
▷ + 'a' → int to string
  - 'a' → string to int

for Sentence → transform (.., ..., ..., :: tolower

▷ • S - I → stoi (num)
  I - S → to_string (num)

▷ # include < algorithm >
- reverse (str . begin (), str . end ());
- sort (str . begin (), str . end ())

▷ L - U → ch[i] = ch[i] - 'a' + 'A' ; [A-65 a-97]
  U - L → ch[i] = ch[i] - 'A' + 'a' ;



- Remove all adjacent Duplicate in a string
- Remove all occurence of a string
- Palindrome - II
- Minimum time difference
- Palindrome Substring

- Valid anagram
- Reverse only
- Longest common Prefix
- Reverse vowels of a string

- Isomorphic Strings
- Recognise Strings
- Group Anagrams
- Longest Palindrome Sub-string
- Index of 1st occ in a string
- String compression (Check last logic)
- Integer to Roman
- Zig zag conversion
- Largest Number

100%

- Binary Search
  - Find 1st occ. of element
  - Last occ of element
  - Total occ. of the element
- Missing element without duplicate
- Peak element in Mountain Array
- Square root of a No. using B.S
- Binary search in 2D matrix
- Binary search in Nearly sorted array
- Divide 2 no. using BS
- Find the odd occuring element in an array
- a) Pivot element
  b) Search in a rotated & sorted array
- K- diff Pairs in an array
- Find K- closest element

▷ set < Pair < int, int >> ans;

- Exponential Search
- Book allocation Problem
- Painters Partition Problem
- Aggressive cows
- Eko Spoj
- Prata Spoj

▷ • Factorial

100%

- Climbing Stairs
- Fibonacci Series
- Find Max^m of array
- Print the element of array
- Key Present or not
- Print all digit of the no.
- Check the array is sorted or not
- Binary Search
- Print all Subsequence of string
- Min. no. of element required to get target
- Cut into Segment
- Max^m Sum of Non-adjacent Elements
- Last occ. of a char
- Palindrome check
- Reverse a string
- Buy and Sell stocks
- House Robber
- Wildcard Matching
- Perfect Square
- Min^m cost for Tickets
- Number of Dice roll with target sum

▷ • Integer to English words

STACK

- #include <stack>
- stack < int > s

s.push()	s.empty()
s.pop()	s.size()
s.top()	

- Reverse the String
- Find the middle element in stack
- Insert At Bottom
- Reverse a Stack

- Valid Paranthesis

- Insert Sorted
- Sort Stack

- Remove Redundant Bracket

▶ Next Min

▶ Longest Valid Paranthesis

- Next Smaller Element
- Prev Smaller Element

■ Largest Rectangle in Histogram

▶ Iron Station

▶ LL

▶ Bit Manipulation

100%

- Permutation of a string
- Rat in a Maze
- N-Queen
- Generate Parenthesis
- Letter combination of a Phone Number
- Sudoku Solver

- Count Inversion → only brute force, I know
- Maximum Subarray
- Combination Sum
- Combination Sum - 2
- Permutation - 2
- Beautiful Arrangement
- Distribute Repeating Integers

- all Sorting algo (+ In-Place Merge Sort)
  - least Inversion optimal

Check for Interview Only

Before Interview

- vector < Pair < int, int >> & numster;
- for (auto i = mp.begin(); i != mp.end(); i++) {
  ans.Push_back (i -> Second)
  }
- mp[s].Push_back (str)
- for (auto i : mp) {
  cout << i.first << i.second;
  }
- # include < unordered_map>
  unordered_map < char, int > mp;

100%

- coin change
- House Robber
- Painting fence
- Count derangement
- 0/1 Knapsack
- Subset Sum
- Equal Subset Sum Partition
- No. of dice Roll with Target Sum
  - ▶ Min^m cost tree from leaf values
- Guess no. higher or Lower - 2
- Longest common Subsequence
  - ↳ ▶ Longest Palindrome Subsequence
- Longest Increase Subsequence
  - ↳ ▶ DP with binary Search
- Russian Doll (1 sort, LIS) (2 sort)
- Max. height by Stacking cuboid (2 sort + LIS)
- Perfect Square
- Min^m cost for Tickets
- Best Time to Buy & Sell Stock - 2/3/4/5
- House Robber - 3
- Predict the Winner
- Word Break - 1
- Target Sum
- Ones and Zeros
- Min^m Swap to make Sequence increasing
- Longest Palindromic Substring
  - ▶ Min ASCII Delete Sum for two strings
- Reducing dishes
  - ▶ Unique BST - 2
- Distinct Subsequence
  - ▶ Word break - 2
- Interleaving String
- Min^m Insertion steps to make a string Palindrome
- Min^m no. of Removals to make mountain array
  - ▶ Make array strictly increasing
  - ▶ Burst Balloons
- Stone game - 1
  - ▶ Stone game - 2

100%

GREEDY

- Min platform → 2a → 1 Type → N-Meeting in one Room (or activity Selection)
- Buy 2 lollipop and get 1 for free
- Deffkin - Defense of a Kingdom
  - Fractional Knapsack (1.0) a/b ← value in double
- Min cost of ropes
- Chocolate Distribution Problem
- Job Sequencing Problem

▶ For moving Round & Round in array,

use this

int idx = (i + k - 1) % arr.size()

INDEX

NAME: _____ STD.: _____ SEC.: _____ ROLL NO.: _____ SUB.: _____

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
		'There is nothing called Difficult, Either it takes less time OR more time'		
		'Consistency is the only solution of all your fear and Self Doubt'		
		'50% right direction > 100% wrong direction'		
		DSA > CGPA		
		'No matter, the Question is easy or hard, you have to think the Logic and require time to code'		
		'The last 1 Months - Full Focus / more hour - Study ≈ Requires' [It's Biggest Point]		
		'Don't forget how you stuck here, maybe you forget your starting, only Matter is ENDING'		
	hey, Its March from 1 March 2023, you need had killed 10, & 3 years of Very hard work (2023), Sorrow get VIT CSE, & one after Right, you are in the 3.5+ in 6th Sem (where clearly you stand at this stage) ↓ Reaches to all this here we come to final hall after 2 months DSA, here is the last 1 Month of Revision, two year intense hard work	(have the attitude to [illegible] in 2 days)		
	Rishi 1/3/24			

, arr[and]);

i=0; i< Size; i++) {

S. No.	Date	Title	Page No.	Teacher's Sign / Remarks
1 →			• 2-Pointer approach (m3) • Basic • Index Approach	
2 →			• Basic approach + APPLICATION → basic → Space & Space → index	
3 →				
6 →				
9 →				
10 →				
4 →				

Acco
our heal

1- ARRAY

• Array

```
• int arr[10];
  for(int i=0; i<10; i++){
      cin >> arr[i];
  }
```

```
• int arr[10];
  for(int i=0; i<10; i++){
      cout << arr[i];
  }
```

• #include < limits.h >
↳ for INT_MAX, INT_MIN

• abs(a-b)
• Swap(a,b)
• max(a,b)
~~• length()~~
→ use directly

```
* int find ( ) → int
* void find ( ) → find()
* bool find ( ) → if (find()) else
```

★ QUESTIONS :-

① Linear Search

```
bool find(int arr[], int size, int Key){
    for(int i=0; i< Size; i++){
        if (arr[i] == Key){
            return true;
        }
    }
    return false;
}
```

② Count number of 1's in an array of (0,1)

```
int find(int arr[], int size){
    int count = 0;
    for(int i=0; i< size; i++){
        if(arr[i] == 1){
            count = count + 1;
        }
    }
    return count;
}
```

③ Maximum number of Array

```
int find(int arr[], int size){
    int max = 0;  // INT_MIN
    for(int i=0; i< Size; i++){
        if(max < arr[i]){
            max = arr[i];
        }
    }
    return max;
}
```

int ans = find()

④ Extreme Print in Array

```
void find(int arr[], int size){
    int start = 0;
    int end = size-1;
    while(true){  // or use for-loop
        if(start > end){
            break;
        }
        if(start == end){
            cout << arr[start];
            return;
        }
        else{
            cout << arr[start];
            cout << arr[end];
        }
        start++;
        end--;
    }
}
```

find()

⑤ Reverse the array

```
int find(int arr[], int size){
    int start = 0;
    int end = size-1;
    while(start <= end){
        Swap(arr[start], arr[end]);
        start++;
        end--;
    }
    for(int i=0; i< Size; i++){
        cout << arr[i];
    }
```

• **Vectors**

↳ #include <vector>

• vector <int> arr {1,2,3};

• arr.size() // for get the size

• vector <int> arr2(n) // before that → int n; cin>>n;

arr2.push_back(5);

• vector <int> arr3(n,7) // sabko 7 se initialize karlo

```
for(int i=0; i< arr.size(); i++) {
    cout << arr[i];
}
```

```
for (auto i : arr2) {      // auto ke jagah specific v
    cout << i;             // bta sakte ho → int/float
}
```

★ **Questions**

⑥ **Find Unique Element** → { 1^2^3^1^2 = 3 }

```
int find(vector <int> arr) {
    int ans=0;
    for(int i=0; i< arr.size(); i++) {
        ans = ans ^ arr[i];
    }
    return ans;
}
```

⑦ **Union of 2 Arrays** (with duplicate)

```
int n;
cin>>n;
int m;
cin>>m;
int arr1(n);
int arr2(m);
for(int i=0; i<n; i++) {
    cin>> arr[i];
}
for(int i=0; i<m; i++) {
    cin>> arr2[i];
}
vector <int> ans;
for(int i=0; i<n; i++) {
    ans.push_back(arr[i]);
}
for(int i=0; i<m; i++) {
    ans.push_back(arr2[i]);
}
for(int i=0; i< ans.size(); i++) {
    cout << ans[i];
}
```

⑧ **Intersection of 2 array:-**

```
vector <int> arr1{1,2,3,4}
vector <int> arr2{2,4,6}
vector <int> ans;

for(int i=0; i< arr1.size(); i++) {
    for(int j=0; j< arr2.size(); j++) {
        if(arr1[i] == arr2[j]) {
            ans.push_back(arr[i]);
        }
    }
}
```

⑨ Union with Duplicate

```
Vector < int > arr1{1,2,3,4,5}
Vector < int > arr2{ 2,4,6}
Vector < int > ans;

for(int i=0; i< arr1.Size(); i++){
    for(int J=0; J< arr2.Size(); J++) {
      if(arr1[i] == arr2[J]) {
          arr2[J] = INT_MIN;
      }
    }
  ans.push_back(arr1[i]);
}
for(int i=0; i< arr2.size(); i++) {
    ans.push_back(arr2[i]);
}
for(int i=0; i< ans.Size(); i++){
    if ( ans[i] != INT_MIN) {
        cout << ans[i];
    }
```

⑩ Pair Sum → Sorted → (works on if array is Not sorted)

```
Vector < int > arr{10,20,40,60,70}
int sum=100;
for(int i=0; i< arr.size(); i++) {
    int element = arr[i];
    for(int J=i+1; J< arr.size(); J++) {
        if ( element + arr[J] == sum) {
            cout << element << "," << arr[J];
            break;
        }
    }
}
```

Using 2 Pointer

20	40	60	80	90

i, j

Sum = 100

logic

→ 20+90 > 100 (J--)

→ 20+80 = 100 (return)

→ 20+60 < 100 (i++)

O(n)

Pre sorted

use 2 Pointer only if the array is sorted

⑪ Triplet Sum

```
Vector < int > arr {10,20,30,40};
int sum= 80;
for (int i=0; i< arr.size(); i++) {
    int element = arr[i];
    for(int J= i+1; J< arr.size(); J++) {
        int ele2 = arr[J];
        for(int K = J+1; K< arr.size(); K++){
            if(element + ele2 + arr[K] == sum) {
                cout << element << ele2 << arr[K];
                break;
            }
        }
    }
}
```

⑫ Sort 0's and 1's

```
Vector<int> arr{0,1,0,0,1,1,0,0,1,0};
int start = 0;
int end = arr.size()-1;
int i=0;
while(i != end) {
    if(arr[i] ==0) {
        swap(arr[i], arr[start]);
        i++;
        Start++;
    }
    else { // arr[i] ==1
        swap(arr[i],arr[end]);
        end--;
    }
}

for(int i=0; i< arr.size(); i++) {
    cout << arr[i];
}
```

↓

only 2 Pointer

↓

```
while(i<j)
    if i==0 → i++
    else if J==1 → j--
    else swap(arr[i], arr[j])
```

• 2D-array

```
for(int i=0; i<n; i++){
    for(int j=0; j<m; j++){
        cout << arr[i][j] << " ";
    }
    cout << endl;
}
```

(13) Column-Wise Sum

```
void func(int arr[][3], int row, int column){
    for(int i=0; i< row; i++){
        int sum = 0;
        for(int j=0; j< column; j++){
            sum = sum + arr[j][i];
        }
        cout << sum;
```

(14) Linear Search using 2D array

```
int func(int arr[][3], int row, int column, int Key){
    for(int i=0; i< row; i++){
        for(int j=0; j< column; j++){
            if(arr[i][j] == Key){
                return 1;
            }
        }
    }
    return -1;
```

(15) Find max in 2-D array

```
int func(int arr[][3], int row, int column){
    int maxi = INT_MIN;
    for(int i=0; i< row; i++){
        for(int j=0; j< column; j++){
            if(maxi < arr[i][j]){
                maxi = arr[i][j];
            }
        }
    }
    return maxi;
}
```

Reference Theory → C++

(using vector)

```
vector<vector<int>> arr;
vector<int> a{1,2,3};
vector<int> b{4,5,6};
vector<int> c{7,8,9};

arr.push_back(a);
arr.push_back(b);
arr.push_back(c);
```

• sz = arr.size();
c = arr[0].size();

```
for(int i=0; i< arr.size(); i++){
    for(int j=0; j< arr[0].size(); j++){
        cout << arr[i][j];
    }
}
```

• vector<vector<int>> arr(3, vector<int>(5,0));

(vector<vector<int>> arr(row, vector<int>(col));)

(16) Transpose of a Matrix

```
void transpose(int arr[][3], int row,
               int column, int transpo
for(int i=0; i< row; i++){
    for(int j=0; j< column; j++){
        trans[j][i] = arr[i][j];
    }
}

void printarr(int arr[][3], int row, int
    for(int i=0; i< row; i++){
        for(int j=0; j< col; j++){
            cout << arr[i][j] << " ";
        }
        cout << endl;
    }
```

(col-wise sum)

```
void ans(vector<vector<int>> arr, int row, int col){
    for(int i=0; i< row; i++){
        int sum = 0;
        for(int j=0; j< col; j++){
            sum = sum + arr[j][i];
        }
        cout << sum;
    }
}
int main(){
    int row, col;
    cin >> row >> col;
    vector<vector<int>> arr(row, vector<int>(col));
    for(int i=0; i< row; i++){
        for(int j=0; j< col; j++){
            cin >> arr[i][j];
        }
    }
    ans(arr, row, col);
    return 0;
}
```

or → arr.size()

→ arr[0].size()

★ More Questions

(17) **Sort colors** → Given an array nums with n objects colored red, white, or blue, sort them in-place so that objects of the same color are adjacent, with the colors in the ~~order~~ order red, white and blue.
You will use the integers 0, 1, 2 to represent the color red, white and blue, respectively.
You must solve this problem without using the library's sort function.

```
void sortColors(vector<int> nums){
    int s = 0;                  // Pointer for 0s
    int e = nums.size()-1;      // Pointer for 2s
    int mid = 0;                // Pointer for iterating through the array

  while (mid <= e){
      if(nums[mid] == 0){
        swap(nums[mid], nums[s]);
        s++;
        mid++;
      }
    else if (nums[mid] == 1){
        mid++;
      }
    else {
        swap(nums[mid], nums[e]);
        e--;
      }
  }
}
```

• M-1 → sort (n log n)
• M-2 → Count all, then place as 0, 1, 2

```
if(nums[i] == 0)
    zeros++;
if(nums[i] == 1)
    ones++;
if(nums[i] == 2)
    twos++;
int i=0;
while (zeros--){
    nums[i] = 0;
    i++;
while (ones--){
    nums[i] = 1;
    i++;
while (twos--){
    nums[i] = 2;
    i++
```

• M-3 → 3 Pointer approach

(18) Move all negative number to the left side of an array

```
void move(int arr[], int n){
    int s = 0;
    int e = n-1;
    while (s < e){
      if (arr[s] < 0){
        s++;
      }
    else if (arr[e] > 0){
        e--;
      }
    else {
        swap(arr[s], arr[e]);
      }
  }
}
```

• We assume there is no need to maintain the same order
• M-1 → Sort (n log n)
• M-2 → 2 Pointer approach (Two Dutch National Flag

(19) **Find Duplicate Number** → Given an array of integers nums containing n+1 integers where each integer is in the range [1, n] inclusive. There is only one repeated number in nums, return this repeated number.

I/P → [1,3,4,2,2]
O/P → 2

```
int answer(vector<int> ans){
    for(int i=0; i<ans.size(); i++){
      if (ans[0] != ans[ans[0]]){
        swap(ans[0], ans[ans[0]]);
      }
    }
    return ans[0];
}
```

(after operation) [2, 1, 2, 3, 4]

∴ Use Index Method

• M-1 → Sort (n log n)
• M-2 → Mark -ve, once visited according to index. (-ve marking Method)
• M-3 → [3, 1, 2, 3] ← data
[0, 1, 2, 3] ← index
↓
• Make 0th index [illegible] more upon I element h [illegible] main [illegible] index [illegible] change, [illegible] so [illegible]

(20) Missing elements from an array with duplicates

→ Given an array of size n having integers in the range [1,N] with some of the elements missing. The task is to find the missing elements.

(use index Method)

i/p → 1,2,3,4,4,7,7
o/p → 5,6

(negative visited check)

M-1 → In the given range [1,N] there should be an element corresponding to each index. So mark the visited indices by multiplying that element with -1. If an element is missing then its index will have a positive element. otherwise, it will have a negative number.

```
void find(int a[], int n){
    for(int i=0; i<n; i++){
        int index = abs(a[i]);
        if(a[index-1]>0){
            a[index-1] *= -1;
        }
    }
    for(int i=0; i<n; i++){
        if(a[i]>0){
            cout << i+1;
        }
    }
}
```

(index-1) Mil number ko -ve karde Mujhe dikhes do!!

(Logically in their sort) (M2)

M2 → In the given range [1,N] there should be an element corresponding to each index. So we can sort them and then if at any index the position and the element are not same, those elements are missing.

```
void cyclicSort(vector<int> a){
    int n = a.size();
    int i = 0;
    while(i<n){
        int index = a[i]-1;
        if(a[i]!=a[index]){
            swap(a[i], a[index]);
        }
        else{
            i++;
        }
    }
```

→ Sort

// is trick se sort kare to (~ log n) aur kaise O(n) aa

```
int main(){
    ...
    for(int i=0; i<n; i++){
        if(a[i]!=i+1){
            cout << i+1;
        }
    }
```

→ Logic

(21) Find First Repeating Element

i/p → arr[] = {1, 5, 3, 4, 3, 5, 6}
o/p → 2
(index → 1 2 3 4 5 6 7 (index start from 1))

↳ • M-1 → use 2-for loop and check
• M-2 → use of unordered_map

```
int findFirstRepeatingElement(int a[], int n){
    unordered_map<int, int> hash;
    for(int i=0; i<n; i++){
        hash[a[i]]++;
    }
    for(int i=0; i<n; i++){
        if(hash[a[i]]>1){
            return i+1;
        }
    }
    return -1;
}
```

// Pehle hi traverse karke data liye then 2 baar aaya, aur usko hash me store karliye.

then use this line

hash[2,5,6] = {0}

Note → Interview bol sakte h, bina hash ke kaise to Khud ke step bna lenge

(22) Common Elements in 3 sorted array → Given three arrays sorted in increasing order. Find the elements that are common in all the three arrays.

(distinct + order)

↳ • M-0 → using 3 for loop
• M-1 → using 3 pointer approach

Check for empty() → is. empty() → return -1

Check for duplicate → ↳ agar subse do baar 3 array, to output do, ek baar 3 ek baar 4 → use set

```
int i=0;
int J=0;
int K=0;
vector<int> ans; // set<int> ans
while(i<arr.size() && J<arr2.size() && K<arr3.size()){
    if (a[i]==b[J] && b[J]==c[K]){
        ans.insert(a[i]);          // ans.push_back(a[i])   but we use set → for no repetition
        i++;
        J++;
        K++;
    }
    else if (a[i]<b[J]){
        i++;
    }
    else if(b[J]<c[K]){
        J++;
    }
    else{
        K++;
    }

    for(auto i: set){
        ans.push_back(i);          // Push all distinct value to vector → 'ans'
    }
    return ans;
}
return -1;
```

Use

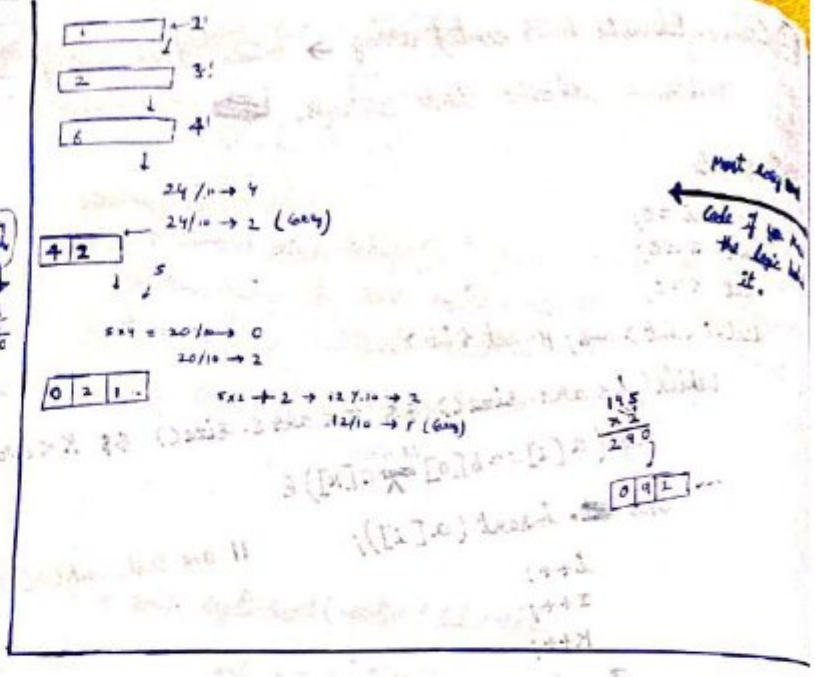
```
for(int val: ans){
    cout << val;
```

(23) Wave Print a Matrix

```
Void WavePrint(vector<vector<int>> arr){
    int row = arr.size();
    int col = arr[0].size();
    for(int J=0; J<col; J++){
        if ((J%1)==0) {        // J=0,2,4 → for even value
            for(int i=0; i<row; i++){
                cout << arr[i][J] << " ";
            }
        }
        else {
            for(int i=row-1; i>=0; i--){
                cout << arr[i][J] << " ";
            }
        }
    }
```

24) Spiral Print

1 2 3
4 5 6
7 8 9 → 123698745




```
Vector < int > ans;
int m = matrix.size();
int n = matrix[0].size();
int total_element = m*n;

int starting_row = 0;
int endingcol = n-1;
int endingrow = m-1;
int startingcol = 0;

int count = 0;

while (count < total_element) {        // isko Main ne Yha lga diya thik h par agar bich me hi Khatam hogya element to
                                          uske liye hm Jagah check Karte rahe.
  // Printing Starting row
  for (int i = startingcol; i <= endingcol && count < total_element ; i++) {
      ans.push_back(matrix[starting row][i]);
    count ++;
  }
  Startingrow ++;
  // Printing ending col
  for (int i = starting row ; i <= endingrow && count < total_element; i++) {
      ans.push_back(matrix[i][endingcol])
      count ++;
  }
  endingcol --;
  // Printing ending row
      for (int i = endingcol; i >= startingcol && count < total_element; i--) {
        ans.push_back(matrix[endingrow][i]);
        count ++;
      }
      endingrow --;
  // Printing starting col
      for (int i = endingrow; i >= startingrow && count < total_element; i--) {
        ans.push_back(matrix[i][startingcol]);
        count ++;
      }
      startingcol ++;
}

return ans;
```

→ O(m) is the time complexity

25 Factorial of a large number

```
vector<int> ans;
ans.push_back(1);
int carry = 0;
for(int i=2; i<=n; i++){
    for(int j=0; j<ans.size(); j++){
        int x = ans[j] * i + carry;
        ans[j] = x%10;
        carry = x/10;
    }
    while(carry){
        ans.push_back(carry%10);
        carry /= 10;
    }
}
reverse(ans.begin(), ans.end());
return ans;
}
```

26 Left Rotate of an array by 'd' element

```
void Rotate(int arr[], int d, int n){
    int temp[n];          // temp array
    int K = 0;
    for(int i=d; i<n; i++){
        temp[K] = arr[i];
        K++;
    }
    for(int i=0; i<d; i++){
        temp[K] = arr[i];
        K++;
    }
    for(int i=0; i<n; i++){
        arr[i] = temp[i];
    }
    // Transferring all element from temp to arr
}
for(int i=0; i<n; i++){
    cout<<arr[i];
}
```

27 Majority elements in an array

```
void Majority(int arr[], int n){
    int maxCount = 0;
    int index = -1;
    for(int i=0; i<n; i++){
        int count = 0;
        for(int j=0; j<n; j++){
            if(arr[i] == arr[j]){
                count++
            }
        }
        if(count > MaxCount){
            MaxCount = count;
            index = i;
        }
    }
    if(maxCount > n/2){
        cout<<arr[index]<<endl;
    }
    else{
        cout<<"No Majority Element"<<endl;
    }
}
```

```
for(int i=0; i<n; i++){
    hash[nums[i]]++;
}
for(auto i : hash){
    if(i.second > n/2){
        ans.push_back(i.first);
    }
}
```

28 Buy and Stock Level-1

→ buy one day, sell different day → Profit should be Max.

```
vector<int> arr = {7,1,5,3,6,4}
int sum = 0;
for(int i=1; i<arr.size(); i++){
    if(arr[i] > arr[i-1]){
        sum = sum + (arr[i] - arr[i-1]);
    }
}
cout<<sum;
```

```
int n = Price.size();
int maxProfit = 0;
for(int i=0; i<n-1; i++){
    for(int j=i+1; j<n; j++){
        int Profit = Price[j]-Price[i];
        maxProfit = max(maxProfit, Profit);
    }
}
return maxProfit;
}
```

i/P → 7, 1, 5, 3, 6, 4
o/P → Buy at day-2, Sell at day-5 → 6-1 = 5 ← output

→ n-1 → 2 loop
n-2 → DP

29 • Add two numbers represented by two arrays

i/P → A[] = {1, 2, 3}, B[] = {2, 1, 3}
o/P → 33,

```
int carry = 0
string ans;
int i = n-1;
int j = m-1;
while(i>=0 && j>=0){
    int x = a[i] + b[j] + carry;
    int digit = x%10;
    ans.push_back(digit + '0');
    carry = x/10;
    i--; j--;
}
while(i>=0){
    int x = a[i] + 0 + carry;
    int digit = x/10;
    ans.push_back(digit + '0');
    carry = x/10;
    i--;
}
while(j>=0){
    int x = 0 + b[j] + carry;
    int digit = x/10;
    ans.push_back(digit + '0');
    carry = x/10;
    j--;
}
```

int x = a[i] + b[j] + carry,
int digit = x%10,
carry = x/10,
i--
j--

if(carry){
    ans.push_back(carry + '0');
}
while(ans[ans.size()-1] == '0'){
    ans.pop_back();
}
reverse(ans.begin(), ans.end);
return ans;

2 - Binary Search

- binary - search (arr.begin(), arr.end(), 15)
- applicable for • sorted search space
  • Monotonic

★ Questions :-

① Binary Search

```
int binarySearch(int arr[], int size, int target){
    int start = 0;
    int end = size-1;
    int mid = start + (end-start)/2;
    while (start <= end){
        int element = arr[mid];
        if (element == target){
            return mid;
        }
        if (target < element){
            end = mid-1;
        }
        else {
            start = mid+1;
        }
        mid = start + (end-start)/2;
    }
    return -1;
}
```

② Find the first occurence of the element

```
int firstOcc(vector<int> arr, int target){
    int s = 0;
    int e = arr.size()-1;
    int mid = s + (e-s)/2;
    int ans = -1;
    while (s<=e){
        if (arr[mid] == target){
            ans = mid;
            e = mid-1;
        }
        else if (target > arr[mid]){
            s = mid+1;
        }
        else if (target < arr[mid]){
            e = mid-1;
        }
        mid = s + (e-s)/2;
    }
    return ans;
}
```

③ Last occurence of the element

```
int lastOcc(vector<int> arr, int target){
    int s = 0;
    int e = arr.size()-1;
    int mid = s + (e-s)/2;
    int ans = -1;
    while (s<=e){
        if (arr[mid] == target){
            ans = mid;
            s = mid+1;
        }
        else if (target > arr[mid]){
            s = mid+1;
        }
        else if (target < arr[mid]){
            e = mid-1;
        }
        mid = s+(e-s)/2;
    }
    return ans;
```

④ Total occurence of the element

↳ Total occ = Lastocc - Firstocc

⑤ Missing Element without duplicate

```
int missingNumber(vector<int> nums){
    int n = nums.size();
    int totalSum = n*(n+1)/2;
    int arraySum = 0;
    for (int i : nums){
        arraySum += i;
    }
    return totalSum - arraySum;
}
```

~~int missing = nums.size();~~
~~for (int i = 0; i < nums.size(); ++i){~~
~~missing = missing ^ i ^ nums[i];~~
~~}~~
~~return missing;~~

⑥ Peak element in Mountain Array

```
int find(vector<int> arr){
    int s=0;
    int e = arr.size()-1;
    int mid = s+(e-s)/2;
    while (s<e) {
        if (arr[mid] < arr[mid+1]) {
            s = mid+1;
        }
        else {
            e = mid;
        }
        mid = s+(e-s)/2;
    }
    return s;
}
```

⑦ Square Root of a Number

```
int find(int n){
    int target = n;
    int s = 0;
    int e = n;
    int ans = -1;
    int mid = s+(e-s)/2;
    while (s<=e) {
        if (mid * mid == target) {
            return mid;
        }
        else if (mid * mid > target) {
            e = mid-1;
        }
        else {
            ans = mid;
            s = mid+1;
        }
        mid = s+(e-s)/2;
    }
    return ans;
}
```

```
int Precision;
cin >> Precision;
double step = 0.1;
double finalans = ans;
for(int i=0; i<Precision; i++) {
    for (double j = finalans; j*j <= n; j = j+step) {
        finalans = j;
    }
    step = step/10;
}
cout << finalans;
```

⑧ Binary Search in 2D matrix

```
int binary(int arr[][4], int row, int col,
           int target) {
    int s = 0;
    int e = row * col-1;
    int mid = s+(e-s)/2;
    while (s<=e) {
        int rowindex = mid/col;
        int colindex = mid % col;
        if (arr[rowindex][colindex] == target) {
            return true;
        }
        if (arr[rowindex][colindex] < target) {
            s = mid+1;
        }
        else {
            e = mid-1;
        }
        mid = s+(e-s)/2;
    }
    return false;
}
```

⑨ Binary Search in a Nearly Sorted Array

```
int binary(vector<int> arr, int target) {
    int s=0;
    int e = arr.size()-1;
    int mid = s+(e-s)/2;
    while (s<=e) {
        if (arr[mid] == target) {
            return mid;
        }
        if (mid-1 >=0 && arr[mid-1] == target)
            return mid-1;
        }
        if (mid+1 < arr.size() && arr[mid+1] == target
            return mid+1;
        }
        if (target > arr[mid]) {
            s = mid+2;
        }
        else {
            e = mid-2;
        }
        mid = s+(e-s)/2;
    }
    return -1;
}
```

(10) Divide '2' Numbers using binary Search

```
int Solve(int dividend, int divisor) {
    int s = 0;
    int e = abs(dividend);
    int mid = s + (e-s)/2;
    int ans = 1;
    while (s <= e) {
        if(abs(mid * divisor) == abs(dividend)) {
            ans = mid;
            break;
        }
        if (abs(mid * divisor) > abs(dividend)) {
            e = mid-1;
        }
        else {
            ans = mid;
            s = mid+1;
        }
        mid = s + (e-s)/2;
    }
    if ((divisor < 0 && dividend < 0) || (divisor > 0 &&
                                          dividend > 0)) {
        return ans;
    }
    else {
        return -ans;
    }
```

(11) Find the odd occuring element in an array

(all elements occur even no. of times except one)
{ • all repeating occurence of element appear in pairs and
{ pairs are not adjacent (there cannot be more than 2 consecutive occurence of any element)

↳ • M·1 → XOR but T·C → (n) / S·C → (1)

i/p → [1,1,2,2,3,3,4,4,3,5,5,6,6]
o/p → 3

```
int find(vector<int> arr) {
    int s=0;
    int e = arr.size()-1;
    int mid = s+(e-s)/2;
    while (s<=e) {
        if (s==e) {
            return s;
        }
        if (mid % 2 == 0) {
            if (arr[mid] == arr[mid+1]) {
                s = mid+2;
            }
            else {
                e = mid;
            }
        }
        else {
            if(arr[mid] == arr[mid-1]) {
                s = mid+1;
            }
            else {
                e = mid-1;
            }
        }
        mid = s+(e-s)/2;
    }
    return -1;
```

(12)(a) Pivot Element → sorted & rotated array → [9,10,2,4,6,8]

```
int find(vector<int> arr) {
    int s=0;
    int e = arr.size()-1;
    int mid = s+(e-s)/2;
    while (s<=e) {
        if (s==e) {
            return s;
        }
        if(mid<=e && arr[mid] > arr[mid+1]) {
            return mid;
        }
        if (mid-1 >=s && arr[mid-1] > arr[mid]) {
            return mid-1;
        }
        if(arr[s] > arr[mid]) {
            e = mid-1;
        }
        else {
            s = mid+1;
        }
        mid = s+(e-s)/2;
    }
    return -1;
```

(12)(b) Search in a rotated and sorted array

```
int pivotIndex = find(nums);
if (target >= nums[0] && target <= nums[pivotIndex]) {
    return binarySearch(nums, 0, pivotIndex, target);
}
else {
    return binarySearch(nums, pivotIndex+1,
                        nums.size()-1, target);
}
```

★ Mord questions

(13) **K-diff Pairs in an array** → Given an array of integers nums and an integer k, return the number of unique k-diff pairs in the array.

i/p → nums = [3, 1, 4, 1, 5], k = 2

o/p → 2

explanation → There are two 2-diff pairs in the array (1, 3) and (3, 5). Although we have two 1's in the input, we should only return the number of unique pairs.

↳ M-1 → 2 for loop lga ke ek-ek karke check karlenge. $O(n^2)$

M-2 → 2 Pointer approach

- Sort

1 loop m hi tag karte hai $O(n)$

```
diff = a[j] - a[i]
if (diff = k){
  ans...
  i++, j++
}
else if (diff > k){
  i++;
}
else {
  j++
}
```

↓

```
int findPairs(vector<int>& nums, int k) {
  sort(nums.begin(), nums.end());
  set<pair<int, int>> ans;
  int i=0; j=1;
while (j< nums.size()) {
  int diff = nums[j] - nums[i];
  if (diff==k) {
    ans.insert({nums[i], nums[j]});
    ++i, ++j;
  }
  else if (diff>k) {
    i++;
  }
  else {
    j++;
  }
  if (i==j) j++;   // Ye bhul jaaye dhyan rakhna
}
return ans.size();   // for size only
}
```

} → Sort karke bs lga de

```
// For Displaying the Pairs
for (auto Pair : ans) {
  cout << "(" << Pair.first << ", " << Pair.second << ")";
}
```

↓

(a, b)

M-3

```
#include <iostream>
#include <vector>
#include <set>
#include <algorithm>
using namespace std;
int bs(vector<int>& nums, int start, int x) {
  int end = nums.size()-1;
  while (start <= end) {
    int mid = (start + end)/2;
    if (nums[mid] == x) {
      return mid;
    }
    else if (x > nums[mid]) {
      start = mid + 1;
    }
    else {
      end = mid - 1;
    }
  }
  return -1;
}
int findPairs(vector<int>& nums, int k) {
  sort(nums.begin(), nums.end());
  set<Pair<int, int>> ans;
  for(int i=0; i<nums.size(); i++) {
    if(bs(nums, i+1, nums[i]+k) != -1) {
      ans.insert({nums[i], nums[i]+k});
    }
  }
  return ans.size();
}
int main() {
  vector<int> nums = {3,1,4,1,5};
  int k = 2;
  int result = findPairs(nums, k);
  cout << result << endl;
  return 0;
}
```

i | 1 | 3 | 4 | 5

→ i + k i.e. 1+2 = 3 mil gaya ✓

(14) Find K-closest Element

i/p → arr = [1,2,3,4,5] , k=4 , x=3
o/p → [1,2,3,4]

i/p → arr = [1,2,3,4,5] , k=4 , x=-1
o/p → [1,2,3,4]

↳ M1 → • Sort with respect to difference and then take 'K' element into the

M2 → 12|16|22|30|35|39|42|45|48|50|53|55|56
diff → 23 17 13 5 0 4 7 10 13 15 18 20 21 K=4 x=25

L H

Kisi tak ka data hai du

```
#include <iostream>
#include <vector>
#include <algorithm>

using namespace std;
vector < int > findKclosedElement(vector<int>&arr,
                                  int K, int x) {
    int l=0;
    int h = arr.size()-1;
    while(h-l >=K) {
        if(x-arr[l] > arr[h]-x) {
            ++l;
        }
        else {
            --h;
        }
    }
    vector< int > ans;
    for(int i=l; i<=h; i++) {
        ans.push_back(arr[i]);
    }
    return ans;
}

int main() {
    vector<int> arr={1,2,3,4,5};
    int K=4;
    int x=3;
    vector<int> result = findKcloseElements(arr,K,x);
    for(int num: result) {
        cout << num;
    }
    return 0;
```

M-3 → more optimize

M-3 → 12|16|22|30|35|39|42|45|48|50|53|55|56

L H

Kisi Element ko L kii dege

ki ko high kii dege

```
while(K--) {
    if(x-arr[l] > arr[h-x]) {
        h++;
    }
    else {
        l--;
    }
}
```

(1,0)

15 EXPONENTIAL SEARCH

→ applicable for sorted array

```
int bs(int a[], int start, int end, int x) {
    while (start <= end) {
        int mid = (start + end)/2;
        if (a[mid] == x) {
            return mid;
        }
        else if (x > a[mid]) {
            start = mid+1;
        }
        else {
            end = mid-1;
        }
    }
    return -1;
}

int exp(int a[], int n, int x) {
    if (a[0] == x) {
        return 0;
    }
    int i = 1;
    while (i < n && a[i] <= x) {
        i = i*2;
    }
    return bs(a, i/2, min(i, n-1), x);
}
```

• We binary search so there are optimize bs dekh, we bolteh,

Pori ke binary search kaise se acche h bol ke there so but ke binary search karlete hu

```
int main() {
    int a[] = {3,4,5,6,11,13,14,15,56,70};
    int n = sizeof(a)/sizeof(int);
    int x = 13;
    int ans = expSearch(a,n,x);
    return 0;
}
```

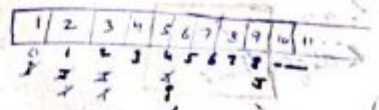
$\boxed{T.C \rightarrow O(\log \frac{\log n}{2})}$

Application:

① Search in ∞ array (called as unbound array)

• better than BS (if x is near begining)

⑤ Find the element in an ∞ array (sorted)



```
while (a[j] < x) {
    i = j;
    j = j*2;
}
```

Start = i
end = j → bs → binary search kar sakte h

• ab is search space me binary search kar sakte h

Search = i

(16) Book Allocation Problem (Daily)

Logic in Mind
Code to write
↳ Think twice, code once

```
# include <iostream>
# include <numeric>
# include <algorithm>

using namespace std;

bool isPossibleSolution(int a[], int n, int m, int sol){
    int PageSum = 0;
    int c = 1;    // Kitne student ko hum baggen.

    for(int i = 0; i < n; i++){
        if (a[i] > sol) {    // To book Main allocate Karna to sahi hai, no solution so bolle h?
            return false;
        }
        if (PageSum + a[i] > sol){
            c++;    // ab Next Student Ki baari h
            PageSum = a[i];
            if (c > m){    // c > m → Ye to sahi hi hamne choice
                return false;
            }
        }
        else {
            PageSum += a[i];    // sahi to EK hi student to dete Jao
        }
    }
    return true;
}

int findPages(int a[], int n, int m){
    if (n > m) {
        return -1;
    }
    int start = 0;
    int end = accumulate(a, a+n, 0);    // Yha sum nikala h bas Pure array Ka
    int ans = -1;
    while (start <= end) {
        int mid = (start + end)/2;
        if (isPossibleSolution(a, n, m, mid)) {
            ans = mid;
            end = mid - 1;
        }
        else {
            start = mid + 1;
        }
    }
    return ans;
}

int main(){
    int a[] = {12, 34, 67, 90};
    int n = sizeof(a)/sizeof(a[0]);
    int m = 2;
    int result = findPages(a, n, m);
    if (result != -1) {
        cout << "Max no. of Pages: " << result << endl;
    }
    else {
        cout << "Invalid Input" << endl;
    }
}
```

4 book → 12 | 34 | 67 | 90 (0, 1, 2, 3)

A(i) → no. of pages in that book
m → no. of students
n → no. of books

• allocation of books should be contiguous (aaa ke b ...)
↓
Suppose m = 2, so books ...
① 12 | 34 67 90 (max = 91)
② 12 34 | 67 90 (max = 101)
③ 12 34 67 | 90 (max = 113)
↓
we have to take Min (Max of all ...)
↓
ans → 113

• if (no. of students > no. of books) { return -1; }

• Thinking Style :-
① Main Search Space define Karta hu → agar ek hi book ko sab ... de,
• i.e. 0 ↔ sum of array
i = [0 – 203]

② 0 ——— 101 ——— 203
Min Max
mid = (0+203)/2 = 101
↓
try to allocate each student at most 101 Pages

I → 12+34+67 = 113
II → 67+90 = 157
III → 90
(EK Student Jaata so sahi h Main No. of pages Kaa Limit kaise decide ka)
∴ (101 is not Possible sol.) (m = 2 the i)

if (Not found) {
start = mid+1;
}

③ 102 ——— 151 ——— 203
I → 12+34+67
II → 90
if (Possible) {
end = mid-1;
}

④ 102 ——— 126 ——— 151
I → 12+34+67
II → 90
end = mid-1

⑤ 102 ——— 113 ——— 125
I → 12+34+67
II → 90

⑥ 102 ——— 107 ——— 112
I → 12+34
II → 67
III → 90

⑦ 102 ——— 104 ——— 107

⑧ 102 ——— 103 ——— 104

(17) Painters Partition Problem

→ Given are N boards of with length of each given in the form of array, and K painters, such that each painter takes 1 unit of time to paint 1 unit of the board. The task is to find the minimum time to paint all boards under the constraints that any painter will only paint continuous sections of boards, say board {2, 3, 4} or only board {1} or nothing but not board {2, 4, 5}

Dog house ke dog-dog board ko Paint Karne Ke Time Minimum Karna (contiguous allot)

[redacted] → N = 5

5	10	30	20	15

5 length, 1 unit of time

A[i] → length of ith board

K → Painters

↓

"1 unit of time to paint 1 unit board"

(cout << → Back Allocation Tracker)

```
# include <iostream>
# include <algorithm>

using namespace std;

bool isPossibleSolution(int a[], int n, int K, long long mid) {
    long long timeSum = 0;
    int c = 1;
    for (int i = 0; i < n; i++) {
        if (a[i] > mid) {
            return false;
        }
        if (a[i] + timeSum > mid) {
            c++;
            timeSum = a[i];
            if (c > K) {
                return false;
            }
        else {
            timeSum += a[i];
        }
    }
    return true;
}

long long minTime(int arr[], int n, int K) {
    long long start = 0;
    long long end = 0;
    for (int i = 0; i < n; i++) {
        end += arr[i];
    }
    long long ans = -1;
    while (start <= end) {
        long long mid = start + (end - start)/2;
        if (isPossibleSolution(arr, n, K, mid)) {
            ans = mid;
            end = mid - 1;
        }
        else {
            start = mid + 1;
        }
    }
    return ans;
}

int main() {
    int arr[] = {5, 10, 30, 20, 15};
    int n = sizeof(arr) / sizeof(arr[0]);
    int K = 3;
    long long result = minTime(arr, n, K);
    if (result != -1) {
        cout << " Min. time required " << result;
    }
    else {
        cout << " Invalid Input " << endl;
    }
```

EXACT SAME

(18) AGGRESIVE COWS

N1 → O(n²)
N2 → O(n)

Q - You are given an array consisting of n integers which represent the position of a stall. You are also given an integer K which denotes the number of aggressive cows. You are given the task of assigning stalls to K cows such that the minimum distance between any two of them is the maximum possible.

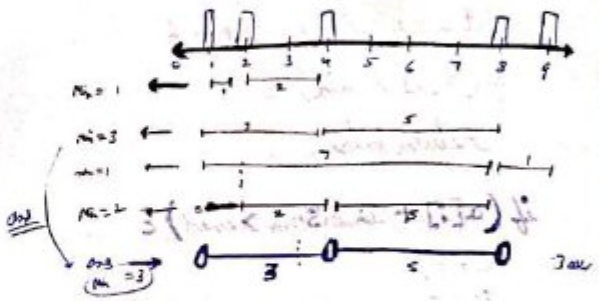
i/p → n=5
K=3
stalls = [1 2 4 8 9] ← Position

o/p → 3

```
# include <iostream>
# include <vector>
# include <algorithm>
using namespace std;
```

(can we place K cows, with at least mid distance b/w cows)

```
bool isPossibleSolution(vector<int>& stalls, int K, int mid) {
    int c = 1;
    int Pos = stalls[0]; // Pehle cow ko 1st index pe stall dia
    for (int i = 1; i < stalls.size(); i++) {
        if (stalls[i] - Pos >= mid) {
            c++;
            Pos = stalls[i];
        }
    }
    if (c == K) {
        return true;
    }
}
return false;
}

int Solve(int n, int K, vector<int>& stalls) {
    sort(stalls.begin(), stalls.end());
    int start = 0;
    int end = stalls[stalls.size()-1] - stalls[0];
    int ans = -1;
    while (start <= end) {
        int mid = start + (end - start)/2;
        if (isPossibleSolution(stalls, K, mid)) {
            ans = mid;
            start = mid + 1;
        }
        else {
            end = mid - 1;
        }
    }
    return ans;
}

int main() {
    int n = 5;
    int K = 3;
    vector<int> stalls = {1, 2, 4, 8, 9}
    int result = Solve(n, K, stalls);
    if (result != -1) {
        cout << "largest min. distance:" << result << endl;
    }
    else {
        cout << "No Valid Solution found" << endl;
    }
```



• (Sort karlo)

Min dist = 1 — ka hasil
Min dist = 2 ✓
Min dist = 3 ✓
Min dist = 4 ✗ uski Plan Kar Lage

(iske aage Sort ki zaruraat nahi)

start end
0 — (Max Pos − Min Pos)
0 — (9 − 1)
[0 — 8] Search Space
4 → mid
↳ at least 4 distance bet cows ka setting

19 Eko SPOJ → RELEASE MEE IT'S VERY EASY

i/p → 4 7
20 15 10 17

o/p → 15

→ I need 7 meter of tree (min^m)

• Jab 5 m ke cut kiye diye to total tree mile → 15+10+3+10 = 42 m ✓

• Jab 15 m ke cut kiye to → 5+2 = 7 ✓ (ha ye sahi h)

```
# include <iostream>
# include <vector>
# include <algorithm>
using namespace std;
bool isPossibleSolution(vector< long long int > trees,
                        long long int m,
                        long long int mid) {
    long long int woodCollected = 0;
    for (long long int i=0; i < trees.size(); i++) {
        if ( trees[i] > mid) {
            woodCollected += trees[i] - mid;
        }
    }
    return woodCollected >= m;
}
long long int maxSawBladeHeight(vector< long long int > trees,
                                long long int m) {
    long long int start = 0, end, ans = -1;
    end = *max_element(trees.begin(), trees.end());
    while ( start <= end) {
        long long int mid = start + (end-start)/2;
        if ( isPossibleSolution(trees, m, mid)) {
            ans = mid;
            start = mid+1;
        }
        else {
            end = mid-1;
        }
    }
    return ans;
}
int main() {
    long long int n, m;
    cin >> n >> m;
    vector< long long int > trees;
    while (n--) {
        long long int height;
        cin >> height;
        trees.push_back(height);
    }
    cout << maxSawBladeHeight(trees, m) << endl;
    return 0;
}
```

if (woodCollected >= m)
return true
else {
return false
}

0 — 20 → Search space

mid=10 ← Ans the height

20 PRATA SOJ → Know Logic Code Self

```
# include < iostream >
# include < vector >
# include < algorithm >
using namespace std;
bool isPossibleSolution(vector< int > cookerranks, int np, int mid) {
    int currp = 0;
    for (int i = 0; i < cookerranks.size(); i++) {
        int r = cookerranks[i], j = 1;
        int timetaken = 0;
        while( true) {
            if ( timetaken + j * r <= mid) {
                ++ currp;
                timetaken += j * r;
                ++ j;
            }
            else {
                break;
            }
            if (currp >= np) {
                return true;
            }
        }
        return false;
    }
int minTimetoCompletekorder (vector< int> cookerranks, int np) {
    int start = 0;
    int highestrank = *max_element( cookerranks.begin(), cookerranks.end());
    int end = highestrank * (np* (np+1)/2);
    int ans = -1;
    while (start <= end) {
        int mid = ( start + end)/2;
        if ( isPossibleSolution (cookerranks, np, mid)) {
            ans = mid;
            end = mid - 1;
        }
        else {
            start = mid + 1;
        }
    }
    return ans;
}
int main() {
    int t;
    cin >> t;
    while (t --) {
        int np, nc;
        cin >> np >> nc ;
        vector < int > cookerranks ;
        while (nc--) {
            int r;
            cin >> r;
            cookerranks. push_back (r);
        }
        cout << minTimetoCompletekorder (cookerranks, np) << endl;
    }
```

Min time → 0

Max time (if highest rank) (to sarg parathe de dia) → 4(1+2+3+...10) = 220 min

220 (highest)

0 — 220

110 (mid)

Search space

Binary space 0–220 ek ek karke dekhenge

Lon wala wali order of 10 Paratha in 110 min?

C1 → 1+2+3+...10 = 55 Min (ye to hai hi bna diya)

C1 C2 C3 C4

for 1st lon eg:-

4 × (10 × (10+1)/2) = 220

and → highest rank × (np × (np+1)/2)

0 — 109

54

9 Paratha by C1 → 45 min (1+2+...9)

1 Paratha by C2 → 2 min (2)

0 — 53

26

6 Paratha by C1 → 21

4 Paratha by C2 → 20 min (2+4+6+8)

(they will finish)

end = 4

0 — 25

12

4 Paratha by C1 (10 Min)

3 Paratha by C2 (12 M)

2 Paratha by C3 (9 M)

1 Paratha by C4 (4 Min)

0 — 11

5

(Not Possible)

start = mid

7 → 9 → 11 ×

3 - char Array and String

- char ch[100];
  cin >> ch // No need for '&' for 'char', ch is itself the name [illegible]
- cin >> ch[2]; // for specific
- strcmp // for cmp

- char ch[100];
  cin.getline(ch, 50); // Isme Pure Anish Kumar Sinha likh sakte h
  // Max 50 length tk dt sakte h

★ Questions

(inbuilt function → strlen(name) under #include <cstring>

① Find the length of string

```
int getlength(char name[100]) {
    int length = 0;
    int i = 0;
    while (name[i] != '\0') {
        length++;
        i++;
    }
    return length;
}
```

② Reverse the string i/p → ANISH, o/p → HSINA } → [illegible] cin.getline

```
void reverse(char ch[100]) {
    int n = strlen(ch);
    int i = 0;
    int j = n-1;   // using 2 pointer approach
    while (i <= j) {
        swap(ch[i], ch[j]);
        i++;
        j--;
    }
}
```

③ Replace all space with '@'

```
char ch[100];
cin.getline(ch, 50);
int n = strlen(ch);
for (int i=0; i<n; i++) {
    if (ch[i] == ' ') {
        ch[i] = '@';
    }
}
```

```
#include <iostream>
#include <string>
using namespace std;
void space (string $s) {
    for (int i = 0; i < s.length(); i++) {
        if (s[i] == ' ') {
            s[i] = '@';
        }
    }
}
int main () {
    string s;
    getline (cin, s);
    space (s);
    cout << s;
    return 0;
}
```

④ Palindrome check → • M-1 → reverse nikal ke check kro length (sc = n, SC = n)
• M-2 → Use 2 pointer approach

```
bool find (char ch[100]) {
    int i = 0;
    int j = strlen(ch) - 1;
    while (i <= j) {
        if (ch[i] == ch[j]) {
            i++;
            j--;
        }
        else {
            return false;
        }
    }
}
```

⑤ Convert to upper case i/p → anish, o/p → ANISH

```
void ConvertToUpper (char ch[100]) {
    int n = strlen(ch);
    for (int i=0; i<n; i++) {
        ch[i] = ch[i] - 'a' + 'A';   // a = 97, A = 65
    }
}
```

START

• # include < string >

for char use → String str; cin >> str;

for string use → String str; getline(cin, str);

★ ALL important Library/inbuilt function

```
# include < iostream >
# include < string >
using namespace std;

int main() {
    string str = "ANISHSINHA";
    cout << str.length(); // str.size()
    cout << str.empty();
    str.Push_back('A');
    cout << str;
    str.Pop_back();
    cout << str;
    cout << str.substr(0, 5); → ANISH
```

start index, total length

str = "Anish"

str2 = "Sinha"

for append

str = str + str2 → AnishSinha

str.append("str2")

• checking 2 strings are equal or not

```
bool compare(string a, string b) {
    if (a.length() != b.length()) {
        return false;
    }
    else {
        for (int i = 0; i < a.length(); i++) {
            if (a[i] != b[i]) {
                return false;
            }
        }
    }
    return true;
}
```

```
String a = 'love';
String b = 'lover';
    if (a.compare(b) == 0) {
        cout << "Strings are equal";
    }
    else {
        cout << "strings are not equal";
    }
```

```
String d = "abcd";
String e = "bcda";
cout << d.compare(e);   // 0, -1, 1
```

```
String sentence = "I am Anish";
String target = "am";
cout << sentence.find(target);   // gives 1st index
```

→ Ulta to valid index aaiyegi

→ Ulta to npos (no location)

```
String name = "anish is love";
String word = "coding";
name.replace(0, 5, word);
cout << name;
```

```
String hay = "abcdefghijklmnopqrstuvwxyz";
    hay.erase(0, 4);
    cout << hay;
```

start, how many characters

'q'

★ length, empty → Push/Pop, Substr

compare

find, replace, erase

free

(S) Substr, compare, find, replace, erase

⑥ Remove all adjacent Duplicates in a string i/p → caabbaca
o/p → a

```
String removeConsecutiveDuplicates (string s) {
    string ans = " ";
    int i = 0;
    while ( i < s.length() ) {
        if (ans.length() > 0) {
            if ( ans[ans.length()-1] == s[i]) {
                ans.Pop_back();
            }
            else {
                ans.Push_back(s[i]);
            }
        }
        else {
            ans.Push_back(s[i]);
        }
        i++;
    }
    return ans;
}
```

→ To check to see if the 'ans' string is not empty.

→ If 'ans' is not empty, it checks whether the last character of 'ans' is equal to the current character 's[i]' in the input string. If they are the same (including consecutive duplicates), it removes the last character from 'ans' using "pop_back()". otherwise, it appends the current character 's[i]' to the 'ans' string.

→ If 'ans' is initially empty, it directly appends the current character 's[i]' to it

(1st time always we will be here)

⑦ Remove all occurence of a string i/p → [abcdcbdeabc] , part → [abc]
o/p → [dcde]

i/p → [dacbcbacbc] , part → [abc]
o/p → [dab]

```
String removeSubstring (string s, string Part) {
    int pos = s.find (Part);
    while (Pos != string :: npos) {
        s.erase (Pos, Part.length());
        Pos = s.find (Part);
    }
    return s;
}
```

→ If it finds a match, it returns the position (index) of the first character of the found substring; otherwise, it returns 'string :: npos' (a constant representing "no position")

(Find the position, the position is equal to zero then ...)

⑧ Palindrome-2 i/p → s = abca
o/p → Yes, removed char → a (abca → aba)

```
bool checkPalindrome( string s, int i, int j) {
    while (i <= j) {
        if (s[i] != s[j]) {
            return false;
        }
        i++;
        j--;
    }
    return true;
}
bool canMakePalindrome (string s) {
    int i = 0;
    int j = s.length()-1;
    while (i <= j) {
        if (s[i] != s[j]) {
            return checkPalindrome (s, i+1, j) || checkPalindrome (s, i, j-1);
        }
        else {
            i++;
            j--;
```

⑨ Minimum Time Difference

12:10	10:25	13:35	17:20	15:00	19:47	23:59

Minimum minute difference?

- Sort Minutes ranks
- then sort
  - then find min
  - take care of special case

```
int findMinTimeDifference(vector<string>& timepoints) {
    // Step1: convert time strings into minutes integer value
    vector<int> minutes;
    for(int i=0; i< timepoints.size(); i++) {
        string curr = timepoints[i];
        int hours = stoi(curr.substr(0,2));
        int mins = stoi(curr.substr(3,2));
        int totalMinutes = hours*60 + mins;
        minutes.push_back(totalMinutes);
    }
    // Step2: sort
        sort(minutes.begin(), minutes.end());
    // Step3: Calculate the minimum time difference
    int mini = INT_MAX;
    int n = minutes.size();
    for(int i=0; i<n-1; i++) {
        int diff = minutes[i+1] - minutes[i];
        mini = min(mini, diff);
    }
    // last special case
    int lastDiff = (minutes[0]+1440) - minutes[n-1];
    mini = min(mini, lastDiff);
    return mini;
}
```

- stoi → string to integer
- substr(0,2) → starting idx, add characters

```
unordered_map<char, int> hash;
for(auto ch:s) {
    hash[ch]++;
}
for(auto ch:t) {
    hash[ch]--;
}
for(auto ch:s) {
    if (hash[ch]!=0) {
        return false;
    }
}
return true;
```

23:59 | 00:00 → 1439 | 0 → diff = 1439 (takin\ was min)

Sort → 0 | 1439

1440 − 1439 = 1

⑩ Palindromic Substrings

```
int expandAroundIndex(string s, int left, int right) {
    int count = 0;
    while (left >=0 && right < s.length() && s[left] == s[right]) {
        count ++;
        left --;
        right ++;
    }
    return count;
}
int countSubstrings(string s) {
    int count = 0;
    int n = s.length();
    for(int i=0; i<n; i++) {
        // odd
        int oddKaAns = expandAroundIndex(s, i, i);
        count += oddKaAns;
        // even
        int evenKaAns = expandAroundIndex(s, i, i+1);
        count += evenKaAns;
    }
    return count;
}
```

odd Substring / Even Substring

n	o	o	n

4 + 2 → 6

Substring will be two and delete 1 one palindrome hya nhi?

$O(n^2)$

i/p → [a, b, c]

Substring (return) → a ✓ (palin) ab ✗ abc ✗ b ✓ bc ✗ c ✓

o/p → a, b, c → 3 ✓

i/p → [a, a, a]

o/p → a, aa, aaa → 3

n−1 → • find all substring $O(n^2)$ • check palindrome & count $O(n)$

More Questions

⑪ Valid Anagram

→ i/p → s = "anagram", t = "nagaram"
o/p → True

Sort(s.begin(), s.end())

• M-1 → Sort both & then check

• M-2 → Counting ++ for s & count -- for t → 0 → True

Make frequency Table

```
bool areAnagrams(string s, string t) {
    int freqTable[256] = {0}; // Initialize the frequency table with zeros.
    // count the frequency of characters in string s
    for(int i=0; i<s.size(); i++) {
        freqTable[s[i]]++;
    }
    // Decrement the frequency for characters in string t
    for(int i=0; i<t.size(); i++) {
        freqTable[t[i]]--;
    }
    // check if the frequency table contains only zeros, indicating anagrams
    for(int i=0; i<256; i++) {
        if(freqTable[i]!=0) {
            return false;
        }
    }
    return true;
}
```

⑫ Reverse Only

→ i/p → s = "a-bC-dEf-ghIj"
o/p → "j-Ih-gfE-dCba" (use 2 pointers)

```
bool isAlphabetic(char c) {
    if(c>='a' && c<='z') {
        return true;
    }
    if(c>='A' && c<='Z') {
        return true;
    }
    else {
        return false;
    }
}
string reverseAlphabetic(string s) {
    int l=0;
    int h = s.size()-1;
    while(l<h) {
        if(isAlphabetic(s[l]) && isAlphabetic(s[h])) {
            Swap(s[l], s[h]);
            l++;
            h--;
        }
        else if(!isAlphabetic(s[l])) {
            l++;
        }
        else {
            h--;
        }
    }
    return s;
}
```

(23) use Condition → isAlpha(s[l]) → under isalpha < cctype >

SAME

⑬ Longest Common Prefix

→ i/p → ["flower", "flow", "flight"]
o/p → fl

```
string longestCommonPrefix(vector<string> str) {
    string ans;
    int i=0;
    while(true) {
        char curr_ch=0;
        for(int j=0; j<str.size(); j++) {
            if(i>=str[j].size()) {
                // out of bound
                curr_ch=0;
                break;
            }
            if(curr_ch==0) {
                curr_ch = str[j][i];
            }
            else if(str[j][i] != curr_ch) {
                curr_ch=0;
                break;
            }
        }
        if(curr_ch==0) {
            break;
        }
        ans.push_back(curr_ch);
        i++;
    }
    return ans;
}
```

→ → last Page of Chapter

⑭ Reverse Vowels of a string

→ i/p → Hello
→ o/p → Holle
→ • use 2 pointer with some condition

```
bool isVowel(char ch) {
    ch = tolower(ch);
    return(ch=='a'||ch=='e'||ch=='i'||ch=='o'||ch==
}
String reverseVowels(string s) {
    int l=0;
    int h = s.size()-1;
    while(l<h) {
        if(isVowel(s[l]) && isVowel(s[h])) {
            swap(s[l], s[h]);
            l++;
            h--;
        }
        else if(!isVowel(s[l])) {
            l++;
        }
        else {
            h--;
        }
    }
    return s;
}
```

15 Isomorphic Strings

i/p → s = "egg", t = "add"
o/p → True

i/p → s = "foo", t = "bar"
o/p → False

Constraint → t.length == s.length

• M-1 → WITH USE OF HASH

```
bool areMappedStrings(string s, string t) {
    int hash[256] = {0};  // Mapping of each character of string 's' to string 't'
    bool isCharsMapped[256] = {0};  // stores if character t[i] is already mapped with s[i]

    for(int i = 0; i < s.size(); i++) {
        if (hash[s[i]] == 0 && !isCharsMapped[t[i]]) {
            hash[s[i]] = t[i];
            isCharsMapped[t[i]] = true;
        }
    }

    for(int i=0; i < s.size(); i++) {
        if(char(hash[s[i]]) != t[i]) {
            return false;
        }
    }
    return true;
}
```

• M-2 → WITHOUT HASH

```
bool areMappedStrings(string s, string t) {
    bool isCharsMapped[256] = {0};
    for(int i=0; i < s.size(); i++) {
        if(! isCharsMapped[t[i]]) {
            isCharsMapped[t[i]] = true;
        }
        else if(s[i] != t[i]) {
            return false;
        }
    }
    return true;
}
```

16 Reorganize String

→ Rearrange the characters of s so that any two adjacent characters are not same.

→ M-1 → Priority Queue
M-2 → Directly Method

i/p → s = "aab"
o/p → "aba"

i/p → s = "aaab"
o/p → ""

```
String reorganizeString(string s) {
    int hash[26] = {0};
    for(int i=0; i < s.size(); i++) {
        hash[s[i] - 'a']++;
    }

    // find most frequent char
    char max_freq_char;
    int max_freq = INT_MIN;
    for(int i=0; i < 26; i++) {
        if(hash[i] > max_freq) {
            max_freq = hash[i];
            max_freq_char = i + 'a';
    }
    }

    int index = 0;
    while (max_freq > 0 && index < s.size()) {
        s[index] = max_freq_char;
        max_freq--;
        index += 2;
    }

    if(max_freq != 0) {
        return "";
    }

    hash[max_freq_char - 'a'] = 0;
    // Place the rest of the characters
    for(int i=0; i < 26; i++) {
        while (hash[i] > 0) {
            index = index >= s.size() ? 1 : index;
            // if index is out of bounds, set it to 1
            s[index] = i + 'a';
            hash[i]--;
            index += 2;
    }
    }
    return s;
}
```

aaabbcf

aaab

(17) Group Anagrams

i/p → ["eat", "tea", "tan", "ate", "nat", "bat"]

o/p → [["bat"], ["nat", "tan"], ["ate", "eat", "tea"]]

→ Finish karo code SELF

(No sorting)

• M-1

```
# include < iostream >
# include < string >
# include < vector >
# include < unordered_map >
# include < algorithm >
using namespace std;
vector < vector < string >> groupAnagrams (vector < string > & strs) {
    unordered_map < string , vector < string >> mp;
    for ( auto str : strs) {
        string s = str;
        sort (s.begin(), s.end())
        mp [s] . Push_back( str);
    }
    vector < vector < string >> ans;
    for (auto it = mp. begin (); it != mp. end(); it ++) {
        ans . Push_back( it -> second);
    }
    return ans;
}
int main () {
    vector < string > strs = { "eat", "tea", "tan", "ate", "nat", "bat"};
    vector < vector < string >> groupedAnagrams = groupAnagrams (strs);
    cout << "Grouped Anagrams" << endl;
    for (const auto & group : groupedAnagrams) {
        for (const string & str : group) {
            cout << str << " ";
        }
        cout << endl;
    }
```

• all sort

aet, aet, ant, aet, ant, abt

group1 group2 group3

Key	Value
① aet	eat, tea, ate
② ant	tan, nat
③ abt	bat

vector < Pair < string, vector < string >> >

[Iske corresponding ek entry bna di]

traversing the map

```
for (auto i : mp)
    ans . Push_back (i . second);
```

• M-2

a, b dono ka hash bna do, Jisa sum equal aaye to wo anagram hoga

• Sort string ka sort karke,

eat

str ⑤ → eat, aet, ant, aet, ant, abt

gr gr gr

Map	First (Key)	Second (Value)
	String	Vector
it	aet	eat, tea

18 Longest Palindromic sub-string

i/p → bataad
o/p → bab or aba

↳ • M-1 → DP ✓
• M-2 → Brute force (m²)

```
bool isPalindrome(string &s, int start, int end){
    while (start <= end){
        if (s[start] != s[end]){
            return false;
        }
        start++;
        end--;
    }
    return true;
}

string longestPalindrome(string s){
    string ans = "";
    for(int i=0; i<s.size(); i++){
        for(int j=i; j<s.size(); j++){
            if (isPalindrome(s, i, j)){
                string t = s.substr(i, j-i+1);
                ans = t.size() > ans.size() ? t : ans;
            }
        }
    }
    return ans;
}
```

substring with loop

19 Find the index of first occurence in a string

(M-1 → sliding window (hard))

i/p → haystack = "sadbutsad"
needle = "sad"
o/p → 0

→ 2 loop

```
int str(string haystack, string needle){
    int n = haystack.size();
    int m = needle.size();

    for(int i=0; i<=n-m; i++){
        int j;
        for(j=0; j<m; j++){
            if (needle[j] != haystack[i+j]){
                break;
            }
        }
        if (j==m){
            return i;
        }
    }
    return -1;
}
```

20 String to Integer

→ int integer = stoi(s.c_str());

→ i/p → "42"
o/p → 42

```
int mystoi(string s){
    int num = 0, i=0, sign=1;
    // skip leading space
    while (i<s.size() && s[i]==' '){
        i++;
    }
    // check for a sign
    if (i<s.size() && (s[i]=='-' || s[i]=='+'))
        sign = (s[i]=='+') ? 1 : -1;
        i++;
    }
    //convert digits(char)
    while (i<s.size() && isdigit(s[i])){
        int digit = s[i] - '0';  (String to Integer)
        // check for integer overflow
        if (num > INT_MAX/10 || (num == INT_MAX/10 && digit > 7)){
            return (sign == -1) ? INT_MIN : INT_M
        }
        num = num*10 + digit;
        i++;
    }
    return num * sign;
}
```

21 String Compression

i/p → chars = ["a","a","b","b","c","c","c"]
o/p → 6 → [a2b2c3] (in place)

```
#include <iostream>
#include <string>
#include <algorithm>
using namespace std;
int compressString(string &s){
    int index = 0, count = 1;
    char prev = s[0];
    for(int i=1; i<s.size(); ++i){
        if (s[i] == prev){
            count++;
        }
        else {
            s[index++] = prev;
            if(count > 1){
                int start = index;
                while(count){
                    s[index++] = (count%10) + '0';
                    count /= 10;
                }
                reverse(s.begin() + start, s.begin() + index);
            }
            prev = s[i];
            count = 1;
        }
    }
    s[index++] = prev;
    if (count > 1){
        while (count){
            s[index++] = (count%10) + '0';
            count /= 10;
        }
    }
    return index;
}
```

```
int main(){
    string s = "aaabbcdddeee";
    int newlength = compressString(s);
    cout << s.substr(0, newlength);
    return 0;
}
```

index = 0, count = 1
if (s[i] == prev){
    count++;
}
else {
    s[index] = prev;
    if (count > 1){
        s[index] = count;
    }
}
count = 1 // reset

HOW I do
SELF CODE

→ → last 14

22 Integer to Roman

→ i/p → 3
→ o/p → III

```
string intToRoman(int num){
    string romanSymbols[]={"M","CM","D","CD","C","XC","L","XL","X","IX","V","IV","I"};
    int values[] = {1000,900,500,400,100,90,50,40,10,9,5,4,1};

    string ans = " ";

    for(int i = 0; i<13; i++){
        while(num >= values[i]){
            ans += romanSymbols[i];
            num -= values[i];
        }
    }
    return ans;
}
```

23 Zig-Zag Conversion

i/p → Str = ABCDEFGHIJ , numRows = 3

output →

AEIBDFHJCG (output)

① numRows = 1 → return original string
② change direction after Sometime → • Top - Bottom • Bottom - Top & Top - Bottom

Logic ✓ CODE SELF

```
#include <iostream>
#include <vector>
using namespace std;
string convertToZigZag(string s, int numRows){
    if(numRows == 1 || s.length() <= numRows){
        return s;
    }
    vector<string> zigzag(numRows);
    int row = 0;
    bool down = true;
    for(char ch : s){
        zigzag[row] += ch;
        if(row == 0){
            down = true;
        }
        else if(row == numRows-1){
            down = false;
        }
        down ? ++row : --row;
    }
    string result;
    for(const string & str : zigzag){
        result += str;
    }
    return result;
}
```

```
string convert(string s, int numRows){
    if(numRows==1){
        return s;
    }
    vector<string> zigzag(numRows);
    int i=0, row=0;
    bool direction = true;
    while(true){
        if(direction){
            while(row<numRows && i<s.size()){
                zigzag[row++].push_back(s[i++]);
            }
            row = numRows-2;
        }
        else{
            while(row>=0 && i<s.size()){
                zigzag[row--].push_back(s[i++]);
            }
            row = 1;
            if(i>=s.size()){
                break;
            }
            direction = !direction;
        }
    }
    string ans = "";
    for(int i=0; i<zigzag.size(); i++){
        ans += zigzag[i];
    }
    return ans;
}
```

24 Largest Number

i/p → nums = [10, 2]
o/p → "210"

custom comparator use hoga

```
static bool myComp(string a, string b){
    string t1 = a+b;   // 330
    string t2 = b+a;   // 303
    return t1 > t2;
}

string largestNumber(vector<int> & nums){
    vector<string> snums;
    for(auto n:nums){
        snums.push_back(to_string(n));
    }
    sort(snums.begin(), snums.end(), myComp);
    if(snums[0] == "0") return "0";
    string ans = "";
    for(auto str: snums){
        ans += str;
    }
    return ans;
}
```

Longest Common Prefix

```
String longestCommonPrefix(vector< string>& strs) {
    String ans = " ";
    // loop on first string
    for( int i = 0; i< strs[0]. length(); i++) {
        char ch = strs[0][i];
        bool match = true;
    // Compare this character with all
    // The remaining strings at the same index
    for(int j = 1; j< strs. size(); j++) {
        // Compare
        if(strs[j]. size() < i || ch != strs[j][i]){
            match = false;
            break;
        }
    }
    if (match == false)
        break;
    else
        ans. Push_back(ch);
    }
    return ans;
}
```

```
String mostCommonWord (String paragraph, vector < string > & banned) {
    Unordered_set < string > bannedSet (banned. begin(), banned. end());
    Unordered_map < String , int > wordCount;
// convert paragraph to lowercase and replace Punctuation with spaces
    for (char & c: paragraph) {
        if( ispunct(c)) {
            c = ' ';
        }
        else {
            c = tolower (c);
        }
    }
// Split the paragraph into words
    istringstream stream (paragraph);
    String word;
    while (stream >> word) {
        if (bannedSet. find(word) == bannedSet. end()) {
            wordCount [word]++;
        }
    }
// find the most frequent non-banned word
    String mostFrequentWord;
    int maxCount = 0;
    for (auto entry: wordCount) {
        if (entry. second > maxCount) {
            mostFrequentWord = entry. first;
            maxCount = entry. second;
        }
    }
    return mostFrequentWord;
}
```

```
String Solve (String paragraph, vector < string > &b
    String s = " ";
    Unordered_map < String , int > hash;

for (int i = 0; i < paragraph. size(); i++) {
    if (isalpha(paragraph[i]) {
        s. Push_back (tolower (paragraph[i]));
    }
    else {
        s. Push_back(' ');
    }
}

String word = " ";
for (int i = 0; i <= s. size (); i++) {
    if ( isalpha (s[i])) {
        word = word + s[i];
    }
    else if ( ! word. empty()) {
        hash [word]++;
        word = " ";
    }
}

for (auto i : banned) {
    auto it = hash. find (i);
    if (it != hash. end()) {
        hash. erase (it);
    }
}

pair < String, int > maxi = {" ", 0};
for (auto i : hash) {
    if (i. second > maxi. second){
        maxi. second = i. second;
        maxi. first = i. first;
    }
}
return maxi. first;
```

4 - RECURSION

Questions

① Factorial

```
int factorial (int n) {
  // base case
  if(n==1) {
    return 1;
  }
  int choteProblemAns = factorial(n-1);
  int badeProblemAns = n* choteProblemAns;
  return badeProblemAns;
}
```

// Main bas ek case solve Karunga, baki Recursion Sambhal lega.

② Fibonacci Series

```
int fib(int n) {
  // base case
  // 1-st No. → 0
  if (n==1) {
    return 0;
  }
  // 2nd No. → 1
  if (n==2) {
    return 1;
  }
  int ans = fib(n-1) + fib(n-2);
  return ans;
```

③ Climbing Stairs

```
int climbStair (int n) {
  // base case
  if(n==0 || n==1) {
    return 1;
  }
  int ans = climbStair(n-1) + climbStair(n-2);
  return ans;
}
```

// agar ek dam niche khada hu to wha Pahuchne ka kitna tariqa hai
→ 1 tareeqa main khud ko wahi aaunga, ya agar 1st stair par hu to wha Pahuchne ka bhi 1 tariqa,

④ Print the element of array using Recursion

```
void Print (int arr[], int n, int i) {
  // base case → array ke bahar na jaye i
  if(i>=n) {
    return;
  }
  // 1 case solve Karliya
  cout << arr[i];
  // baki recursion sambhal lega
  Print(arr, n, i+1);
}
```

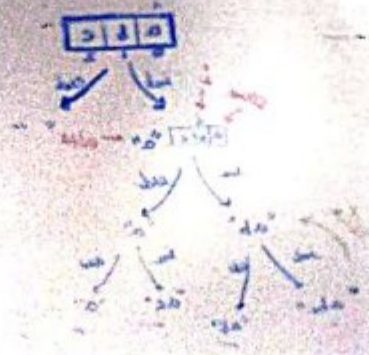
⑤ Find maximum of array using Recursion

```
void findmax(int arr[], int n, int i, int maxi)
  // base case
  if(i>=n) {
    return;
  }
  // 1 case Solve Karlunga
  if (arr[i] > maxi) {
    maxi = arr[i];
  }
  // baki Recursion sambhal lega
  findmax(arr, n, i+1, maxi);
}
```

⑥ Key Present or not

→ i/p → string = "Lovebabbar", Key = 'a'
o/p → Yes

```
bool checkKey (string str, int i, int n, char key)
  // base case
  if(i>=n) {
    return false;
  }
  // 1 case main solve Karunga
  if(str[i] == key) {
    return true;
  }
  // baki recursion sambhal lega
  return checkKey(str, i+1, n, key)
```



⑦ Print All Digit of the Number (using Recursion)

```
Void PrintDigits (int n) {
  // base case
  if (n==0) {
    return;
  }

int newValueofN = n/10;
// baki recursion sambhal liga
PrintDigits (newValueofN);
// I case main solve Karunga
  int digit = n%10;
  cout << digit ;
```

Direct first call

Reverse chahiye tha isse - liye se line upper Niche Karni h

⑧ Check the array is sorted or not (using Recursion)

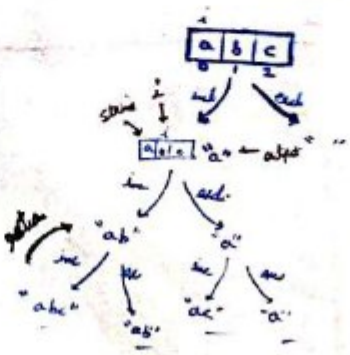
```
bool checkSorted (vector <int> $arr, int $n, int i) {
  // base case
  if (i==n-1) {
    return true;
  }
  // I-case main sambhal lunga
  if (arr[i+1] < arr[i]) {
    return false;
  }
  // baki recursion sambhal lega
  return checkSorted (arr, n, i+1);
}
```

⑨ Binary search using Recursion

```
int binarySearch (vector<int> arr, int s,
                  int e, int Key) {
  // base case
  // Case 1 → Key not found
  if (s>e) {
    return -1;
  }

  int mid = (s+e)/2;
  // Case 2 → Key found
  if (arr[mid] == Key) {
    return mid;
  }
  // arr[mid] < Key → right me search Karo.
  if (arr[mid] < Key) {
    return binarySearch (arr, mid+1, e, K
  }
  else {
    return binarySearch (arr, s, mid-1,
  }
```

⑩ Print all subsequence of string using Recursion

```
Void PrintSubsequence (string str, string output, int i) {
  // base case
  if (i>=str.length()) {
    cout << output;
    return;
  }
  // exclude
  PrintSubsequence (str, output, i+1);
  // include
  output.Push_back (str[i]);
  PrintSubsequence (str, output, i+1);
```



→ (sab ko Print karo)

(11) Min^m no of elements required to get target.

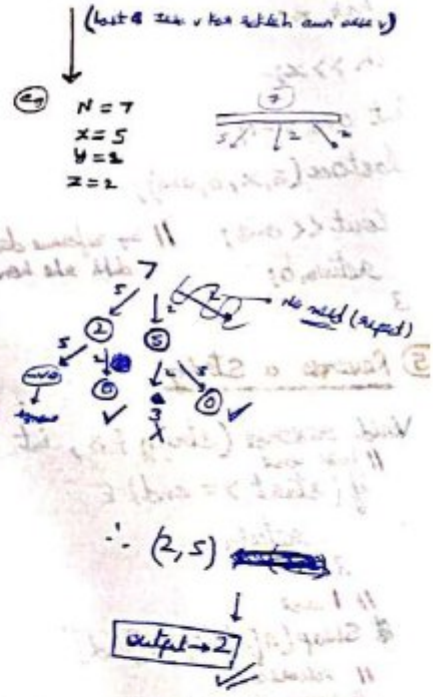
```
int solve(vector<int>$ arr, int target) {
    //base case
    if (target == 0) {
        return 0;
    }
    if (target < 0) {
        return INT_MAX;
    }
    // solve 1 case
    int mini = INT_MAX;
    for (int i=0; i < arr.size(); i++) {
        int ans = solve(arr, target - arr[i]);
        if (ans != INT_MAX) {
            mini = min(mini, ans+1);
        }
    }
    return mini;
}
```

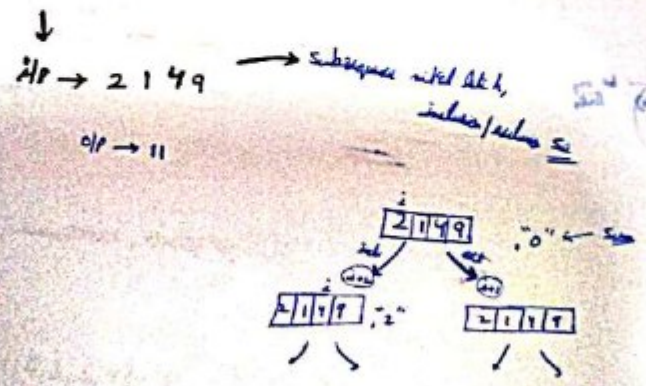
(12) Cut into Segments

```
int Solve(int n, int x, int y, int z) {
    // base case
    if (n == 0) {
        return 0;
    }
    if (n < 0) {
        return INT_MIN;
    }
    int ans1 = Solve(n-x, x, y, z)+1;
    int ans2 = Solve(n-y, x, y, z)+1;
    int ans3 = Solve(n-z, x, y, z)+1;
    int ans = max(ans1, max(ans2, ans3));
    return ans;
}
```

(13) Maximum Sum of Non-Adjacent Elements (same as House Robber Problem)

```
int solve(vector<int> arr, int i, int sum, int $ maxi) {
    //base case
    if (i >= arr.size()) {
        // maxi update
        maxi = max(sum, maxi);
        return;
    }
    // include
    Solve(arr, i+2, sum + arr[i], maxi);
    // exclude
    Solve(arr, i+1, sum, maxi);
}
```





★ More Questions

(14) Last Occurence of a char

→ String → "abcddefg", char → 'd'
ans → 6

L to R

↳ M-1/2 → Left/Right se EK EK
Reverse loop
check loop

M3 → STL fun → strchr() - to get the last occurence of the char

```
void lastoc(string &s, char x, int i, int &ans) {
    // base case
    if (i >= s.size()) {
        return;
    }
    // 1 case
    if (s[i] == x) {
        ans = i;
    }
    // recursion sambhal lega
    lastoc(s, x, i+1, ans);
}
int main() {
    string s;
    cin >> s;
    char x;
    cin >> x;
    int ans = -1;
    lastoc(s, x, 0, ans);
    cout << ans;   // by reference done ka fayda dekh ale hera
    return 0;
}
```

(15) Reverse a String

```
void reverse(string &s, int start, int end) {
    // base case
    if (start >= end) {
        return;
    }
    // 1 case
    swap(s[start], s[end]);
    // recursion
    reverse(s, start+1, end-1);
}
```

(16) Add Strings

→ I/P → "121" + "33"
O/P → 154

without use of any built-in library

Not allowed to convert string to integer directly

(17) Palindrome check

```
bool isPali(string &s, int start, int end) {
    // base case
    if (start >= end) {
        return true;
    }
    // 1 case
    if (s[start] == s[end]) {
        start++;
        end--;
    }
    else {
        return false;
    }
    // recursion
    return ispali(s, start+1, end-1);
}
```

(18) Remove all occurence of a substring

→ already done with best TC, For Practice → Using Recursion

↳ i/p → S = " daabcbaabcbc "
part = "abc"
o/p → "dab"
→ check vs (TC is too high)

(19) Print all subarrays using Recursion

→ i/p → abcde
o/p → a, b, c, d, e
ab, bc, cd, de
abc, bcd, cde
abcd, bcde
abcde

```
void PrintSubarray_util(vector<int> $ nums, int start, int end) {
// base
  if(end == nums.size()) {
    return;
  }
// 1 case
  for(int i = start; i <= end; i++) {
    cout << nums[i] << " ";
  }
// RE
  PrintSubarray_util(nums, start, end+1);
```

→ (ha thik h, but exclusion/inclusion se karle bhai)

```
void PrintSubarray_util(vector<int> $ nums) {
    for(int start = 0; start < nums.size(); start++) {
        int end = start;
        PrintSubarray_util(nums, start, end);
    }
}
```

(20) Buy and Sell Stock

```
void findMaxProfit(vector<int> $ prices, int i, int $ minprice, int $ maxprofit) {
// base case
  if(i == prices.size()) {
    return;
  }
// update minprice
  if(prices[i] < minprices) {
    minprice = price[i];
  }
// calculate today's profit
  int todaysprofit = prices[i] - minprice;
// updating maxprofit if today's profit is greater
  if(todaysprofit > maxprofit) {
    maxprofit = todaysprofit;
  }
// Recursion call
  findMaxProfit(prices, i+1, minprice, maxprofit);
}

int main() {
  vector<int> prices = {7,1,5,3,6,4};
  int minprice = INT_MAX;
  int maxprofit = INT_MIN;
  findMaxProfit(prices, 0, minprice, maxprofit);
  cout << maxprofit;
```

→ 1 case

only one time in entire life you can buy and sell stock

price = [7,1,5,3,6,4]

o/p → 5

(6-1)

★1) **House Robber** → i/p → 1,2,3,1 (adjacent - robbing) o/p → 4

→ ek aur [2, 1, 1, 2] → 2+1=3, 1+2=3 → 3

Isse 2 entries to last le

(:) logic h or me → 4 ✓

int rob(vector<int>& nums, int i) {

if (i >= nums.size()) {

return 0;

// Two case: Rob the current house or skip it

int robCurrentHouse = nums[i] + rob(nums, i+2);

int skipCurrentHouse = rob(nums, i+1);

return max(robCurrentHouse, skipCurrentHouse);

}

→ 6 ke aur max adjacent sum me enter h

isme chg-chg sich kaise kr sakte

→ But isme koi v lo aur koi chodo (has estimation nhi hone chahiye)

House Robber - II

→ Isme has house circular h, to 1st aur aakhri house ...

↓

add one condition

22) **Wildcard Matching** → (Do by Recursion Now)

? → single character

★ → 0 ★ ∞

↓

se tu handle, 1, 2, ... Match karke dekh lage to agra

i/p → S = "aa", P = "a" o/p → False

i/p → S = "aa", P = "*" o/p → True

i/p → S = "aa", P = "?a" o/p → True

i/p → S = "aa", P = "a*" o/p → True

i/p → S = abcdefg, P = ab*fg o/p → True

(* → 0 to ∞ tk chla dega) → Koi v ek case me true aaya to answer True hojayega

```
bool isMatchHelper(string& s, int si, string& p, int pi) {
    // base
    if(si == s.size() && pi == p.size()) {
        return true;
    }
    if(si == s.size() && pi < p.size()) {
        while (pi < p.size()) {
            if (p[pi] != '*')
                return false;
        }
        return true;
    }
    // Single char matching
    if (s[si] == p[pi] || '?' == p[pi]) {
        return isMatchHelper(s, si+1, p, pi+1);
    }

    if(p[pi] == '*') {
        // treat '*' as empty or null
        bool caseA = isMatchHelper(s, si, p, pi+1);

        // let '*' consume one char
        bool caseB = isMatchHelper(s, si+1, p, pi);
        return caseA || caseB;
    }
```

Base Case

1 Case

2 Case

```
bool isMatch(string s, string p) {
    int si = 0;
    int pi = 0;

    return isMatchHelper(s, si, p, pi);
}
};
```

Integer to English Words

i/p → nums = 123
o/p → "One Hundred Twenty Three"

i/p → nums = 12345
o/p → "Twelve thousand Three Hundred Forty Five"

→ 100 Thousand (10^5)
→ Million (10^6)
→ Billion (10^9)
} → Base words

(In approach Number to Integer Using the Weight in Million Billion)

Base words → 1 to 20, 30, 40, 50, 60, 70, 80, 90 | 100 — hundred, 1000 — thousand, 10^6 — million, 10^9 — billion

INTERVIEW

• Iterate Karenge kude se chota (in-For Loop)

Solⁿ at → 35:15

• 123 > 100
one hundred → $\frac{123}{100} = 1$
$123/100 \rightarrow 23$

• 23 > 20
↳ twenty
$23\%20 \rightarrow 3$

• 3 = 3 → three
↓
one hundred twenty three

▶ • Find N Just >= in the Map
• if (N ≥ 100)
↳ How many map need?

▶ 12 345
Ten, Thousand, 345, 12

The only logic → Jab V6 code, V.V. easy ho jaye

```
int Solve (vector <int> & nums), int i, int n) {
    if (i>n) {
        return 0;
    }
    // include
    int include = nums[i] + solve(nums, i+2, n);
    // exclude
    int exclude = Solve(nums, i+1, n);
    return max(include, exclude);
}
```

vector < int > dp

if (dp[i] != -1) {
return dp[i];
}

dp

dp

```
int rob( vector < int > & nums) {
    int n = nums.size();
    if (n==1) return nums[0];
    if (n==2) return max(nums[0], nums[1]);
    int ans1 = Solve (nums, 0, n-2);
    int ans2 = Solve (nums, 1, n-1);
    return max(ans1, ans2);
}
};
```

dp1

dp2

dp[i] = max(include, exclude)
return dp[i]

vector < int > dp1(n+1, -1);
vector < int > dp2(n+1, -1);

```
for(auto i : map) {
    if(num >= i.first) {
        if (num >= 100)
            a = Solve (num/i.first)
        b = i.second;
        if (num % i.first != 0)
            c = Solve(num % i.first);
```

▶ vector < Pair < int, string > > map

• Ye 39 → DP se hoga, Recursion Me TLE aayega ~ for Brute force

(24) Perfect Square

i/p → 12
o/p → 3
(12 = 4+4+4)

i/p → 13
o/p → 2 (13 = 4+9)

12 = 1+11 → 1+10 → 1+9 → ...
for (1 to 3)
1+11 → 2+9 → 2+7 → ...
3+9 → 3+6 → ...
→ following this logic

```
int numSquare(int n) {
  // Base case
  if(n==0) {
    return 0;
  }

  if(n<0) {
    return INT_MAX; // This ensures that this choice is not consider
  }

  int ans = INT_MAX;
  int end = sqrt(n);
  for(int i=1; i<=end; ++i) {
    int PerfectSquare = i*i;
    int numberOfPerfectSquare = 1+ numSquare(n- PerfectSquare);
    ans = min(ans, numberOfPerfectSquares);
  }
  return ans;
```

→ 1 case

(25) Minimum Cost for Tickets

days → [1, 4, 6, 7, 12, 20] ; cost = [2, 7, 15]
o/p → 11

Pass			
day →	1	7	30

```
int minCostTickets_helper(vector<int>& days, vector<int>& costs,
                          int i) {
  // base case
  if(i>=days.size()) {
    return 0;
  }
  // Solution for 1 case
  // 1 DAY Pass
  int cost1 = costs[0] + minCostTickets_helper(days, costs, i+1);
  // 7 DAY Ticket
  int PassEndDay = days[i]+6;
  int j = i;
  while(j< days.size() && days[j]<= PassEndDay) {
    j++;
  }
  int cost7 = costs[1] + minCostTickets_helper(days, costs, j);
  // 30 DAY Pass
  PassEndDay = days[i] + 29;
  j = i;
  while(j< days.size() && days[j]<= PassEndDay) {
    j++;
  }
  int cost30 = costs[2] + minCostTickets_helper(days, costs, j);
  return min(cost1, min(cost7, cost30));
}

int minCostTickets(vector<int>& days, vector<int>& costs) {
  return minCostTickets_helper(days, costs, 0);
}
```

Solve
Ek case main solve kiyenge, baki Recursion sambhal lega

(Ye Pehle likha, yha galti hoegi)

(26) Number of Dice roll with target Sum

i/p → n = 1, k = 6, target
o/p → 1

no of dice

(1,2,3,4,5,6) face

```
int numRollsToTarget(int n, int K, int target) {
  // base cases
  if (target <0) {
    return 0;
  }
  if (n==0 && target ==0) {
    return 1;
  }
  if (n==0 || target ==0) {
    return 0;
  }

  int ways = 0;
  for (int i=1; i<=K; i++) {
    ways += numRollsToTarget(n-1, K, target-i);
  }
  return ways;
```

BACKTRACKING [Extended Recursion]

5 — Rat in a Maze

★ Questions

① Permutation of a String

```
void PrintPermutation (string $ str, int i) {
    // base case
    if (i >= str.length()) {
        cout << str;
        return;
    }
    // swapping
    for (int j = i; j < str.length(); j++) {
        // swap
        swap(str[i], str[j]);
        // rec. call
        PrintPermutation(str, i+1);
        // backtracking - to recreate the original input string
        swap(str[i], str[j]);
    }
}
```

② Rat in a Maze

```
bool isSafe (int i, int j, int row, int col,
             int arr[][4], vector<vector<bool>> $ visited) {
    if (((i>=0 && i<row) && (j>=0 && j<col)) &&
        (arr[i][j] == 1) && (visited[i][j] == false)) {
        return true;
    }
    else {
        return false;
    }
}

void SolveMaze (int arr[][4], int row, int col, int i,
                int j, vector<vector<bool>> $ visited,
                vector<string> $ Path, string output) {
    // base case
    if (i == row-1 && j == col-1) {
        // answer found
        Path.push_back(output);
        return;
    }
    // Down -> i+1, j
    if (isSafe(i+1, j, row, col, arr, visited)) {
        visited[i+1][j] = true;
        SolveMaze(arr, row, col, i+1, j, visited,
                  Path, output + 'D');
        // backtrack
        visited[i+1][j] = false;
    }
    // Left -> i, j-1
    if (isSafe(i, j-1, row, col, arr, visited)) {
        visited[i][j-1] = true;
        SolveMaze(arr, row, col, i, j-1,
                  visited, Path, output + 'L');
        // backtrack
        visited[i][j-1] = false;
    }
    // Right -> i, j+1
    if (isSafe(i, j+1, row, col, arr, visited)) {
        visited[i][j+1] = true;
        SolveMaze(arr, row, col, i, j+1,
                  visited, Path, output + 'R');
        // backtrack
        visited[i][j+1] = false;
    }
    // up -> i-1, j
    if (isSafe(i-1, j, row, col, arr,
               visited)) {
        visited[i-1][j] = true;
        SolveMaze(arr, row, col, i-1, j,
                  visited, Path, output + 'U');
        // backtrack
        visited[i-1][j] = false;
    }
}
```

(Interview)

③ N-Queen

→ isSafe ke TC ko O(n) se O(1) kar sakte → Some logic bar. (Q1 Most Place Karunga but solution saddal hai)

optimal → value

↳ Hashing ? but

```
Unordered_map<int, bool> rowCheck;
Unordered_map<int, bool> upperLeftDiagonalCheck;
Unordered_map<int, bool> bottomLeftDiagonalCheck;

void storeSolution(vector<vector<char>>$board, int n, vector<vector<string>>$ans) {
    vector<string> temp;
    for(int i=0; i<n; i++) {
        string output = "";
        for(int j=0; j<n; j++) {
            output.push_back(board[i][j]);
        }
        temp.push_back(output);
    }
    ans.push_back(temp);
}

bool isSafe(int row, int col, vector<vector<char>>$board, int n) {
    if (rowCheck[row] || upperLeftDiagonalCheck[n-1+col-row] || bottomLeftDiagonalCheck[row+col],
        return false;
    return true;
}

void solve(vector<vector<char>>$board, int col, int n, vector<vector<string>>$ans) {
    // base
    if(col >= n)
        storeSolution(board, n, ans);
        return;
}
// Try placing the queen in each row of the current column
    for(int row = 0; row < n; row++) {
        if (isSafe(row, col, board, n)) {
        // Place the Queen
        board[row][col] = 'Q';
        rowCheck[row] = true;
        upperLeftDiagonalCheck[n-1+col-row] = true;
        bottomLeftDiagonalCheck[row+col] = true;
        // Recursively solve for the next column
        Solve(board, col+1, n, ans);
```

```
// backtracking
    board[row][col] = '.';
    rowcheck[row] = false;
    upperleftDiagonalCheck[n-1+col-row] = false;
    bottomleftDiagonalCheck[row+col] = false;
  }
}
}

vector<vector<string>> SolveNQueens (int n) {
    vector<vector<char>> board(n, vector<char>(n,'.'));
    vector<vector<string>> ans;
    int col = 0;
// 0 → empty cell
// 1 → Queen at the cell
    Solve(board, col, n, ans);
    return ans;
```

Do this in interview

```
#include <iostream>
#include <vector>
using namespace std;
void printSolution(vector<vector<char>>& board, int n) {
    for(int i=0; i<n; i++) {
        for(int j=0; j<n; j++) {
            cout << board[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;
}
bool isSafe (int row, int col, vector<vector<char>>& board, int n) {
    int i = row;
    int j = col;
// check row
    while (j>=0) {
        if (board[i][j] == 'Q') {
            return false;
        }
        j--;
    }
// check upperleft diagonal
    i = row;
    j = col;
    while (i>=0 && j>=0) {
        if (board[i][j] == 'Q') {
            return false;
        }
        i--;
        j--;
    }
// check bottom left diagonal
    i = row;
    j = col;
    while (i<n && j>=0) {
        if (board[i][j] == 'Q') {
            return false;
        }
        i++;
        j--;
    }
return true;
void Solve (vector<vector<char>>& board, int col, int n) {
// base case
    if (col >= n) {
        printSolution (board, n);
        return;
    }
for (int row=0; row<n; row++) {
    if (isSafe(row, col, board, n)) {
        // 1 case
        board[row][col] = 'Q';
        // recursion
        solve(board, col+1, n);
        // backtracking
        board[row][col] = '.';
    }
}
}

int main() {
    int n = 4;
    vector<vector<char>> board (n, vector<char>(n,'.'));
    int col = 0;
    Solve(board, col, n);
    return 0;
}
```

Not a box

Not	0	0	0
1	1	0	0
1	1	0	0
0	1	1	Not

2 Left Set → DDRRR DRDDRR

Quick keep me for Image (to Image whole visited = Dual Kaise)

visited → 1

Not visited → 0

Do carry bro dis ?

visited

D = (i+1, j)

L = (i, j-1)

R = (i, j+1)

U = (i-1, j)

Yahan aate Hoyt hum false mark karenge

④ **Generate Parenthesis**

i/p → n = 3

o/p → ["((()))", "(()())", "(())()", "()(())", "()()()"]

(balanced) → ()

↳ generate all combination

* (use of backtracking)

```
Void Solve (vector< string> $ans, int n, int used_open, int used_close, int rem_o
            int rem_close, string output) {
  //base case
  if (rem_open == 0 && rem_close == 0) {
      ans.push_back(output);
      return;
  }

  // include an open bracket
  if (rem_open > 0) {
      solve(ans, n, used_open + 1, used_close, rem_open - 1, rem_close, output + '(');
  }
  // include a closed bracket
  if (used_open > used_close) {
      Solve(ans, n, used_open, used_close + 1, rem_open, rem_close - 1, output + ')');
  }
}

Vector < string > generateParenthesis(int n) {
    Vector < string > ans;
    int used_open = 0;
    int used_close = 0;
    int rem_open = n;
    int rem_close = n;
    string output = "";
    solve(ans, n, used_open, used_close, rem_open, rem_close, output);
    return ans;
}
```

incl. exclude

i / ~~incl~~ ← (condition) open > close

use this in your interview

```
void Solve(vector< string> $ans, int n, int open, int close,
           string output) {
  // base case
  if (open == 0 && close == 0) {
      ans.push_back(output);
      return;
  }
  // include open bracket
  if (open > 0) {
      output.push_back('(');
      Solve(ans, n, open-1, close, output);
      // backtrack
      output.pop_back();
  }
  // include close bracket
  if (close > open) {
      output.push_back(')');
      Solve(ans, n, open, close-1, output);
      // backtrack
      output.pop_back();
  }
}

vector < string > generateParenthesis(int n) {
    vector < string > ans;
    int open = n;
    int close = n;
    string output = "";
    Solve(ans, n, open, close, output);
    return ans;
}
```

if n = 3
→ open = 3
close = 3

⑤ Letter combinations of a Phone Number

(Backtrack)

i/p → digits = "23"
o/p → ["ad", "ae", "af", "bd", "be", "bf", "cd", "ce", "cf"]

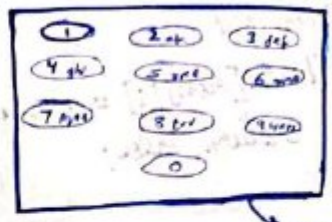
```
void solve(vector<string>$ ans, int index, string output, string digits, ~~vector<string~~
           vector<string>$ mapping) {
    //base case
    if(index >= digits.length()) {
        ans.push_back(output);
        return;
    }
    // 1 case
    char digitCharacter = digits[index];
    int digitInteger = digitCharacter - '0';
    string value = mapping[digitInteger];

    for(int i=0; i<value.length(); i++) {
        // include  char ch = value[i];
        // recursion  output.push_back(ch);
                   solve(ans, index+1, output+ch, digits, mapping);
        // backtrack  output.pop_back();
    }
}
```

index → 0 → 2
index → 1 → 3

using this, we can change character to integer ("23") convert to (23)

value nikala

string to integer



1 for Dashes

a = d, e, f
b = d, e, f
c = d, e, f

```
vector<string> letterCombinations(string digits) {
    vector<string> ans;
    // empty string ko diya
    if(digits.length()==0) {
        return ans;
    }

    int index = 0;
    string output = "";
    vector<string> mapping(10);
    mapping[2] = "abc";
    mapping[3] = "def";
    mapping[4] = "ghi";
    mapping[5] = "jkl";
    mapping[6] = "mno";
    mapping[7] = "pqrs";
    mapping[8] = "tuv";
    mapping[9] = "wxyz";
    solve(ans, index, output, digits, mapping);
    return ans;
```

loop

"23"

i = 2 → d, e, f
i = 3 → d, e, f

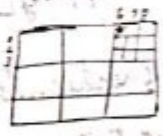
Bekuff ka logic, Iska map ka concept nhi lag rha h, Bas string aur Backtracking lag rha h basic ka

⑥ Sudoku Solver

```
→ bool isSafe(int row, int col, vector<vector<char>> &board, char value) {
    int n = board.size();
    for(int i=0; i<n; i++) {
      // Row check
      if(board[row][i] == value)
        return false;
      // column check
      if (board[i][col] == value)
        return false;
      // 3x3 box check
      if (board[3*(row/3)+(i/3)][3*(col/3)+(i%3)] == value)
        return false;
    }
    return true;
}
// this function return true or false, basis on we found a solution or not
→ bool solve(vector<vector<char>> &board) {
    int n = board.size();
    for(int i=0; i<n; i++) {
      for(int j=0; j<n; j++) {
        // check for empty cell
        if (board[i][j] == '.') {
        // Try to fill with values ranging from '1' to '9'.
        for(char val='1'; val<='9'; val++) {
          // check for safety
          if(isSafe(i, j, board, val)) {
          // insert
          board[i][j] = val;
          // Recursively solve the remaining board
          if (solve(board)) {
            return true;
          }
          else { // backtrack
            board[i][j] = '.';
          }
        }
      }
    }
    // if no valid value from '1' to '9' leads to a solution, backtrack
    return false;
}
// all cells filled
return true;
```

row = 0,1,2
col = 0,1,2

i = 0,1,2,3,...8



$i=0 \rightarrow \left(3\times\left(\frac{0}{3}\right)+\frac{0}{3}\right), \left(3\times\left(\frac{0}{3}\right)+0\%3\right) = (0,0)$

$i=1 \rightarrow 3\times\left(\frac{1}{3}\right)+\frac{1}{3} = 0 \;,\; 3\times\frac{0}{3}+1\%3 = (0,1)$

(0,0)
(1,0)
(1,1)

(agar success. board ko dil diya ab ye...)

→ ka value to se any sol' ki initial liya to
hum backtrack karte wo value to usaya se hata diye
(board value delete ki)

(Wo the se sol' nit match)
(pure tak sab nos ko kaise aur back de)

★ More Questions

(1) **Count Inversion** (N²) → based on merge sort

→ 4, 7 (two points)
$i < j$
$a[i] > a[j]$ ← form inversion

KNOW LOGIC
CODE Self

(M-1) → using 2 for loop.

```
for(i=0; i<N; i++){
    for(j=i+1; j<N; j++){
```

→ $O(n^2)$

Eg → i/p → {8, 4, 2, 1}
{1, 2, 4, 8}
Kitne swap operation karne ki ye sorted ho jaye

(8,4) (8,2) (8,1) (4,2) (4,1) (2,1) → Total 6 inversion h, inke swap karke to sorted hoyegya

o/p → 6

(M-2) → "Merge Sort ka thoda Modify karke karenge"
↓
$O(n \log n)$

8, 4, 2, 1

(8, 4, 2, 1)
(8,4) (2,1)
8 4 2 1

Left | Right

"4 se chota h to 8 se to chota hoga hi"

4, 8 | 1, 2

(4,1) (8,1) | (4,2) (8,2)
swap | swap
mid | end
mid+1

i = 0 1 2 3

4, 8 | 1, 2
i | mid | mid+1 | end

lcount = mid - i + 1
= 1 - 0 + 1
= 2

∴ 1, 2, 4, 8 (swap (1,4) (2,8))

if (a[i] > a[j]) {
left → right
}

1 2 4 8 ⇒ lcount = mid - i + 1

1 ka counting = 1 - 0 + 1
= 2 (ka (4,1)/(8,1))

2 ka count = 1 - 0 + 1
= 2

i | mid | mid+1 | end
8 | 4
inversion count

= mid - i + 1
= 0 - 0 + 1
= 1

```
★ int countInversion(vector<int> arr) {
    int count = 0;
    vector<int> temp(arr.size(), 0);
c = mergeSort(arr, temp, 0, arr.size() - 1);
// for(int i=0; i< arr.size(); i++){
        cout << arr[i];
    }
return c
```

```
int void mergeSort(vector<int>& arr, vector<int>& temp,
                   int start, int end) {
    if (start >= end)
        return c;
    int mid = start + (end-start)/2;
    int c = 0;
    c += mergeSort(arr, temp, start, mid);
    c += mergeSort(arr, temp, mid+1, end);
    c += merge(arr, temp, start, mid, end);
}
```

```
int void merge(vector<int>& arr, vector<int>& temp, int start, int mid, int end){
    int c=0;
    int i = start, j = mid+1, k = start;
    while (i<= mid && j<= end) {
        if(arr[i] <= arr[j])
            temp [k++] = arr[i++];
        else // arr[i] > arr[j]  i.e. inversion count case
            temp[k++]= arr[j++];
            c += mid - i + 1;
    }
    while(i<= mid)
        temp [k++] = arr[i++];
    while (j<= end)
        temp [k++] = arr[j++];
    while(start <= end){
        arr[start] = temp[start];
        ++start;
    }
    return count;
}
```

⑧ In-Place Merge Sort

→ we want $O(1)$ not using temp $(O(n))$

⑨ Maximum Subarray

College 9

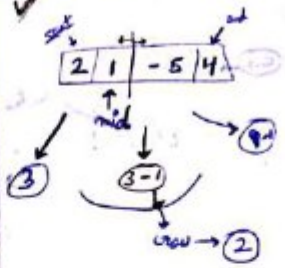
Divide & Conquer

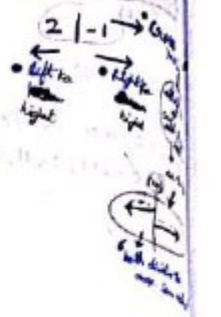
(self)

i/p → [-2, 1, 3, 4, -1, 2, 1, -5, 4]

o/p → 6

→ Subarray → [4, -1, 2, 1]






```
int maxSumArrayHelper(vector<int> &V, int start, int end)
  if (start == end)
    return v[start];

  int maxleftBorderSum = INT_MIN;
  int maxRightBorderSum = INT_MIN;
  int mid = start + (end-start)/2;

  int maxleftSum = maxSumArrayHelper(V, start, mid);
  int maxRightSum = maxSumArrayHelper(V, mid+1, end);

  // max cross Border Sum
  int leftBorderSum = 0 ; rightBorderSum = 0;
  for (int i = mid ; i >= start ; i--) {
      leftBorderSum += v[i];
    if (leftBorderSum > maxleftBorderSum)
      maxleftBorderSum = leftBorderSum;
  }

  for (int i = mid+1; i <= end; i++) {
      rightBorderSum += v[i];
    if (rightBorderSum > maxRightBorderSum)
      maxRightBorderSum = rightBorderSum;
  }

  int crossBorderSum = maxleftBorderSum + maxRightBorderSum
  return max(maxleftSum, max(maxRightSum, crossBorderSum))
```

→ $O(n \log n)$

10 Quick Sort

(Self)

(Pe mein)

→ (take Pivot as end)

11 Combination Sum

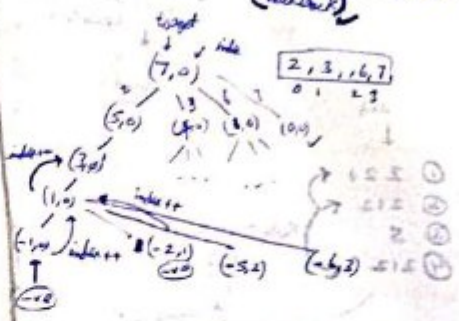
i/p → 2 | 3 | 6 | 7 , target = 7

o/p → (2, 2, 3)
(7)

→ 2 possible combinations

(Yeh dikh rha Sample Test Problem Set hoga)

• "Pick/drop, aur recursion sambhal lega" (backtrack) → (inclusion/exclusion)



o/p → [[2,2,3],[7]]

```
void CombinationSum_helper(vector<int>& candidates, int target,
    vector<int>& v, vector<vector<int>>& ans, int index) {
// base case
    if (target == 0)
        ans.push_back(v);
        return;

    if (target < 0)
        return;

    for(int i = index; i < candidates.size(); i++) {
        v.push_back(candidates[i]);
// recursion
        CombinationSum_helper(candidates, target - candidates[i],
                                v, ans, i);
// backtrack
        v.pop_back();
    }
}
```

▸ Subsets Sum (L-977) is ||

↓

Maximum Subsets
⊕
Combination Sum

(12) Combination Sum - 2

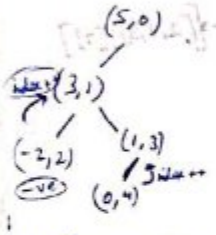
→ No duplicate combination
→ Each candidate only be used once

i/p → [2, 5, 2, 1, 2] , target = 5

o/p → [[5], [2, 2, 1]]

(2, 2, 1) different h, but duplicate h → Not allowed

2 | 5 | 2 | 1 | 2
0 1 2 3 4



ans
① 2 2 1
② 2 1 2
③ 5
④ 2 1 2

Repeat as she (:) (duplicate)

insert to set → using set

```
void CombinationSum_helper(vector<int>$ candidates, int target,
                           vector<int>$ v, vector<vector<int>>
                           $ ans, int index) {
    // base
    if( target == 0)
        ans.push_back(v);
        return;
    if( target < 0)
        return;
    for (int i = index; i < candidates.size(); i++) {
        v.push_back(candidates[i]);
        // recurse
        CombinationSum_helper(candidates, target - candidates[i], v, ans, i+1);
        // backtrack
        v.pop_back();
```

→ Vector<vector<int>> CombinationSum2(vector<int>$ candidates, int target) {

```
    sort(candidates.begin(), candidates.end());
    vector<vector<int>> ans;
    vector<int> v;
    CombinationSum_helper(candidates, target, v, ans, 0);
    set<vector<int>> st;
    for (auto e : ans) {
        st.insert(e);
    }
    ans.clear();
    for (auto e : st) {
        ans.push_back(e); // only unique vectors
    }
    return ans;
}
```

(13) Permutation - 2

Given a collection of numbers, nums, that might contain duplicates, return all possible unique permutations in any order.

i/p → nums = [1, 1, 2]

o/p → [[1, 1, 2], [1, 2, 1], [2, 1, 1]]

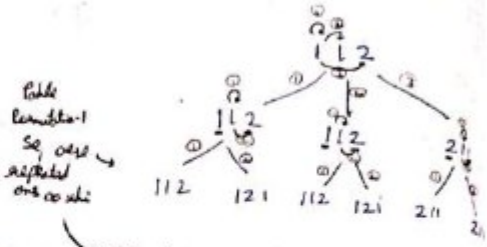


Table Permutation-1 Se aaya, repeated ans aa rahi

→ To hum set me dal to sirf bata rahi

```
void permuteUniqueHelper(vector<int>$ nums,
                         vector<vector<int>>$ ans, int start) {
    // base
    if(start == nums.size()) {
        ans.push_back(nums);
        return;
    }
    for(int i = start; i < nums.size(); i++) {
        swap(nums[i], nums[start]);
        PermuteUniqueHelper(nums, ans, start+1);
        swap(nums[i], nums[start]);
    }
}
```

```
vector<vector<int>> PermuteUnique(vector<int>$ nums) {
    vector<vector<int>> ans;
    PermuteUniqueHelper(nums, ans, 0);
    // Push ans into set
    set<vector<int>> st;
    for (auto e : ans) {
        st.insert(e);
    }
    ans.clear();
    for(auto e : st) {
        ans.push_back(e);
    }
    return ans;
}
```

(P2) → video → 2 hrs

(14) Beautiful Arrangement

Suppose you have n integers labeled 1 through n. A permutation of these n integers perm (1-indexed) is considered a beautiful arrangement if for every i (1<=i<=n), either of the following is true:

- perm[i] is divisible by i
- i is divisible by perm[i]

perm[i] → | 1 | 2 |

idx → 1 2

perm[i] % i = 0
OR
i % perm[i] = 0 (for all distinct in full array)

i/p → 2
o/p → 2

1 2
2 1

1 2 3

[1 3 2] → 1%1 ✓, 3%2 ✓, 2%3 ✗

[2 1 3]

[3 2 1]

2%1=0 ✓
1%2=0 ✗ 2%1=0 ✗ → 1%2 ✗
2%3=0 ✗
3%2=0 ✗

6 permutation me se, 3 is beautiful arrangement o/p → 3

(M1) → Brute force
- find all permutation
- extract out beautiful ones

(M2) → _ _ _ (1 2 3)

(Backtrack se at-at Karke dekh lenge)

i/p → N = 3

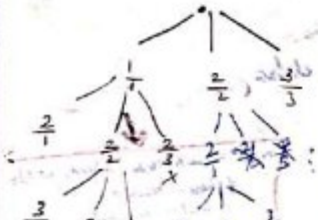
(pehle table, table banaiye ho)

1 ko 1st index par rakh ke dekh ab, Phir 2nd index par rakh ke dekh ab, Phir 3rd index ...

2 ko 1st

i ko 2nd index → 1 ko 3rd index ✓
2 ko 3rd index → 1 ko 2nd index ✗

(ulta order)



void countArrangementHelper(vector<int>&v, int&n, int&ans, int currNum) {

```
//base
if(currNum == n+1) {
    ++ans;
    return;
}

for (int i = 1; i<=n; i++) {
    if (v[i] == 0 && (currNum % i == 0 || i % currNum == 0)) {
        v[i] = currNum;
        countArrangementHelper(v, n, ans, currNum+1);
        v[i] = 0;   // backtracking
    }
}
}

int countArrangement(int n) {
    vector<int> v(n+1);
    int ans = 0;
    countArrangementHelper(v, n, ans, 1);
    return ans;
}
};
```

(Ye Bataiye Khud h, aur Main wha rakh ab hu)

Last ek chij → vha index 1 se start hoega aur kaha tak

(15) Distribute Repeating Integers

Sorted Product Sorted Product

1 | 1 | 2 | 2 → nums (n)

[2, 2] → quantity (m)

0th index 1st index

2 quantity chahiye aur dono same quantity

} → return true or false

1 | 1 | 2 | 2 | 2 | 2 | 3 | 3 → n

[2, 4, 2]

1st → [2 | 2 | 3]

0th → 1, 1 ✓
1st → 2, 2 ✓
3 → Nhi mil pa rahe (brute force se)

use Recursion + Backtracking

▶ Yha answer to mil jaa rha lekin TLE aa rha, Jaise hi test N TLE aa rha hoga "FALSE" ke case N

2 | 2 | 1
2 | 2 | 3 (2-times backtrack)

better quantity ko decreasing order me sort karke lu

```
bool solve(vector<int>& count, vector<int>& quantity, int i) {
    // base case
    if (i == quantity.size()) {
        return true;
    }
    for (int i = 0; i < count.size(); i++) {
        if (count[i] >= quantity[i]) {
            count[i] = count[i] - quantity[i];
            if (Solve(count, quantity, i+1)) {
                return true;
            }
            // backtrack
            count[i] = count[i] + quantity[i];
        }
    }
    return false;
}

bool canDistribute(vector<int>& nums, vector<int>& quantity) {
    unordered_map<int, int> hash;
    for (auto i : nums) {
        hash[i]++;
    }
    vector<int> count;
    for (auto i : hash) {
        count.push_back(i.second);
    }
    sort(quantity.rbegin(), quantity.rend());
    return solve(count, quantity, 0);
}
};
```

6 – Linked List

{ Book Kaboni h see chapter } → Hindi h book xd...

(Non-contiguous Memory)

Basic:-

- LL create & print

```
Node* p = new Node(data);
```

- Node* temp = head

- Node* next

① Insert at left of head:-
- create a node
- NewNode → next = head
- head = newNode

② Insert at right of tail → U know how

③ Insert at specific location

④ Delete 1st Node

⑤ Delete LastNode

⑥ Delete at Middle

⑦ Do all same with doubly linked list

⑧ Reverse the LL → (what Input OR Interview se ask h)

```
Node* reverse(Node* head) {
    Node* prev = NULL;
    Node* curr = head;
    while(curr != NULL) {
        Node* temp = curr -> next;
        curr -> next = prev;
        prev = curr;
        curr = temp;
    }
    return prev;
}
```

- Circular doubly LL

- Get Middle of LL → (using slow-fast approach)

```
Node* getMiddle(Node* & head) {
    if(head == NULL) {
        cout << "LL is empty"
        return head;
    }
    if(head -> next == NULL) {
        return head;   // only 1 LL
    }
    Node* slow = head;
    Node* fast = head;
    while(slow != NULL && fast != NULL) {
        fast = fast -> next;
        if(fast != NULL) {
            fast = fast -> next;
            slow = slow -> next;
        }
    }
    return slow;
}
```

⑨ K groups → reverse LL

- reverse first K nodes of LL
- Recursive call
- Return head of modified LL

```
Node* reverseK(Node* & head, int k) {
    if (head == NULL) {
        cout << "LL is empty"
        return NULL;
    }
    int kn = getLength(head);
    if (k > kn) {
        return head;
    }
    Node* prev = NULL;
    Node* curr = head;
    Node* forward = curr -> next;
    int count = 0;
    while (count < k) {
        forward = curr -> next;
        curr -> next = prev;
        prev = curr;
        curr = forward;
        count++;
    }
    if(forward != NULL) {
        Node* recursionKn = reverseK(forward, k);
        head -> next = recursionKn;
    }
    return prev;
}
```

- isCircular → True/False

⑩ Detect and Delete Loop (same Visitor)

- check loop is present in LL or not
- Starting point of loop
- Remove Loop

- Slow/fast ko next karwao
- Slow ko dubara starting se chalao (Part two)

- Check Palindrome of LL

- Remove duplicates from sorted LL

⑪ Sort 0,1,2 in LL → find count of zeros, ones and twos; fill 0, 1 and 2s in the original LL

- Add 2 no. by LL → Reverse both LL; Add both; again Reverse one Add LL

```
if(slow == fast)
    ∴ Loop Found
```

```
if(head == NULL) {
    head = new Node(data);
    return;
}
if(head -> next == NULL) {
    Node* temp = head;
    head = head -> next;
    temp -> next = NULL;
    delete temp;
    return;
}
```

⑫ Class Node {

```
Public:
    int data;
    Node* next;

    Node(int data) {
        this -> data = data;
        this -> next = NULL;
    }
}
```

→ "If u do this, u do all!"

```
int main() {
    Node* head = NULL;
    Node* tail = NULL;
    insertAtHead(head, tail, 20);
    print(head)
}
```

```
void insertAtHead(Node* & head, Node* & tail, int data) {
    // check for empty LL
    if(head == NULL) {
        Node* newNode = new Node(data);
        head = newNode;
        tail = newNode;
    }
    else {
        // Step 1 → create a node
        Node* newNode = new Node(data);
        // Step 2 → newNode → next = head
        newNode -> next = head;
        head = newNode;
    }
}
```

```
void Print(Node* head) {
    Node* temp = head;
    while (temp != NULL) {
        cout << temp -> data;
        temp = temp -> next;
    }
}
```

Next Imp. link

Questions

① K^{th} node from the end

3 → 2 → 1 → Null

Index → 2 1 0

i/p → 2 (Index)
o/p → 3

(M-1) → Recursion ✓ → (chal na aaye kaunsaa h)

'easy' (M-2) → Length − K (Easy) ✓ → iska upar se code karna

(M-3) → Recursion ✗ (this question)

② Intersection of 2 LL

i/p → head A, head B, intersection

1 → 2 → 3 → 4 → 5 → x

1 → 2 → 3

o/p → 3

approach → koi ek null to dono ke diff nikalo = 1, ab b ka EK aage se hi shuru karo, aur jo milegi wo Ans

→ My Method → dono ke total kiye, a → 5 (null tk gaya) usse ke b → 4 tk gaya. → ab a ko dekhna shuru se start kiye ● [redacted]

aur
Iske diff h, i.e. 1
usna sirf usse extra chal liya starting me.
Now, jo ek next hoga wo intersection.)

Self

③ Merge two Sorted Lists

i/p → ① → ② → ⑤
① → ③ → ④

o/p → ① → ① → ② → ③ → ④ → ⑤

Easy approach

For inserting new node in LL
Node* one = new node(-1);

④ Sort List Using Merge Sort

① Null
② Using Null check array
③ RE check
④ Merge

sort( ) {
(Find/Stop se mid kaun)
left = head, right = mid → next & mid → next = 0
left = sort(left)
right = (mid/end)
recursion call
merge(left, right)
}

⑤ Ebitten Linked List (has head & tail better LL ka)

i/p → head, tail

5 → 10 → 19 → 28 → x

7 → 8 → 30 → x
20 → x
22 → 50 → x
35 → 40 → 45 → x

sort

Main LL
Sub LL

o/p → Sorted LL → sort order me Print karna

⑥ Clone LL with Random Pointer

i/p → 7 → 13 → 11 → 10 → 1 → x

Next Pointer
Random Pointer

o/p → Same chiz, Jhalinge

∴ to, 7 → 13 → 11 → 10 → 1 se banaenge

Phir, dekhLL me jisse delete 13 kiyaa tha
usse sab h, wait chart 10 wali
me Point karenge
Head ki help se
head = head → next
(copy) wapis ka
[redacted]
chart
original LL ka
aur same replicate
karenge clone LL me

Brute Method
Map me store karenge,
(dekh ke use Naaste)

Time of Map

⑦ Rotate List

i/p → ① → ② → ③ → ④ → ⑤
k = 2

o/p → ① → ④ → ⑤ → ① → ②

approach

while (k--)
head = head → next
ab us Node ko
Tail (2) ka
Next NULL bnaaio
aur jo starting ka
Tail ka pointer next
(head ki jgh)
ko Equal karo

⑧ Delete N nodes after M nodes

i/p → 9 → 1 → 3 → 5 → 9 → 7 → 6 → 8 → 1
M = 2
N = 1
o/p → 9 → 1 → 5 → 9 → 7 → 6 → 8 → 1

Soln,
Kahani Bolo
CODE KARO

PURE INTERVIEW Q.
leetcode - 2058

⑨ Find min/max no. b/w critical Points

Local Maxima
or
Local Minima

Edge case (-1, -1)

i/p → 5 3 1 2 5 1 2
o/p → [1, 3]

Min hot kaise
check karna hoga
(Brute force se)

① (6 − 5)

Max dist =
last cp − first cp
= 6 − 3
= ③

∴ o/p → (1, 3)

⑩ Merge Nodes in Between Zones

i/p → 0 → 3 → 1 → 0 → 4 → 5 → 2
o/p → (4, 11)

Fizzy

Edge case main solve karenge
Baki recursion dekhe kiya?

Extra → edge
se Extra

7 - STACK (LIFO)

• Push, Pop, top, size, empty

• 2 Stack in an array

top1 = -1, top2 = Size

• Reverse the String

• Find middle element in stack

• Insert at Bottom

• Reverse a stack

• Valid Parenthesis

• Sort a stack

• Remove Redundant Brackets

• Min Stack

• Next Smaller element

• Length of longest Valid Parenthesis

• Prev. Smaller element

• Largest Rectangle in Histogram

```
void insertAtBottom (stack<int> &s, int target){
    // base case
    if (s.empty()) {
        s.push(target);
        return;
    }
    int topElement = s.top();
    s.pop();
    // rec call
    insertAtBottom(s, target);
    // BT
    s.push(topElement);
}
```

```
void reverseStack(stack<int> &s){
    //base case
    if (s.empty()) {
        return;
    }
    int target = s.top();
    s.pop();
    // reverse stack
    reverseStack(s);
    // insert at bottom target ko
    insertAtBottom(s, target);
}
```

```
void Middle (stack<int> &s, int
    if (s.size() == 0){
        // no elements
        return;
    }
    //base case
    if(s.size() == totalSize/2 + 1){
        cout << "Middle element is"
        return;
    }
    int temp = s.top()
    s.pop();
    // recursive call
    Middle(s, Tsize);
    //backtrack
    s.push(temp)
```

```
isValid(string s){
    stack<char> st;
    for(int i=0; i<s.length(); i++){
        char ch = s[i];
        // opening bracket
        if( ch=='(' || ch=='{' || ch=='[' ){
            st.push(ch);
        }
        else {
            // closing bracket
            if(!st.empty()){
                char topch = st.top();
                if (ch==')' && topch=='(') {
                    // matching bracket
                    st.pop();
                }
                else if (ch=='}' && topch=='{') {
                    // matching bracket
                    st.pop();
                }
                else if (ch==']' && topch=='[') {
                    // matching bracket
                    st.pop();
                }
                else {
                    // bracket not matching
                    return false;
                }
            }
            else return false;
        }
    }
    if(st.empty())
        return true;
```

```
void insertSorted(stack<int> &s, int target){
    // base case
    if(s.empty()) {
        s.push(target);
        return;
    }
    if(s.top() >= target) {
        s.push(target);
        return;
    }
    int topElement = s.top();
    s.pop()
    insertSorted(s, target);
    // BT
    s.push(topElement);
}
void sortStack(stack<int> &s){
    // base case
    if(s.empty()){
        return;
    }
    int topElement = s.top();
    s.pop();
    sortStack(s);
    insertSorted(s, topElement);
```

```
vector<int> nextSmaller(vector<int>&
    Stack<int> s;
    s.push(-1);
    vector<int> ans(input.size());
    for(int i = input.size()-1; i>=0; i--){
        int curr = input[i];
        // apne answer stack me
        while(s.top() >= curr){
            s.pop();
        }
        // chota element mil chukha hai
        ans[i] = s.top();
        // push karo curr element ko
        s.push(curr);
    }
    return ans;
```

```
#include <iostream>
#include <stack>
using namespace std;

int main() {
    string str = "babbar";
    stack<char> s;
    for(int i=0; i<str.length(); i++)
        s.push(str[i]);
    while(!s.empty()) {
        cout << s.top();
        s.pop();
    }
```

★ **Questions**

① Remove all adjacent duplicates string

↳ i/p → abbaca
o/p → ca

② Minimum Bracket Reversal

↳ S = "}{{}}{{{"
o/p → 3

③ Next Greater Element in LL

(Self) LL + Stack

↳ i/p → head = [2,1,5]
o/p → [5,5,0]

④ The celebrity Problem

↳ N = 3 → (total people in party)
M = [[ ]]
o/p → 1

(M-1) (Brute force) → • celebrity → all 0's
celebrity → all 1's
→ row all have value

(M-2)

⑤ N stacks in an array

(hard)

```
Stack<char> St;
for(char ch : s){
    if(ch == '{'){
        St.push(ch);
    }else{
        if(!st.empty() && st.top() == '{'){
            St.pop();
        }else{
            St.push(ch);
        }
    }
}
// If stack is still not empty
while(!St.empty()){
    char a = St.top();
    St.pop();
    char b = St.top();
    St.pop();
    if(a == b){
        ans += 1;
    }else{
        ans += 2;
    }
}
```

⑥ Online Stack Span

```
Stack<pair<int, int>> st;
int next(int price){
    int span = 1;
    while(!st.empty() &&
        st.top().first <= price){
        span += st.top().second;
        st.pop();
    }
    st.push({price, span});
}
```

↳ i/p → 7 2 1 2 2 ← Today (5th day)
o/p → 4 (hah hah when hi price the 4 week kar the)

⑦ Simplify Path

↳ Bhai rahem kaa, "Ye 4 chards!!"

⑧ Check if word is valid after Substitution

(VS)

```
if(s[0] != 'a') return false
• b n to top 'a' kaa chahiye → push
  else → false
• c n to top 'b' kaa chahiye aur top 'b' then top 'a' kaa chahiye aur top 'a'
  else false
  else false
```

↳ i/p → S = "aabcbc"
o/p → true

⑨ Decode String → VS

(Self)

↳ • i/p → S = "3[a]2[bc]"
o/p → aaabcbc

⑩ Max Rectangle in Binary Matrix

(hard)

DP 4

i/p → [[1,0,1,0,0], [1,0,1,1,1], [1,1,1,1,1], [1,0,0,1,0]]
o/p → 6 (max area)

(VS)

Bhai understand in DP

1 0 1 0 0
2 0 2 1 1
3 1 3 2 2
4 0 0 0 0

(Bi-monotonic) → (all time cube)

```
int max = INT_MIN;
vector<int> histogram(matrix[0].size(), 0);
for(int i = 0; i < matrix.size(); i++){
    for(int j = 0; j < histogram.size(); j++){
        if(matrix[i][j] == '1'){
            histogram[j]++;
        }else{
            histogram[j] = 0;
        }
    }
    max = max(max, largestRectangle(histogram));
}
return max;
```

⑪ Car Fleet

(Easy)

i/p → target = 12,
position = [10,8,0,5,3]
speed = [2,4,1,1,3]
o/p → 3

⑫ Car Fleet - II

(hard)

i/p → cars = [[1,2],[2,1],[3,3],[3,2]]
o/p → [1.00000, -1.00000, 2.00000, -1.00000]

• Largest Rectangle in Histogram

i/p → h
o/p → 10

width = next - previous - 1 , area = width × height → max

• vector<int> prev = leftSmallerElement(height);
vector<int> next = nextSmallerElement(height);

```
int maxArea = INT_MIN;
int size = height.size();
for(int i = 0; i < height.size(); i++){
    int length = height[i];
    if(next[i] == -1){
        next[i] = size;
    }
    int width = next[i] - prev[i] - 1;
    int area = length * width;
    maxArea = max(maxArea, area);
}
return maxArea;
```

(VS)

8- Queue (FIFO)

include <queue>

queue < int > q;

- q.push(5);
- q.pop();
- q.front();
- q.size();

STL se hi karo

Class se kaise h to VS code ko to pta

Client se kaise ye use kare input kiya h?

front / back

- push, pop, front, empty, size

include <deque>

deque < int > q → double sided queue

Say as dq

Pick from front se bhi lo or back se bhi push kr sakte h

- push_front()
- push_back()
- pop_front()
- pop_back()

① • Reverse the queue

② • Reverse first K element of queue

③ • Interleave first and second half of queue (at most one extra space)

④ • Find first -ve integer in every window of size K

↳ i/p → 12 -1 -7 8 -15 30 16 28

o/p → {-1, -1, -7, -15, -15, 0}

Sort & offline

⑤ • Find first non-repeating character in a stream

⑥ • Gas Station → OR

↳ i/p → gas = [1, 2, 3, 4, 5], cost = [3, 4, 5, 1, 2]

o/p → 3

Kaise se shortest time me, kaise kaise circular hoke complete hoti hai (which station!)

Let '0' be h, balance = 1 (abhi ek petrol bacha hai agle kaal ke liye) dist = 3

Let '1' be h, balance = 2 (ab 2 ho gaya) dist = 4

Let '2' be h, balance = 3 (ab) dist = 5

Let '3' be h, balance = 4 (abhi bhi) dist = 1

find start?

★ Questions

① Queues using Stack

② Stack using Queue

③ K-Queue in an array

④ Sum of Min/Max elements of all Subarray of size k

⑦ • Sliding Window Maximum → VS → i/p → nums = [1, 3, -1, -3, 5, 3, 6, 7]

o/p → [3, 3, 5, 5, 6, 7]

```
int canCompleteCircuit(vector<int>& gas, vector<int>& cost){
    // Kitna Petrol Kam Padega
    int deficit = 0;
    // Kitna Petrol bacha hua h
    int balance = 0;
    // Circuit kaha se start karna ha
    int start = 0;

    for(int i = 0; i < gas.size(); i++){
        balance += gas[i] - cost[i];
        if (balance < 0){
            deficit += abs(balance);
            start = i+1;
            balance = 0;
        }
    }
    if (balance >= deficit){
        return start;
    }
    else {
        return -1;
    }
}
```

→ O(n²) → (M-1)

→ Matlab circle complete hogya

(M-3)

gas →	1	2	1	3	5
cost →	3	1	5	1	4

→ balance

deficit

```
void reverseQueue(queue<int> &q){
    Stack<int> s;
    while(!q.empty()){
        int element = q.front();
        q.pop();
        s.push(element);
    }
    while(!s.empty()){
        int element = s.top();
        s.pop();
        q.push(element);
    }
}
```

```
void interleaveQueue(queue<int> &q){
    int n = q.size();
    if (q.empty())
        return;
    int k = n/2;
    int count = 0;
    queue<int> q2;
    while (!q.empty()){
        int temp = q.front();
        q.pop();
        q2.push(temp);
        count++;
        if (count == k)
            break;
    }
    while (!q.empty() && !q2.empty()){
        int first = q2.front();
        q2.pop();
        q.push(first);
        int second = q.front();
        q.pop();
        q.push(second);
    }
    // odd wala case
    if (n%2){
        int element = q.front();
        q.pop();
        q.push(element);
    }
}
```

```
void reverseK(queue<int> &q, int k){
    Stack<int> s;
    int count = 0;
    int n = q.size();
    if (k <= 0 || k > n)
        return;
    while (!q.empty()){
        int temp = q.front();
        q.pop();
        s.push(temp);
        count++;
        if (count == k)
            break;
    }
    while (!s.empty()){
        int temp = s.top();
        s.pop();
        q.push(temp);
    }
    count = 0;
    while (!q.empty() && n-k){
        int temp = q.front();
        q.pop();
        q.push(temp);
        count++;
        if (count == n-k)
            break;
    }
}
```

```
string Solve(string str){
    int freq[26] = {0};
    queue<char> q;
    string ans = "";
    for (int i = 0; i < str.length(); i++){
        char ch = str[i];
        freq[ch - 'a']++;
        q.push(ch);
        while (!q.empty()){
            if (freq[q.front() - 'a'] > 1)
                q.pop();
            else {
                ans.push_back(q.front());
                break;
            }
        }
        if (q.empty())
            ans.push_back('#');
    }
    return ans;
}
```

```
void Solve(int arr[], int n, int k){
    deque<int> q;
    // Process first window of size k
    for (int i = 0; i < k; i++){
        if (arr[i] < 0)
            q.push_back(i);
    }
    // Remaining windows ko process kro
    for (int i = k; i < n; i++){
        // Ki current window me koi element bacha hai
        if (!q.empty())
            cout << arr[q.front()];
        else {
            cout << 0;
        }
        // Out of window element ko remove karo
        while ((!q.empty()) && (i - q.front() >= k))
            q.pop_front();
        // Check current element for insertion
        if (arr[i] < 0)
            q.push_back(i);
    }
    // Remaining last window for last element
    if (!q.empty())
        cout << arr[q.front()];
    else {
        cout << 0;
    }
}
```

Q-Trees

→ (Save Mid Pointer + Recursion Kaha)

• Basic-1

```
• class Node {
    public:
      int data;
      Node* left;
      Node* right;

    Node(int data) {
        this->data = data;
        left = NULL;
        right = NULL;
    }
};
```

ans → 10 60 30 80 20 50 70 40

↓ FIFO

↓
• Node
• curr Node (null)

⇓
(a) temp = q.front
(b) q.pop
(c) Print ans
(d) child insert

(4 step)

"q.push K" queue ka hua

```
queue <Node*> q;
q.push(root);
while (!q.empty()) {
    Node* temp = q.front();
    q.pop;
    cout << temp -> data;
    if (temp -> left) {
        q.push(temp -> left);
    }
    if (temp -> right) {
        q.push (temp -> right);
    }
}
```

```
→ Node* buildTree() {
    int data;
    cin >> data;
    if (data == -1) {
        return NULL;
    }
    // 3 step → Node, left, right
    Node* root = new Node(data);
    root -> left = buildTree();
    root -> right = buildTree();
    return root;
}
```

```
→ int height(Node* root) {
    if (root == NULL)
        return 0;
    int leftHeight = height(root -> left);
    int rightHeight = height (root -> right);
    int ans = max (leftHeight, rightHeight) + 1;
    return ans;
}
```

```
→ int main() {
    Node* root = NULL;
    root = buildTree;
    cout << height(root);
```

(V3) (using queue)

```
• void levelOrderTraversal (Node* root) {
    queue < Node* > q;
    q.push(root);
    q.push (NULL);
    while (!q.empty()) {
        //A
        Node* temp = q.front();
        //B
        q.pop()
        if (temp == NULL) {
            cout << endl;
            if (!q.empty()) {
                q.push (NULL);
            }
        }
        else {
            //C
            cout << temp -> data;
            //D
            if (temp -> left) {
                q.push (temp -> left);
            }
            if (temp -> right) {
                q.push (temp -> right);
            }
        }
    }
}
```

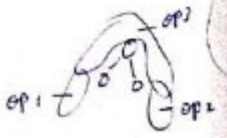
(Insert from end)

```
• void inorderTraversal (Node* root) {
    // Base case
    if (root == NULL)
        return;
    // LNR
    inorderTraversal (root -> left);
    cout << root -> data;
    inorderTraversal (root -> right);
}
```

```
• void preorderTraversal (Node* root) {
    if (root == NULL) {
        return;
    }
    // NLR
    cout << root -> data;
    preorderTraversal (root -> left);
    preorderTraversal (root -> right);
}
```

```
• void postorderTraversal (Node* root) {
    if (root == NULL) {
        return;
    }
    // LRN
    postorderTraversal (root -> left);
    postorderTraversal (root -> right);
    cout << root -> data;
```

```
* int diameter_of_B_T(Node* root) {
    if (root == NULL)
        return 0;
    int op1 = diameter_of_B_T (root -> left);
    int op2 = diameter_of_B_T (root -> right);
    int op3 = height (root -> left) + 1 + height (root -> right);
    int ans = max (op1, max (op2, op3));
    return ans;
```



• Balanced Binary Tree

```
class Solution {
    public:
        bool isBalanced(TreeNode* root) {
            // Base case
            if(root == NULL)
                return true;
            // 1 case
            int leftHeight = height(root -> left);
            int rightHeight = height(root -> right);
            int diff = abs(leftHeight - rightHeight);
            bool ans1 = (diff <= 1);
            // Recursion
            bool leftAns = isBalanced(root -> left);
            bool rightAns = isBalanced(root -> right);
            if(ans1 && leftAns && rightAns)
                return true;
            else
                return false;
```

• Lowest Common Ancestor of BT

```
class Solution {
    public:
        TreeNode* lowestCommonAncestor(TreeNode* root, TreeNode* p, TreeNode* q) {
            // base case
            if(root == NULL)
                return NULL;
            // check for p and q
            if(root -> data == p -> data)
                return p;
            if(root -> data == q -> data)
                return q;
            TreeNode* leftAns = lowestCommonAncestor(root -> left, p, q);
            TreeNode* rightAns = lowestCommonAncestor(root -> right, p, q);
            if(leftAns == NULL && rightAns == NULL)
                return NULL;
            else if(leftAns != NULL && rightAns == NULL)
                return leftAns;
            else if(leftAns == NULL && rightAns != NULL)
                return rightAns;
            else {
                return root;
            }
        }
```

• Check whether BT is sum tree or not (convert BT into sum tree)

```
ConvertIntoSumTree(root) {
    // B.C
    if(root == NULL)
        return 0;
    int leftAns = ConvertIntoSumTree(root -> left);
    int rightAns = ConvertIntoSumTree(root -> right);
    root -> data = leftAns + rightAns + root -> data;
    return root -> data;
}
```

• K^{th} Ancestor

① find P
② K step upper to
③ one subtree

```
bool Kth Ancestor(Node* root, int &K, int P) {
    // base case
    if (root == NULL)
        return false;
    if (root -> data == P) {
        return true;
    }
    bool leftAns = KthAncestor(root -> left, K, P);
    bool rightAns = KthAncestor(root -> right, K, P);

    if (leftAns || rightAns)
        K--;
    if (K == 0) {
        cout << root -> data;
        K = -1;
    }
    return leftAns || rightAns;
}
```

(parent node upper and upper side)
(left upper side left K to return parent)

if (leftAns == true || rightAns == true)

• Target Sum → Path Left

```
class Solution {
    Public:
        void solve(TreeNode* root, int targetSum,
            int &currSum, vector<int> &Path,
            vector<vector<int>> &ans) {
    // base case
    if (root == NULL) {
        return;
    }
    // leaf node
    if (root -> left == NULL && root -> right == NULL) {
        // include curr node
        Path.push_back(root -> val);
        currSum += root -> val;
        // check for target sum
        if (currSum == targetSum) {
            ans.Push_back(Path);
        }
        // Exclude
        Path.pop_back();
        currSum -= root -> val;
        return;
    }
    // include curr node
        Path.Push_back(root -> val);
        currSum += root -> val;
        Solve(root -> left, targetSum, currSum, Path, ans);
        Solve(root -> right, targetSum, currSum, Path, ans);
        // backtrack
        Path.pop_back();
        currSum -= root -> val;
    }

    vector<vector<int>> PathSum(TreeNode* root,
        int targetSum) {
        vector<vector<int>> ans;
        int sum = 0;
        vector<int> temp;
        Solve(root, targetSum, sum, temp, ans);
        return ans;
    }
```

leaf node ke liye

Baki nodes ke liye

Basic - 3

⊛ i/p → inorder (LNR)
↳ Preorder (NLR)

o/p → Postorder
or
buildTree

(2-step Process)

```
#include <iostream>
using namespace std;
```

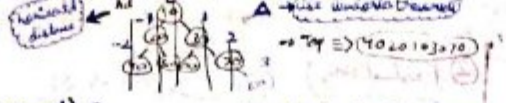
```
Class Node {
    Public:
    int data;
    Node* left;
    Node* right;

    Node (int data) {
        this -> data = data;
        this -> left = NULL;
        this -> right = NULL;
    }
};

int findPosition (int arr[], int n, int element) {
    for( int i=0; i<n ; i++) {
        if (arr[i] == element) {
            return i;
        }
    }
    return -1;
}

// build tree from inorder and preorder traversal
Node* buildTreeFromPreorderInorder (int inorder[], int preorder[], int size,
int & preIndex, int inorderStart, int inorderEnd) {
    // base case
    if(preIndex >= size || inorderStart > inorderEnd) {
        return NULL;
    }

    // Step A:
    int element = preorder[preIndex++];
    Node* root = new Node (element);
    int pos = findPosition (inorder, size, element);

    // Step B: solve → root -> left
    root -> left = buildTreeFromPreorderInorder(inorder, preorder, size,
                                preIndex, inorderStart, pos-1);

    // Step C: solve → root -> right
    root -> right = buildTreeFromPreorderInorder(inorder, preorder, size,
                                preIndex, pos+1, inorderEnd);

    return root;
}

int main () {
    int inorder[] = {40, 20, 50, 10, 60, 30, 70};
    int preorder[] = {10, 20, 40, 50, 30, 60, 70};
    int size = 7;
    int preIndex = 0;
    int inorderStart = 0;
    int inorderEnd = size-1;

    Node* root = buildTreeFromPreorderInorder(inorder, preorder, size,
                                preIndex, inorderStart, inorderEnd);

    levelOrderTraversal(root);

    return 0;
}
```

⊛ i/p → inorder (LNR)
↳ Postorder (LRN)

o/p → buildTree

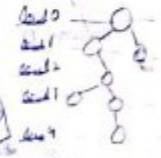
```
Node* buildFromPostorder( int inorder[ ], int postorder[
    int & postIndex, int size, int inorderSt
    int inorderEnd) {

    // base case
    if( postIndex < 0 || inorderStart > inorderEnd) {
        return NULL;
    }

    // A
    int element = postorder[ postIndex];
        postIndex - - ;
    Node* root = new Node(element);
    int pos = findPosition(inorder, size, element,

    // root -> right
    root -> right = buildTreeFromPostorderInorder(
            postorder, postIndex, size, pos+1,
                inorderEnd);

    // root -> left
    root -> left = buildTreeFromPostorderInorder
            inorder, postorder, postIndex,
            inorderStart, pos-1) ;

    return root;
```

Basic-4 VIEW



```
• void PrintTopView(Node* root) {
    if(root == NULL)
        return;
    // create a map for storing Horizontal distance -> TopNode -> data
    map<int, int> topNode;

    // Level order
    // We will store a pair consisting of Node and Horizontal distance
    queue<pair<Node*, int>> q;
    q.push(make_pair(root, 0));
    while(!q.empty()) {
        pair<Node*, int> temp = q.front();
        q.pop();

        Node* frontNode = temp.first;
        int hd = temp.second;
        // To kiski horizontal distance aaya h, check if answer for that hd already exists or not
        if(topNode.find(hd) == topNode.end()) {
            // create entry
            topNode[hd] = frontNode -> data;
        }
        if(frontNode -> left)
            q.push(make_pair(frontNode -> left, hd-1));
        if(frontNode -> right)
            q.push(make_pair(frontNode -> right, hd+1));
    }
    // ab aapka answer store hua hoga aapke map me
    for(auto i : topNode) {
        cout << i.first << "->" << i.second;
    }
}
```

```
• Void PrintRightView(Node* root,
        vector<int> &ans, int level) {
    // Base case
    if(root == NULL)
        return;
    if(level == ans.size()) {
        ans.push_back(root->data);
    }
    // right
    PrintRightView(root->right, ans, level+1);
    // left
    PrintRightView(root->left, ans, level+1);
```



```
• Void PrintLeftView(Node* root,
        vector<int> &ans, int level
    // base case
    if(root == NULL)
        return;
    if(level == ans.size()) {
        ans.push_back(root->data);
    }
    // left
    PrintLeftView(root->left, ans, level+1);
    // Right
    PrintLeftView(root->right, ans, level+1);
}
```

```
• void PrintLeftBoundary(Node* root) {
    // base case
    if(root == NULL)
        return;
    // if root is leaf node, then go back
    if(root->left == NULL && root->right == NULL) {
        return;
    }
    cout << root->data;
    if(root->left) {
        PrintLeftBoundary(root->left);
    }
    else {
        PrintLeftBoundary(root->right);
    }
}

• Void PrintLeafBoundary(Node* root) {
    // base case
    if(root == NULL)
        return;
    if(root->left == NULL && root->right == NULL)
        cout << root->data;
    PrintLeafBoundary(root->left);
    PrintLeafBoundary(root->right);
}

• Void PrintRightBoundary(Node* root) {
    // base case
    if(root == NULL)
        return;
    if(root->left == NULL && root->right == NULL)
        return;
    if(root->right) {
        PrintRightBoundary(root->right);
    }
    else {
        PrintRightBoundary(root->left);
    }
    cout << root->data;
}

• Void boundaryTraversal(Node* root) {
    if(root == NULL)
        return;
    cout << root->data;
    PrintLeftBoundary(root->left);
    PrintLeafBoundary(root);
    PrintRightBoundary(root->right);
```

Questions [INTERVIEW]

1) (Fastest way) → de hain ab, ek hi baar traversal me track karlo (O(n))

Diameter of BT

```
int D = 0;
int height(TreeNode* root) {
    if (!root) return 0;
    int lh = height(root -> left);
    int rh = height(root -> right);
    // diameter
    int currD = lh + rh;
    D = max(D, currD);
    return max(lh, rh) + 1;
}
int diameterOfBinaryTree(TreeNode* root) {
    height(root);
    return D;
}
};
```

2) (Fastest way) → (Recursion Tab ko Bottom tab hi Track Karte Hain) O(n)

Height Balanced Tree? → (lh − rh ≤ 1)

```
bool isBalanced = true;
int height(TreeNode* root) {
    if(!root) return 0;
    int lh = height(root -> left);
    int rh = height(root -> right);
    // check for current node, is it balanced?
    if(isBalanced && abs(lh - rh) > 1) {
        isBalanced = false;
    }
    return max(lh, rh) + 1;
}
bool isBalanced(TreeNode* root) {
    height(root);
    return isBalanced;
}
};
```

(M-1)

approach → (Right Tab to ... d = 0 ... Left ... right D = D+1 ... but ...)

3) **Same Tree** [redacted] → (Same structure Same Value Checking)

(done to Both-Side traversal Karke data Merge)

```
bool isSameTree(TreeNode* p, TreeNode* q) {
    if(!p && !q)
        return true;
    if(p && q)
        return (p -> val == q -> val)
        && isSameTree(p -> left, q -> left)
        && isSameTree(p -> right, q -> right);
    return false;
};
```

(8) Same approach — (Mirror Tree?)

```
bool (p -> left, q -> right)
&& (p -> right, q -> left);
```

4) **Diagonal Traversal**

```
vector<int> diagonal(Node* root) {
    vector<int> ans;
    if(!root) return ans;
    queue<Node*> q;
    q.push(root);
    while (!q.empty()) {
        Node* temp = q.front();
        q.Pop();
        while(temp) {
            ans.push_back(temp -> data);
            if (temp -> left) {
            // bond one data Merge
            q.push(temp -> left);
            }
            temp = temp -> right;
        }
    }
    return ans;
```

op → 8 10 14 3 6 7 13 1 4

(M-2) using level Trave...

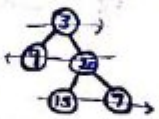
```
temp → 8
while(temp != NULL) {
    Print → temp -> val
    if (temp -> left) q.push
    temp = temp -> right
}
```

⇒ 8 10 14 3 6 7 4

5) **Binary Tree Zigzag Level order Traversal**

ip → root = [3, 9, 20, null, null, 15, 7]

op → [[3], [20, 9], [15, 7]]



```
vector<vector<int>> zigzag(TreeNode* root) {
    vector<vector<int>> ans;
    if(!root) return ans;
    bool LtoRdir = true;
    queue<TreeNode*> q;
    q.push(root);
    while (!q.empty()) {
        int width = q.size();
        vector<int> oneLevel(width);
        for(int i = 0; i < width; i++) {
            TreeNode* front = q.front();
            q.Pop();
            int index = LtoRdir ? i : width - i - 1;
            oneLevel[index] = front -> val;
            if (front -> left)
                q.push(front -> left);
            if (front -> right)
                q.push(front -> right);
        }
        // toggle the direction
        LtoRdir = !LtoRdir;
        ans.push_back(oneLevel);
    }
    return ans;
}
```

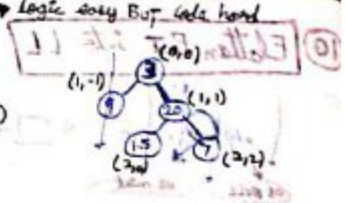
```
if (p == NULL && q == NULL)
    return true;
if (p == NULL || q == NULL)
    return false;
if (p -> val == q -> val)
    return isSame(p -> right, q -> right) && isSame(
return false;
```

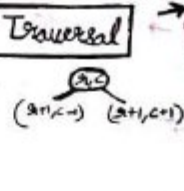
⑥ Convert to Sum Tree

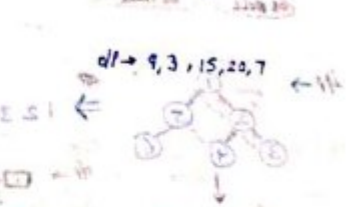
```
int sum(Node* root){
    if(!root) return 0;
    if(!root->left && !root->right){
        int temp = root->data;
        root->data = 0;
        return temp;
    }
    int lsum = sum(root->left);
    int rsum = sum(root->right);
    int temp = root->data;
    root->data = lsum + rsum;
    return root->data + temp;
}
void toSumTree(Node *node){
    sum(node);
}
};
```

⑦ Vertical order Traversal

→ Logic easy But code hard



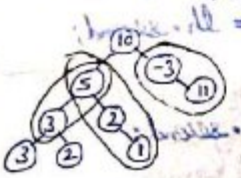




⑧ K-Sum Paths

i/p → TargetSum = 8

∴ o/p → 3

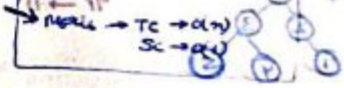


```
int ans = 0;
void PathFromOneRoot(TreeNode* root, int sum){
    if(!root) return;
    if(sum == root->val){
        ++ans;
    }
    PathFromOneRoot(root->left, sum - root->val);
    PathFromOneRoot(root->right, sum - root->val);
}

int PathSum(TreeNode* root, int targetSum){
    if(root){
        PathFromOneRoot(root, targetSum);
        PathSum(root->left, targetSum);
        PathSum(root->right, targetSum);
    }
    return ans;
}
};
```

⑨ Morris Traversal

Inorder me → TC → O(n), SC → O(n)

Morris → TC → O(n), SC → O(1)



```
vector<int> inorderTraversal(TreeNode* root){
    vector<int> ans;
    TreeNode* curr = root;
    while(curr){
        // left node is NULL, then visit it and go right
        if(curr->left == nullptr){
            ans.push_back(curr->val);
            curr = curr->right;
        }
        // left node is Not NULL
        else{
            // find inorder predecessor
            TreeNode* pred = curr->left;
            while(pred->right != curr && pred->right){
                pred = pred->right;
            }
            // if pred right node is NULL, then go left after establishing link from pred to curr.
            if(pred->right == nullptr){
                pred->right = curr;
                curr = curr->left;
            }
            else{
                // left is already visited, go after visiting curr node while removing the link
                pred->right = nullptr;
                ans.push_back(curr->val);
                curr = curr->right;
            }
        }
    }
    return ans;
}
```

⑩ Flatten BT into LL

i/p →

⇒ 1 2 3 4 5 6 (In-order)

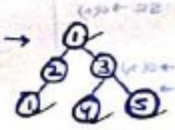
o/p → [1] → [2] → ... [6]

```
void flatten(TreeNode* root){
    TreeNode* curr = root;
    while(curr){
        if (curr->left){
            TreeNode* Pred = curr->left;
            while (Pred->right)
                Pred = Pred->right;
            Pred->right = curr->right;
            curr->right = curr->left;
            curr->left = nullptr;
        }
        curr = curr->right;
    }
}
```

```
Pred->right = curr->right
curr->right = curr->left
curr->left = NULL
curr = curr->right
```

⑪ Maximum Sum of Non Adjacent Nodes

i/p →



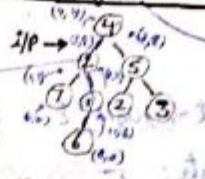
o/p → 11

⇒ Use Recur Approach → Include/Exclude

```
pair<int, int> getMaxSum_helper(Node* root) {
    if(!root) return {0,0};
    auto left = getMaxSum_helper(root->left);
    auto right = getMaxSum_helper(root->right);
    // Sum including the node
    int a = root->data + left.second + right.second;
    // Sum excluding the node
    int b = max(left.first, left.second) + max(right.first,
                                               right.second);
    return {a,b};
}

int getMaxSum(Node* root) {
    auto ans = getMaxSum_helper(root);
    return max(ans.first, ans.second);
};
```

⑫ Sum of the Longest Bloodline of a Tree

i/p →

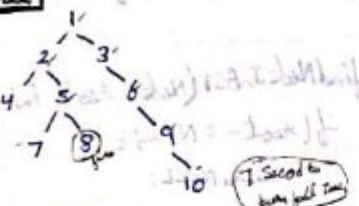


longest path → Root to leaf

o/p → 13

```
pair<int, int> height(Node* root){
    if(!root) return {0,0};
    auto lh = height(root->left);
    auto rh = height(root->right);
    int sum = root->data;
    if (lh.first == rh.first) {
        sum += lh.second > rh.second
            ? lh.second : rh.second;
    }
    else if(lh.first > rh.first) {
        sum += lh.second;
    }
    else {
        sum += rh.second;
    }
    return {max(lh.first, rh.first)+1,
            sum};
}

int SumofLongestToleafPath(Node* root) {
    auto h = height(root);
    return h.second;
};
```

(13) Burning Tree



Target Node = 8

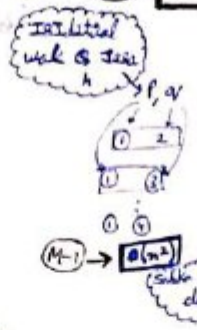
o/p → 7

① Find Target Node
② Make Node to Parent Node Mapping (Right upar Tree ka koi bacha nhi bcha h Tree me)
③ Time for Burn the Tree

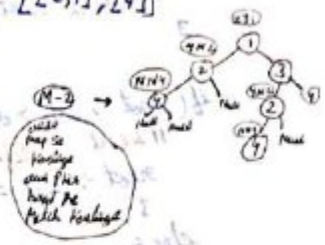
```
int burnTheTree(Node targetNode, unordered_map<Node*, Node*> & parentMap) {
    unordered_Map<Node*, bool> isBurnt;
    queue<Node*> q; // currently set on fire nodes
    int T=0;
    q.push(&targetNode);
    isBurnt[&targetNode] = true;
    while(!q.empty()) {
        int size = q.size();
        bool isFireSpreaded = false;
        for(int i=0; i<size; i++) {
            Node* front = q.front();
            q.Pop();
            if (front->left && !isBurnt[front->left]) {
                q.Push(front->left);
                isBurnt[front->left] = true;
                isFireSpreaded = true;
            }
            if(front->right && !isBurnt[front->right]) {
                q.Push(front->right);
                isBurnt[front->right] = true;
                isFireSpreaded = true;
            }
            if (parentMap[front] && !isBurnt[parentMap[front]]) {
                q.Push(parentMap[front]);
                isBurnt[parentMap[front]] = true;
                isFireSpreaded = true;
            }
        }
        if (isFireSpreaded) {
            ++T;
        }
    }
    return T;
}
```

```
Node* makeNodeToParentMappingAndFindingTargetNode(Node* root, unordered_map<Node*, Node*>& parentMap,
                                                  int target) {
    Node* targetNode = nullptr;
    queue<Node*> q;
    q.Push(root);
    parentMap[root] = nullptr;
    while(!q.empty()) {
        Node* front = q.front();
        q.Pop();
        if (front->data == target) {
            targetNode = front;
        }
        if(front->left) {
            q.Push(front->left);
            parentMap[front->left] = front;
        }
        if (front->right) {
            q.Push(front->right);
            parentMap[front->right] = front;
        }
    }
    return targetNode;
}
```

(14) Find Duplicate Subtree



o/p → [[2,4],[4]]




```
vector<TreeNode*> ans;
unordered_map<string, int> SubTreeMap;
String Preorder(TreeNode* root) {
    if(!root) return "N";

    String curr = to_string(root->val);
    String l = Preorder(root->left);
    String r = Preorder(root->right);
    String s = curr + "," + l + "," + r;
    if (SubTreeMap.find() == SubTreeMap.end()) {
        if (SubTreeMap[s] == 1)
            ans.push_back(root);
        SubTreeMap[s]++;
    }
    else {
        SubTreeMap[s] = 1;
    }
    return s;
```

10 - BST

(Inorder = Sorted)

Left side data & Right side bada

(Inorder of BST is always sorted)

```
• Node * insertIntoBST(Node* root, int data){
    if(root==NULL){
    // this is the first node we have to create
      root = new Node(data);
      return root;
    }

    if(root -> data > data){
      // insert in left
      root -> left = insertIntoBST(root -> left, data);
    }
    else{
      // insert into right
      root -> right = insertIntoBST(root -> right, data);
    }
    return root;
  }

• void takeInput(Node * & root){
    int data;
    cin >> data;
    while(data != -1){
      root = insertIntoBST(root, data);
      cin >> data;
    }
  }

• Node* deleteNodeInBST(Node* root, int target){
    if(root==NULL){
      return NULL;
    }
    if(root -> data == target){
    // 4 case
      if(root -> left == NULL && root -> right == NULL){
        return NULL;
      }
      else if(root -> left == NULL && root -> right != NULL){
        Node* child = root -> right;
        return child;
      }
      else if(root -> left != NULL && root -> right == NULL){
        Node* child = root -> left;
        return child;
      }
      else{
        int inorderPre = maxVal(root -> left); // find inorder Predecessor in left subtree
        root -> data = inorderPre   // replace root -> data value with inorder Predecessor
        root -> left = deleteNodeInBST(root -> left, inorderPre); // delete inorder predecessor from left subtree
        return root;
      }
    }
    else if(target > root -> data){
      root -> right = deleteNodeInBST(root -> right, target);
    }
    else if(target < root -> data){
      root -> left = deleteNodeInBST(root -> left, target);
    }
    return root;
  }

• Node* findNodeInBST(Node* root, int target){
    if(root==NULL){
      return NULL;
    }
    if(root -> data == target)
      return root;
    if(target > root -> data){
      return findNodeInBST(root -> right, t
    }
    else{
      return findNodeInBST(root -> left, t
    }
  }

• int minVal(Node * root){
    Node* temp = root;
    if(temp == NULL){
      return -1;
    }
    while(temp -> left != NULL){
      temp = temp -> left;
    }
    return temp -> data;
  }

• int maxVal(Node * root){
    Node* temp = root;
    if(temp == NULL){
      return -1;
    }
    while(temp -> right != NULL){
      temp = temp -> right;
    }
    return temp -> data;
```

max in left tree
or
min in right tree

BST using Inorder

```
Node* bstUsingInorder(int inorder[], int s, int e) {
    if (s>e) {
        return NULL;
    }
    int mid = (s+e)/2;
    int element = inorder[mid];
    Node* root = new Node(element);
    root -> left = bstUsingInorder(inorder, s, mid-1);
    root -> right = bstUsingInorder(inorder, mid+1, e);
    return root;
}
```

Lowest Common Ancestors

```
Node* lowestCommonAncestor(Node* root, Node* p, Node* q) {
    if (root == NULL)
        return NULL;
// case 1
    if (p->val < root->val && q->val < root->val) {
        return lowestCommonAncestor(root->left, p, q);
    }
// case 2
    else if (p->val > root->val && q->val > root->val) {
        return lowestCommonAncestor(root->right, p, q);
    }
// case 3
    else if (p->val < root->val && q->val > root->val) {
        return root;
    }
// case 4
    else if (q->val < root->val && p->val > root->val) {
        return root;
    }
}
};
```

2-Sum Target (BST)

```
void StoreInorder(Node* root, vector<int>& inorder) {
    if (root == NULL)
        return;
    StoreInorder(root->left, inorder);
    inorder.push_back(root->val);
    StoreInorder(root->right, inorder);
}

bool findTarget(Node* root, int k) {
    vector<int> inorder;
    StoreInorder(root, inorder);
    int s = 0;
    int e = inorder.size() - 1;
    while(s<e) {
        int sum = inorder[s] + inorder[e];
        if (sum == k)
            return true;
        if (sum > k)
            e--;
        else
            s++;
    }
    return false;
```

Valid BST

```
bool solve(Node* root, int lb, int ub) {
    if (root == NULL)
        return true;
    if (root->data > lb && root->data < ub) {
        bool leftAns = solve(root->left, lb, root->data);
        bool rightAns = solve(root->right, root->data, ub);
        return leftAns && rightAns;
    }
    else {
        return false;
    }
}
bool isValidBST(Node* root) {
    int lowerbound = INT_MIN;
    int upperbound = INT_MAX;
    bool ans = solve(root, lowerbound, upperbound);
    return ans;
}
};
```

K^{th} Smallest of BST

```
int KthSmallest(Node* root, int &K) {
    if (root == NULL) {
        return -1;
    }
// Inorder
// left
    int leftAns = KthSmallest(root->left, K);
    if (leftAns != -1)
        return leftAns;
// N -> root
    K--;
    if (K==0) {
        return root->val;
    }
// right
    int rightAns = KthSmallest(root->right, K);
    return rightAns;
}
```

#3

BST to Sorted Doubly LL

```
void ConvertIntoSortedDLL(Node* root, Node* $head) {
    // base case
    if(root == NULL) {
        return;
    }
    // right subtree ko into LL
    ConvertIntoSortedDLL(root -> right, head);

    // attach root node
    root -> right = head;

    if(head != NULL)
        head -> left = root;
    // update head
    head = root
    // left subtree linked list
    ConvertIntoSortedDLL(root -> left, head);
}
```

Sorted LL into BST

```
Node* sortedLL intoBST(Node * $head, int n) {
    // base case
    if( n <= 0 || head == NULL)
        return NULL;

    Node * leftSubtree = sortedLL intoBST(head, n/2);
    // ab head mera usage kaafi hai, head ko mid ko usage karna (root head left h)
    Node* root = head;
    root -> left = leftSubtree;

    head = head -> right;
    // right bat solve karna h
    root -> right = sortedLL intoBST(head, n-1-n/2)
    return root;
}
```

FindLargestBST in Binary Tree

```
Node* findLargestBST( Node * root, int $ ans) {
    if(root == NULL) {
        Node* temp = new Node( 0, INT_MIN, INT_MAX, true);
        return temp;
    }

    Node* leftKaAns = findLargestBST( root -> left, ans);
    Node* rightKaAns = findLargestBST( root -> right, ans);
    // checking starts here
    Node* currNodeKaAns = new Node();

    currNodeKaAns -> size = leftKaAns -> size + rightKaAns -> size + 1;
    currNodeKaAns -> maxVal = max( root -> data , rightKaAns -> maxVal);
    currNodeKaAns -> minVal = min( root -> data, leftKaAns -> minVal);

    if( leftKaAns -> validBST && rightKaAns -> validBST && ( root -> data > leftKaAns -> maxVal &&
        root -> data < rightKaAns -> minVal)) {
        currNodeKaAns -> validBST = true;
    }
    else {
        currNodeKaAns -> validBST = false;
    }

    if (currNodeKaAns -> validBST) {
        ans = max(ans, currNodeKaAns -> size);
    }
    return currNodeKaAns;
}
```

Bottom up approach

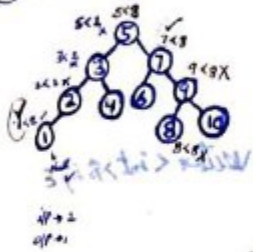
Questions

① Inorder Predecessor in BST

(M-1) → • Store inorder in vector & search just smaller than x.

(M-2) → • Perform inorder traversal using R.E & find value just lesser than x.

(M-3) → • use BST method



i/p → 8
o/p → 7

Just iska ulta approach

```
Node* findPredecessor(Node* root, Node* p) {
    Node* Pred = 0;
    Node* curr = root;
    while (curr) {
        if (curr -> data < (p -> data)) {
            Pred = curr;
            curr = curr -> right;
        }
        else {
            curr = curr -> left;
        }
    }
    return Pred;
}
```

② Inorder Successor in BST

(M1) → • Store inorder in vector & search just greater than x

(M2) → • Perform inorder traversal & we will find first value > x will be the ans

(M3) → use BST method.

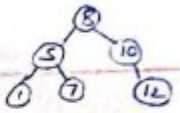
```
Node* inorderSuccessor(Node* root, Node* x) {
    Node* succ = 0;
    Node* curr = root;
    while (curr) {
        if (curr -> data > x -> data) {
            succ = curr;
            curr = curr -> left;
        }
        else {
            curr = curr -> right;
        }
    }
    return succ;
}
```

③ Build BST using Preorder Traversal

(M-1) → Brute force

8	5	1	7	10	12

(has Node ko liye check kaunga)



→ $O(n^2)$

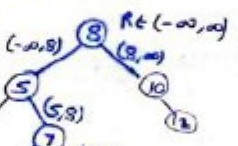
ek ek to dubara se check karnaa

(M-2) → Preorder diyaa, to ab Main inorder nikal lunga (direct sort karke)

↳ ab ye done se BST bna lunga

(M-3)

0	1	2	3	4	5
8	5	1	7	10	12



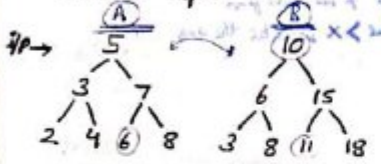
→ (Range ka use karke Main $O(n)$ me karlunga dubara se str dekhne padega)

→ $O(n)$

```
TreeNode* build(int& i, int min, int max,
                vector<int>& Preorder) {
    if (i >= Preorder.size()) {
        return nullptr;
    }
    TreeNode* root = nullptr;
    if (Preorder[i] > min && Preorder[i] < max) {
        root = new TreeNode(Preorder[i++]);
        root -> left = build(i, min, root->val, Preorder);
        root -> right = build(i, root->val, max, Preorder);
    }
    return root;
}

TreeNode* bstFromPreorder(vector<int>& Preorder) {
    int min = INT_MIN, max = INT_MAX;
    int i = 0;
    return build(i, min, max, Preorder);
}
```

④ Brothers from Different Roots

Given two BSTs containing N1 and N2 distinct nodes respectively and given a value x. Your task is to complete the function countPairs(), that returns the count of all pairs from both the BSTs whose sum is equal to x.



i/p → 3

$x = 16$

o/p → (5,11), (6,10), (8,8)

M1 → Brute force, A ke node se 5 liya, (16-5) = 11 ab 11 search kiya BST-2 me, mila to count ++; → $O(n^2)$

M2 → BST1 → inorder → 2, 3, 4, 5, 6, 7, 8 (i)

BST2 → opp. inorder → 18, 11, 15, 10, 8, 6, 3 (j)

ab hamara work

20 > 16 ∴ j--

15 < 16 i++

16 == 16 Store data; i++; j--;

M-3 → ab hum space ka ka usko liye ye method,

ab BST1 → normal inorder traversal kardunga

BST2 → ulta inorder traversal kardunga (and wahi M-2 ka approach)

(Inorder using Stack)

(using stack)

while(a){ Stack.push(a); a = a->left; }

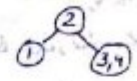
left kaate time stack me daalo (left inorder)

⑤ Balance a BST

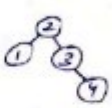
↳ • Inorder nikal lo,

1 2 3 4 → mid ko root node kardunga

(mid)



(call Recursively)



```
void inorder(TreeNode* root, vector<int>& in){
    if(!root) return;
    inorder(root->left, in);
    in.push_back(root->val);
    inorder(root->right, in);
}

TreeNode* buildTree(vector<int>& in, int start,
                    int end){
    if(start > end) return NULL;
    int mid = (start + end)>>1;
    TreeNode* root = new TreeNode(in[mid]);
    root->left = buildTree(in, start, mid-1);
    root->right = buildTree(in, mid+1, end);
    return root;
}

TreeNode* balanceBST(TreeNode* root){
    vector<int> in;
    inorder(root, in);
    // build balanced BST using inorder traversal
    return buildTree(in, 0, in.size()-1);
};
```

```
int countPairs(Node* root1, Node* root2, int x){
    int ans = 0;
    Stack<Node*> S1, S2;
    Node* a = root1;
    Node* b = root2;
    while(1){
        while(a){
            // inorder
            S1.push(a);
            a = a->left;
        }
        while(b){
            // reverse inorder
            S2.push(b);
            b = b->right;
        }
        if(S1.empty() || S2.empty()){
            break;
        }
        auto atop = S1.top();
        auto btop = S2.top();
        int sum = atop->data + btop->data;
        if(sum == x){
            ++ans;
            S1.pop();
            S2.pop();
            a = atop->right;
            b = btop->left;
        }
        else if(sum < x){
            S1.pop();
            a = atop->right;
        }
        else{
            S2.pop();
            b = btop->left;
        }
    }
    return ans;
}
```

⑥ Find Median of BST

↳ inorder initial se mid initial tk → (overall depth ke)

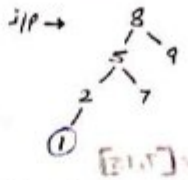
Traversal

	TC	SC
• RE →	$O(n)$	$O(n)$
• Morris →	$O(n)$	$O(1)$

• no of nodes
• desired element

⑦ Check whether BST has Dead End

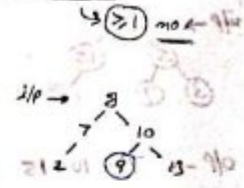
i/p →



o/p → Yes ('1' is dead end)

↳ (≥1) no ←

i/p →



o/p → Yes ('9' is dead end)

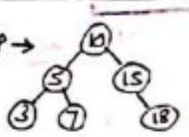
dead end exist ←

- only for leaf node
- if already $(x+1)$, $(x-1)$ exist

```
void fun(Node* root, unordered_map<int, bool>& visited,
                                    bool &ans) {
    if (root == 0) return;
    // visit the node
    visited[root -> data] = 1;
    if (root -> left == 0 && root -> right == 0) {
    // Leaf node
    int xp1 = root -> data + 1;
    int xm1 = root -> data - 1 == 0 ?
                root -> data : root -> data - 1;
    if (visited.find(xp1) != visited.end() &&
                visited.find(xm1) !=
                visited.end()) {
            ans = true;
            return;
    }
    fun(root -> left, visited, ans);
    fun(root -> right, visited, ans);
}

bool isDeadEnd(Node * root) {
    bool ans = false; // tells if there was a DE
    unordered_map<int, bool> visited;
    fun(root, visited, ans);
    return ans;
}
```

⑧ Count BST Nodes Lying in Range

i/p →



, low = 7
high = 15

o/p → 7 + 10 + 15 = 32 in range [7, 15]

M1 → Simple inorder traversal ✓ (Ye bhi kar sakte ho Anish)

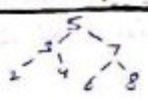
M2 → 5 < 7 to right Jao (left Not Jao)
15 > 15 to left Jao (right Not Jao)
↓

```
int rangeSumBST(TreeNode * root, int low, int high) {
    if(!root) return 0;
    int ans = 0;

    bool WasInRange = false;
    if(root -> val >= root -> val <= high) {
        WasInRange = true;
        ans + = root -> val;
    }
    if(WasInRange) {
        ans + = rangeSumBST(root -> left, low, high)
                + rangeSumBST(root -> right, low, high);
    }
    else if(root -> val < low) {
        ans + = rangeSumBST(root -> right, low, high);
    }
    else if (root -> val > high) {
        ans + = rangeSumBST(root -> left, low, high);
    }
    return ans;
}
}
```

⑨ Flatten BST to Sorted LL

(Some last chapter)

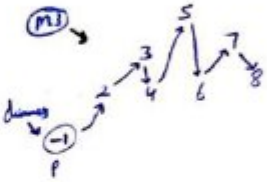
left NULL ko aur Right Next Ko



M1 → BST to [redacted] inorder so LL bna do → TC = [redacted]

* M2 → inorder in Node* vector & create the list

M3 →



Prev → left = Null
Prev → right = root
Prev = root

```
void in(node * root, node* & Prev) {
    if (!root) return;
    in(root -> left, Prev);
    Prev -> left = 0;
    Prev -> right = root;
    Prev = root;
    in(root -> right, Prev);
}

node * flatten(node* root) {
    node* dummy = newNode(-1);
    node* Prev = dummy
    in (root, Prev);
    Prev -> left = Prev -> right = 0;
    root = dummy -> right;
    return root;
}
```

⑩ Replace elements with least Greater Elements to its Right

i/p → arr[] = {8, 58, 71, 18, 31, 32, 63, 92, 43, 3, 91, 93, 25, 80, 28}
o/p → [redacted]
{18, 63, 80, 25, 32, 43, 80, 93, 80, 25, 93, -1, 28, -1, -1}

M1 → 2 for loop → O(n²)

Best bhi → inorder Successor BST

M2 → ① R → L array
② have Right ko dal do

(case wise bhi kar)

↓

```
Node* insert (Node* root, int val, int & succ) {
    if (!root) {
        return new Node (val);
    }
    if (val >= root -> data) {
        root -> right = insert(root -> right, val, succ);
    }
    else {
        succ = root -> data;
        root -> left = insert (root -> left, val, succ);
    }
    return root;
}

vector<int> findLeastGreater(vector<int> & arr, int n) {
    vector<int> ans(arr.size(), -1);
    Node * root = nullptr;
    int succ = -1;
    for(int i = arr.size() - 1; i >= 0; i--)
        root = insert(root, arr[i], succ);
        ans[i] = succ;
    }
    return ans;
}
};
```

① Valid BST from Preorder

Valid BST i/p → 7, 7, 10, 10, 9, 5, 2, 8
o/p → 0

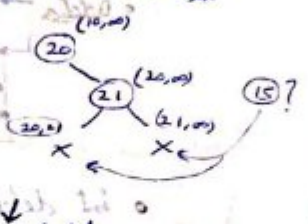
Duplicate not allowed in BST
∴ o/p → 0

Valid BST i/p → 10, 8, 7, 9, 20, 15, 21 (Preorder)
o/p → 1

Valid
i/P → 10, 8, 7, 9, 20, 21, 15
o/P → 0

↓

The 15 'i' lost & seek inorder



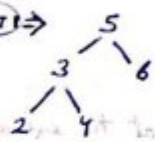
↓

Not Possible

```
void build (int & i, int min, int max,
            vector<int> & A) {
    if (i >= A.size()) {
        return;
    }
    if (A[i] > min && A[i] < max) {
        int rootData = A[i++];
        build(i, min, rootData, A);
        build(i, rootData, max, A);
    }
}

int Solution (vector<int> & A) {
    int min = INT_MIN, max = INT_MAX;
    int i = 0;
    build(i, min, max, A);
    return i == A.size();
}
```

⑫ Merge Two BST

i/P → BST 1 ⇒



BST 2 →

o/P → 1 2 2 3 3 4 5 6 6 7

M1 → Inorder 1 & 2 → then (sorted 2 array to merge knowledge) of BST

M2 → Inorder traversal using stack & keeping two pointer & merge & make an undo on go

Begin from diff root

Pre-merge

```
vector<int> ans;   // stores sorted elements of both the trees
Stack<Node*> sa, sb;
Node* a = root1, b* = root2;
while (a || b || !sa.empty() || !sb.empty()) {
    while (a) {
        sa.push(a);
        a = a->left;
    }
    while (b) {
        sb.push(b);
        b = b->left;
    }
    if (sb.empty() || (!sa.empty() && sa.top()->data <= sb.top()->data))
        auto atop = sa.top();
        ans.push_back(atop->data);
        sa.top();
        a = atop->right;
    }
    else {
        auto btop = sb.top();
        sb.pop();
        ans.push_back(btop->data);
        b = btop->right;
    }
}
return ans;
}
```

11 - Heaps

Parent = i
L.C = 2*i
R.C = 2*i+1

1
2 3
4 5 6 7

Max - heap
Min - heap

→ (Complete Binary Tree + holds a heap property) → Max / Min

• Hum STL heap use Kar lenge,

Index = i
Parent = i/2

• (visualize hum tree se Karenge, lekin solve hum array se Karenge)

• Insertion
→ • add node at the end of array
• Sab Tayah Pa lekar Jao

```
# include <iostream>
using namespace std;
class Heap {
    Public:
      int arr[101];
      int size;
    Heap() {
      size = 0;
    }
    void insert(int value) {
    // value insert Karo and ma
      size = size + 1;
      int index = size;
      arr[index] = value;
    // is value Ko Sab Tayah leJao
      while (index > 1) {
          int ParentIndex = index/2;
          if (arr[index] > arr[ParentIndex]) {
              swap(arr[index], arr[ParentIndex]);
              index = ParentIndex;
          }
          else {
              break;
          }
      }
}

int main() {
    Heap h;

    h.insert(50);
    h.insert(30);
    h.insert(70);

    for (int i = 0; i <= h.size(); i++) {
        cout << h.arr[i];
```

• Deletion → Homesha (top node) root node delete hoga
→ • last value Ko replace with root node
• Place the root node in correct position

```
• int delete() {
    int ans = arr[1];
    // replace root node value with last node data
    arr[1] = arr[size];
    size--;
    // Place root node Ko data on its correct Position
    int index = 1;
    while (index < size) {
        int left = 2* index;
        int right = 2* index+1;
        int largest = index;
        if (left < size && arr[largest] < arr[left]) {
            largest = left;
        }
        if (right < size && arr[largest] < arr[right]) {
            largest = right;
        }
        if (largest == index) {
            break;
        }
        else {
            swap (arr[index], arr[largest]);
            index = largest;
        }
    }
    return ans;
}
```

→ Sab Tayah h

• Kisi bhi array ko heap me badal sakte h using → heapify (arr, n, i)

→ delete wala code to kiya the usi ke liye h but recursion use kiya h (loop ke liyeh)

(VI)

• Void heapify (int arr[], int n, int i) {

```
    int index = i;
    int leftIndex = 2*i;
    int rightIndex = 2*i+1;
    int largest = index;
  if (leftIndex < n && arr[largest] < arr[leftIndex]) {
        largest = leftIndex;
    }
    if (rightIndex < n && arr[largest] < arr[rightIndex]) {
        largest = rightIndex;
    }

    if (index != largest) {
      // left ya right child me se koi > hogya currentNode se
      Swap (arr[index], arr[largest]);
        index = largest;
        heapify (arr, n, index);
```

(1st node ko bug sahi jagah chodna h)

Build Heap && heapSort

• Void buildHeap (int arr[], int n)

```
    for (int i = n/2; i > 0; i--) {
        heapify (arr, n, i);
    }
```

→ array to Heap (bas n/2 se upar ko heapify)

leaf node ko heapify karne ki jaruraat nhi h (kyuki wo to already sahi jagah par h)

• Void heapSort (int arr[], int n) {

```
    while (n != 1) {
        swap(arr[1], arr[n--]);
        heapify (arr, n, 1);
    }
}
```

(42) • Priority queue → STL ka (max/min heap ka use karke)

STL-Basics

```
# include <iostream>
# include <queue>
using namespace std;

int main() {
    // max-heap
    Priority_queue <int> pq;

    pq.push(3);
    pq.push(6);
    pq.push(9);
    pq.push(4);

cout << pq.top();
pq.pop();
cout << pq.top();

cout << pq.size();

if (pq.empty()) {
    cout << "Max heap is empty";
}
else {
    cout << " Not empty";
}
```

```
# include <iostream>
# include <queue>
using namespace std;

int main() {
    // min-heap
    Priority_queue <int, vector <int>, greater <int> > pq;

    pq.push(3);
    pq.push(6);
    pq.push(9);
    pq.push(4);

cout << pq.top();
pq.pop();
cout << pq.top();
cout << pq.size();

if (pq.empty()) {
    cout << "Min heap is empty";
}
else {
    cout << " Not empty";
}
```

(43)

K^th Smallest no.

```
# include <iostream>
# include <queue>
using namespace std;

int getKthLargestElement(int arr[], int n, int k) {
    // create min heap
        Priority_queue < int, vector <int>, greater < int >> pq;
    // insert initial K elements of array
    for( int i = 0; i < k; i++) {
        pq.push (arr[i]);
    }
    // for remaining elements, push only if they are greater than top
        for( int i = k; i < n; i++) {
            int element = arr[i];
            if (element > pq.top()) {
                pq.pop();
                pq.push(element);
            }
        }
        int ans = pq.top();
        return ans;
}
```

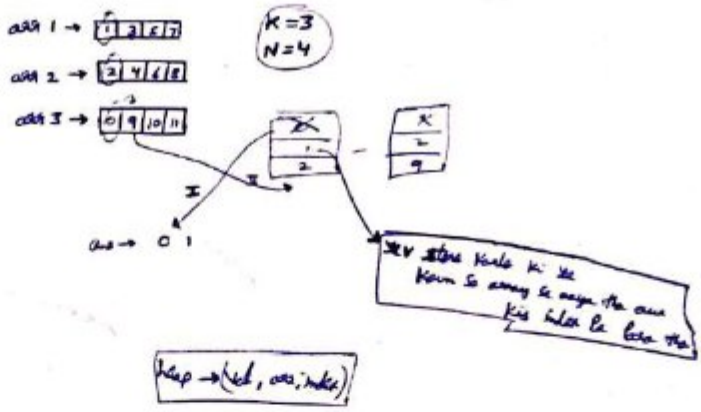
```
int getKthSmallestElement( int arr[], int n, int k)
    // create max heap
        Priority_queue < int > pq;
    // insert initial K elements of array
    for( int i = 0; i < k; i++) {
        pq.push (arr[i]);
    }
    // for remaining elements, push only if they are less than top
    for (int i = k; i < n; i++) {
        int element = arr[i];
        if (element < pq.top()) {
            pq.pop();
            pq.push(element);
        }
    }
    int ans = pq.top();
    return ans;
```

Space → O(k)
TC →

```
pair <bool, int> solve (Node* root) {
    //base case
    if (root == NULL) {
        pair <bool, int> p = make_pair(true, INT_MIN);
        return p;
    }
    if (root -> left == NULL && root -> right == NULL) {
        // leaf node
        pair <bool, int> p = make_pair(true, root->data);
        return p;
    }
    pair <bool, int> leftAns = solve(root -> left);
    pair <bool, int> rightAns = solve(root -> right);

    if ( leftAns . first == true && rightAns . first == true && root -> data > leftAns . second && root -> data > rightAns . second) {
        pair <bool, int> p = make_pair(true, root -> data);
        return p;
    }
    else {
        pair <bool, int> p = make_pair(false, root -> data);
        return p;
    }
```

3 + 4 → VS ✓

▷ Merge "k" sorted array



▷ Smallest range in k list

✓▷ Remove stones to minimize the Total

▷ Longest Happy String

▷ Median in a stream

12- Map/ Trie

- OR we will have Bio
- [illegible]

Hashmap → DS → <key, value>

→ ordered → O(log n)
✓ → un-ordered → O(1)

Time Pass the challenge

include < map>
↓
(It has ordered keys by the (int key))

```
#include <iostream>
#include <unordered_map>
using namespace std;

int main() {
    // Creation
    unordered_map<string, int> m;

    // Insertion
    pair<string, int> P = make_pair("Scorpio", 9);
    m.insert(P);

    m["fortuner"] = 10;

    // access
    cout << m.at("Scorpio");
    cout << m["fortuner"];

    // Search
    cout << m.count("Innova");
    if(m.find("Fortuner")!= m.end()) {
        cout << "found";
    }
    else
        cout << "Not found";

    m.size();

    // Printing all values
    for(auto i:m) {
        cout << i.first << "->" << i.second;
    }
```

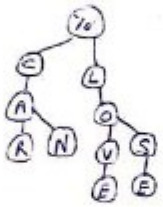
→ (return 0/1, 1/0 or 0/1)

Iterate

repeat fortuner throughout check main and the value Check to match fortuner will give

TRIE

- DS
- multi-way tree structure
- Pattern Searching
- Pattern length & TC



- Node Present → to waha chale Jao
  → Node absent → to create Karo
  ↓
  aur waha chale Jao

String

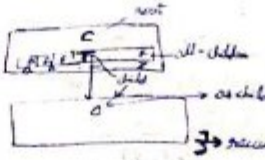
- Find Frequency

```
string str = "thiruvananthapuram";
unordered_map<char, int> freq;
for(int i=0; i< str.length(); i++) {
    char ch = str[i];
    freq[ch]++;
}
for (auto i: freq) {
    cout << i.first << " " << i.second;
}
```

- bool checkCircular(Node* head) {

```
    unordered_map<node*, bool> vis;
    Node* temp = head;
    while(temp != NULL) {
        if(vis.find(temp)!= vis.end()) {
            vis[temp] = true;
        }
        else {
            return true;
        }
        temp = temp->next;
    }
    return false;
```

```
class TrieNode {
    Public:
    char data;
    TrieNode* children[26];   // 3 chij Jaadu th Hume
    bool isTerminal;

    TrieNode(char d) {
        this -> data = d;
        for (int i=0; i<26; i++) {
            children[i] = NULL;
        }
        this -> isTerminal = False;
    }
};
```




```
void insertWord(TrieNode* root, string word) {
    // base case
    if(word.length() == 0) {
        root -> isTerminal = true;   → (Yha Pe Basically hum Sabse Dhee Node ko)
        return;
    }

    char ch = word[0];
    int index = ch - 'A';
    TrieNode* child;
    // Present
    if(root -> children[index] != NULL) {
        child = root -> children[index];
    }
    else {
        // Not Present
        child = new TrieNode(ch);
        root -> children[index] = child;
    }
    // Baki ka recursion sambhal lega
    insertWord(child, word.substr(1));   // Baki children ko bhej do
}
```

```
int main() {
    TrieNode* root = new TrieNode('-');
    insertWord(root, "coding");
}
```

"codes"
"Ind"
"India"

• // Search → (Same Pure approach)

```
bool searchWord(TrieNode* root, string word) {
    // base case
    if(word.length() == 0) {
        return root -> isTerminal;
    }
    char ch = word[0];
    int index = ch - 'A';
    TrieNode* child;

    if(root -> children[index] != NULL) {
        child = root -> children[index];
    }
    else {
        return false;
    }

    searchWord(child, word.substr(1));
```

• **Deletion**

↳ • 2 step Process → eg → Delete → "coding"

ⓐ Sabse Pehle - "coding" Ko search Karunga

ⓑ agar Present to (a) i.e. last character ek Jaunga aur usko Terminal False Karunga

eg → VS

• Find Longest common Prefix (eg 3) → 2 (n)

• Show Suggestion (eg 3) → 1 (n)

Questions - Map

① Array Subset of another array

i/p → a1 = (11, 7, 1, 13, 21, 3, 7, 3)
a2 = (11, 3, 7, 1, 7)

o/p → Yes

```
String isSubset(int a1[], int a2[], int n, int m) {
    unordered_map < int, int > map;
    String ans = "Yes";
  // map hashing using a1
  for(int i=0; i<n; i++) {
    map[a1[i]]++;
  }
  // check whether all elements of a2 present in a1
  for(int i=0; i<m; i++) {
    if(map.find(a2[i]) == map.end()) {
      ans = "No";
      break;
    }
    else {
      // present
      if(map[a2[i]] > 0) {
        map[a2[i]]--;
      }
      else {
        ans = "No";
        break;
      }
    }
  }
  return ans;
}
```

② Union of Two Linked Lists

i/p → L1 = 9→6→4→2→3→8
L2 = 1→2→8→6→2

o/p → 1 2 3 4 6 8 9

→ ordered map me duplicate entry nhi hoti h ☺

int	Node*
9	<9>
6	<6>
4	<4>
2	<2>
3	<3>
8	<8>
1	<1>

→ "L1 ko dale aur L2 dubara dale, to wo overwrite hogya"

```
struct Node* makeUnion (struct Node* head1, struct Node* head2) {
    map<int, Node*> map;
    Node* curr = head1;
  // L1 hashed
  while (curr) {
    map[curr->data] = curr;
    curr = curr->next;
  }
  curr = head2;
  // L2 hashed
  while (curr) {
    map[curr->data] = curr;
    curr = curr->next;
  }
  Node UL* = NULL;
  curr = 0;
  // iterate MAP and make find UL
  for(auto it = map.begin(); it != map.end(); it++) {
    if (!UL) {
      UL = it->second;
      curr = UL;
    }
    else {
      curr->next = it->second;
      curr = curr->next;
    }
  }
  curr->next = 0;
  return UL;
}
```

③ Intersection of Two LL

i/p → L1 → 9→6→4→2→3→8→x
L2 → 1→2→8→6→x

- hash L2 items
- iterate L1 & eliminate items not in L2
- get the final IL

check in hash

L2	no. of occ
1	1
2	⋮
8	⋮
6	⋮

```
Node* findIntersection(Node* head1, Node* head2) {
    unordered_map < int, int > map;
    Node * curr = head2;
  // hash L2 items
  while (curr) {
    map[curr->data]++;
    curr = curr->next;
  }
  Node *IL = NULL;
  Node* it = 0;
  while(curr) {
    if(map.find(curr->data) != map.end()) {
      // node val found in L2
      if (!IL) {
        IL = curr;
        it = IL;
      }
      else {
        if(map[curr->data] > 0) {
          it->next = curr;
          it = it->next;
          map[curr->data]--;
        }
      }
      curr = curr->next;
    }
    it->next = NULL;
    return IL;
  }
```

④ Sum equals to Sum

i/p → {3, 4, 7, 1, 2, 9, 8}

o/p → (3, 7) and (9, 1)

- Since a+b ke value ko map kaise do aur based on c+d ke value ko map se check karenge, wahi to brute force to O(n²) chla jayega.

a+b =	c+d
(3,4)	(7,1)
(3,7)	(7,1)
(3,1)	⋮
⋮	

(1,2)
(1,9)
⋮

Logic

map

3+4 →	7 → True
3+7 →	10 → True

```
int findPairs(int a[], int n) {
    unordered_map<int, bool> map;
    //Pairs
    for(int i=0; i<n; i++) {
        for(int j=i+1; j<n; j++) {
            int sum = a[i]+a[j];
            if(map.find(sum) != map.end()) {
                return true;
            }
            else {
                map[sum] = true;
            }
        }
    }
    return false;
};
```

⑤ Largest Subarray with 0 Sum and Subarray with 0s and 1s

(29)

13 - DP

(OR ye to Recursion + Top-down se hoyegya)

(n to 0)

Top-down

Recursion + Memoization

Memo ko Stack me Rakhe h, Pop ke sth Nikalenge

Bottom up

Tabulation method

Solve by iterative way

0 to n

Space optimization

(is Pehle to Recursion samjhna)

(Pehle Sir Sikhe)

↳ • Jo ek baar calculate kar chuke ho, use dubara calculate karne ki Jarurat nhi h

• "Interview Me all 4 method explain Karna"

① Recursion
② Top-down
③ Bottom-up
④ Space Optimization

• **When to apply?**

→ • Overlapping Sub-Problems → "Some Problem ko baar baar Solve Karna ho"

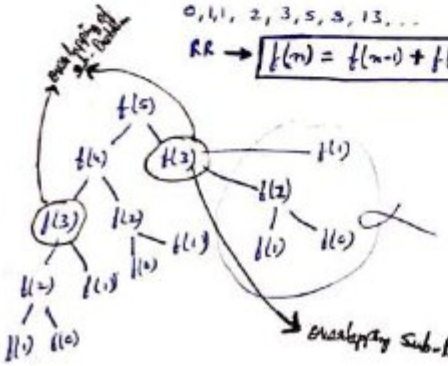
→ • Optimal SubStructure → "Optimal Sol^n of Bigger Problem can be calculated using optimal sol^n of smaller Problem"

★ **Fibonacci :-**

0, 1, 1, 2, 3, 5, 8, 13, . .

RR → $\boxed{f(n) = f(n-1) + f(n-2)}$

$\boxed{\begin{matrix} f(1) = 1 \\ f(0) = 0 \end{matrix}}$ → Base Case



overlapping Sub-Problem (dubara Solve ho rahe ☹)

★★★★★ Top-down → (90% Solve)

• ① Create dp array
↳ vector<int> dp(n+1, -1);
② ofs store dp (Space ↑ ratio)
③ B.C Ke Just baad
↓
check if answer already exist

0 → n
(n+1)

```
int fib(int n) {
    // BC
    if(n==1 || n==0) {
        return n;
    }
    // ↓
    if(dp[n]! = -1)
        return dp[n];
    int ans = fib(n-1) + fib(n-2);
    return ans;
    dp[n]
```

(already ans exist karta h, to Main kyu Karu ☹ nhi h → tab Main answer banaunga)

•

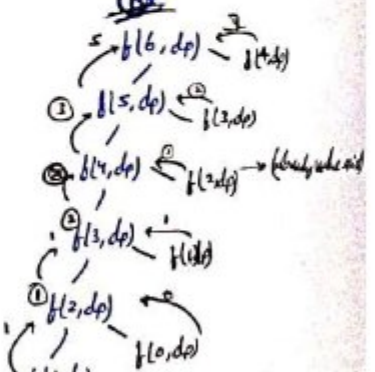
```
# include <iostream>
# include <vector>
using namespace std;
int topdown(int n, vector<int> &dp) {
    // base case
    if(n==0 || n==1) {
        return n;
    }
    if(dp[n]! = -1) {
        return dp[n];
    }
    dp[n] = topdown(n-1, dp) + topdown(n-2, dp);
    return dp[n];
}
```

"Yaad EK Parameter K value change ho rahi h" → $\boxed{\text{1-D DP}}$

```
int main() {
    int n;
    cin >> n;
    vector<int> dp(n+1, -1);
    int ans = topdown(n, dp);
    cout << ans;
    return 0;
}
```

n+1 / Advice

nhi Samajh aaya to (n+1) size K bna do, Jyada Ke bnaoge kya [illegible]



Bottom-up

```
int bottomUpSolve (int n) {
// Step 1 → create dp array
vector<int> dp(n+1, -1);
// Step 2 → observe base case in above sol^n
dp[0] = 0;
if(n==0) return dp[0];
dp[1] = 1;
if(n==1) return dp[1];
// Step 3 : iteration
for(int i=2; i<=n; i++) {
    dp[i] = dp[i-1] + dp[i-2];
}
return dp[n];
```

[redacted] Space optimization

(array ki change kar 2 variable se karlenge)

long long ka use to kar le

0 1 2 3

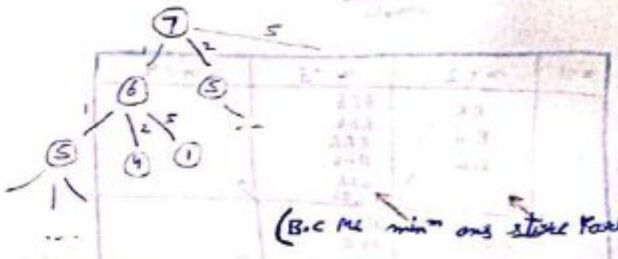
```
int SpaceOptSolve(int n) {
// Step 2: base case
int prev2 = 0;
int prev1 = 1;
if(n==0) return prev2;
if(n==1) return prev1;
int curr;
// Step 3: topdown approach me n kaise travel kara h
for(int i=2; i<=n; i++) {
    curr = prev1 + prev2;
    // shifting
    prev2 = prev1;
    prev1 = curr;
}
return curr;
}
```

Bottom-up

Q1) Coin change i/p → coins = [1,2,5], amount = 11
o/p → 3 (min coin = 5+5+1 → 3 coin)

"If that amount of money cannot be made up by any combination of the coins, return -1."

"You may assume that you have an infinite no. of each kind of coin."



(B.c me min^m ans store karle)

① Recursion

```
int coin(vector<int> &arr, int amount) {
    if(amount==0) {
        return 0;
    }
    if(amount<0) {
        return INT_MAX;
    }
    int mini = INT_MAX;
    for(int i=0; i<arr.size(); i++) {
        int ans = coin(arr, amount - arr[i]);
        if(ans != INT_MAX) {
            mini = min(ans+1, mini);
        }
    }
    return mini;
}

int coinChange(vector<int> &coins, int amount) {
    int ans = coin(coins, amount);
    if(ans == INT_MAX) {
        return -1;
    }
    else {
        return ans;
    }
}

int main() {
    vector<int> arr = {1,2,5};
    int amount = 11;
    int ans = coinChange(arr, amount);
```

② Topdown

```
int solveMem(vector<int> &coins, int amount, vector<int> &dp) {
    // base case
    if(amount==0) {
        return 0;
    }
    if(amount<0) {
        return INT_MAX;
    }
    // ans already exist
    if(dp[amount] != -1) {
        return dp[amount];
    }
    int mini = INT_MAX;
    for(int i=0; i<coins.size(); i++) {
        int ans = solveMem(coins, amount - coins[i], dp);
        if(ans != INT_MAX) {
            mini = min(mini, ans+1);
        }
    }
    dp[amount] = mini;
    return mini; // or return dp[amount]
}

int coinChange(vector<int> &coins, int amount) {
    vector<int> dp(amount+1, -1);
    int ans = solveMem(coins, amount, dp);
    if(ans == INT_MAX)
        return -1;
    else
```

(ok likhta h)

TMCK (to value change ho rahi h)

• NO space optimization

• Bottom-up → VS

32 House Robber

I/P → nums = [1, 2, 3, 1]

O/P → 4

Longest-Longest Chori Mat Karna

• Use the Space optimize technique

• **Recursion**

n → represent the index of current house

```
int Solve(vector<int> & nums, int n) {
    if (n<0)
      return 0;
    if (n==0) {
      return nums[0];
    }
    int include = Solve(nums, n-2) + nums[n];
    int exclude = Solve(nums, n-1) + 0;

    return max(include, exclude);
```

• **Top-Down**

```
int Solve(vector<int> & nums, int n, vector<int> & dp) {
    if (n<0)
      return 0;
    if (n==0)
      return nums[0];
    if (dp[n] != -1)
      return dp[n]
    int include = Solve(nums, n-2, dp) + nums[n];
    int exclude = Solve(nums, n-1, dp) + 0;
    dp[n] = max(include, exclude);
    return dp[n];
```

6 Painting Fence

→ The Problem states that there are 'n' fences which can be coloured with one of the 'K' colours in such a way that not more than two adjacent fences have the same colour.

I/P → n fences, K-colors

O/P → no. of ways

n = 3, K = 2

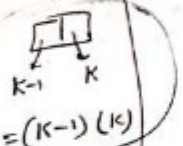
Using Recursion

O/P → 6

	n=1	n=2	n=3	n=4
Same	0	RR BB GG	BBB RGG BRR BGG GRR GBB	
diff	R B G	RB RG BR BG GR GB	RBR RGR RBG RGB BRG BGR GBR GBG	

K

'Permutation of colors'



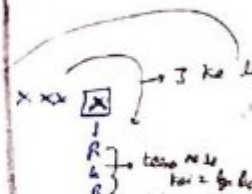
$= (K-1)(K)$

$K + (K*(K-1))$

f(4) = XXXX

Same → f(2) → XX RR, BB, GG → We sequence Similar logic

diff → f(3) → (K-1)



$\therefore f(4) = f(2) * (K-1) + f(3) * (K-1)$

$$f(n) = [f(n-2) + f(n-3)] * (K-1)$$

- Painting Fence
- count derangement

Formula, but before determine know the logic

• **Recursion**

```
int SolveUsingRecursion(int n, int K) {
    if(n==1) {
        return K;
    }
    if(n==2) {
        return (K+K*(K-1));
    }
    int ans = (SolveUsingRecursion(n-2, K) + SolveUsingRecursion(n-1, K))*(K-1);
    return ans;
}
```

• **Top-Down**

```
int SolveUsingMemo(int n, int K, vector<int> $ dp) {
    if(n==1) {
        return K;
    }
    if(n==2) {
        return (K+K*(K-1));
    }
    if(dp[n] != -1) {
        return dp[n];
    }
    dp[n] = (SolveUsingMemo(n-2, K, dp) + SolveUsingMemo(n-1, K, dp))*(K-1);
    return dp[n];
}
```

• **Bottom-up**

```
int SolveUsingTab(int n, int K)
    vector<int> dp(n+1, 0);
    dp[1] = K;
    dp[2] = (K+K*(K-1));
    for(int i=3; i<=n; i++) {
        dp[i] = (dp[i-2] + dp[i-2]*(K-1);
    }
    return dp[n];
}
```

• **Space optimization** → (i) I h we (i-2) and (i-1) be depend kar sake h

```
int SolveSpace(int n, int K) {
    int Prev2 = K;
    int Prev1 = (K + K*(K-1));
    for(int i=3; i<=n; i++) {
        int curr = (Prev2 + Prev1)*(K-1);
        Prev2 = Prev1;
        Prev1 = curr;
    }
    return Prev1;
}
```

• Top - Down

HW (Climbing Stairs)

(19) → count derangement

$i/p \rightarrow \{0,1,2\}$ → $\boxed{i \ i \ 0}$, $\boxed{2,0,1}$ ∴ $\boxed{o/p \rightarrow 2}$

```
int countDer(int n) {
    // base case
    if(n==1) return 0
    if(n==2) return 1
    count = (n-1)*[countDer(n-1) + countDer(n-2)]
    return count;
```

0/1 Knapsack Problem

↳ Given a set of n items numbered from 1 up to n, each with a weight w_i and a value v_i, along with a maximum weight capacity W, maximize the sum of the values of the items in the Knapsack so that the sum of the weights is less than or equal to the Knapsack's capacity.

Same Pattern
- → Subset Sum
- → Equal Subset sum Partition
- → Min Subset sum difference

n - items

weight → $\{W_1, W_2, W_3, W_4\}$

value → $\{V_1, V_2, V_3, V_4\}$

I II III IV

Subset that max value sum → [ ] → max capacity = W

{I} → 1
{II} → X
{III} → 3
{I, II} → x
{I, III} → x
{II, III} → x
{I, II, III} → x

max = 3

n = 3
weight → {4, 5, 1}
value → {1, 2, 3}
capacity = 4

Subsequence wala concept of by gya
↓
Incl/Excl.

n = 2
weight = {4, 5, 1}
value = {1, 2, 3}
capacity = 4

Knapsack → value

{3, 0}

inc. → {4, 1}
ex → {3, 0}

Ans

Initially answer is 0 hogaya

// here n = index of last element

• Using Recursion

int SolveUsingRecursion(int weight[], int value[], int index, int capacity){
// base case only 1 item
if (index == 0) {
if (weight[0] <= capacity) {
return value[0];
}
else {
return 0;
}
// include and exclude
int include = 0;
if (weight[index] <= capacity) {
include = value[index] + SolveUsingRecursion(weight, value, index-1, capacity - weight[index]);
}
int exclude = 0 + SolveUsingRecursion(weight, value, index-1, capacity);
int ans = max(include, exclude);
return ans;
}

We do same thing as shift it to the 2D-DP h.w

We -ve no. to avoid wrong if condition not handle this bugs

• Top - Down

int SolveUsingMem(int weight[], int value[], int index, int capacity, vector<vector<int>> & dp){
if (index == 0) {
if (weight[0] <= capacity) {
return value[0];
}
else {
return 0;
}
if (dp[index][capacity] != -1) {
return dp[index][capacity];
// include & exclude
int include = 0;
if (weight[index] <= capacity)
include = value[index] + SolveUsingMem(weight, val, index-1, capacity - weight[index], dp);
int exclude = 0 + SolveUsingMem(weight, value, index-1, capacity, dp);
dp[index][capacity] = max(include, exclude);
return dp[index][capacity];

Create 2D-DP

n, n+1 → memo kaise kare

main fnt{
vector<vector<int>> dp(-+1, vector<int>(capacity+1, -1));
int ans = SolveUsingRecursion(weight, value, n-1, capacity);

Partition Equal Subset Sum

↳ Given an integer array nums, return true if you can partition the array into two subsets such that the sum of the elements in both subsets is equal or false otherwise.

So, 2 equal subset h

$\frac{N}{2}$ subset

i/p → nums = [1,5,11,5]
o/p → true

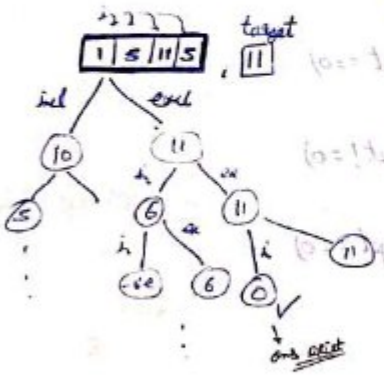
Explanation → The array can be partitioned as [1,5,5] and [11]

[] → {1, 5, 11, 5}
→ total = 22
∴ $\frac{total}{2}$ = 11 → bag 11 banana h ab

target

Sara kaha we did → Subset Sum

→ ab agar all pairs 11 ke equal h to dekhe wala to automatic equal hi hoga na bakaf (:)



- **Using Top-Down**

```
bool usingMemo(int index, vector<int>& nums, int target,
                vector<vector<int>>& dp) {
    int n = nums.size();
    if(index >= n) return 0;
    if(target < 0) return 0;
    if(target == 0) return 1;
    if(dp[index][target] != -1) return dp[index][target];
    bool include = usingMemo(index+1, nums, target-nums[index], dp);
    bool exclude = usingMemo(index+1, nums, target, dp);
    dp[index][target] = (include || exclude);
    return dp[index][target];
}

vector<vector<int>> dp(nums.size(), vector<int>(target+1, -1));
```

Kya 2 chiz change ho rahi?
① Index
② Target

- **Using Recursion**

```
bool solveUsingRecursion(int index, vector<int>& nums,
                         int target) {
    // base case
    int n = nums.size();
    if (index >= n) return 0;
    if(target < 0) return 0;
    if(target == 0) return 1;

    bool include = solveUsingRecursion(index+1, nums,
                   target - nums[index]);
    bool exclude = solveUsingRecursion(index+1, nums, target);
    return (include || exclude);
```

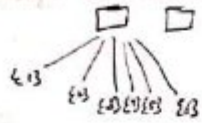
```
bool canPartition(vector<int>& nums) {
    int sum = 0;
    for (int i = 0; i < nums.size(); i++) {
        sum += nums[i];
    }
    int target = sum/2;
    int index = 0;
    bool ans = solveUsingRecursion(index, nums, target);
    return ans;
}
```

Sab galat karunga

agar all h sum-target
Else hi aayega

```
if(sum % 2)
    return false;
```

★ No. of Dice Roll with Target Sum



n = 2 → Dice
K = 6
Target = 7

"Since answer kafi bara h to Q. me li likha h mod kya lo (to vs me mod ke sath proper calculate h)"

• **Top-Down**

```
int SolveUsingMemo(int n, int K, int target, vector<vector<int>>&dp){
    if(n<0)  return 0;
    if(n==0 && target==0)  return 1;
    if(n==0 && target!=0)  return 0;
    if(n!=0 && target==0)  return 0;

    if(dp[n][target]!=1)
        return dp[n][target];

    int ans = 0;
    for(int i=1; i<=K; i++){
        ans += SolveUsingMemo(n-1, K, target-i, dp);
    }
    dp[n][target] = ans;
    return dp[n][target];
}
```

if(target - i >= 0)

• **Recursion**

```
int SolveUsingRecursion(int n, int K, int target){
    if(n<0)
        return 0;
    if(n==0 && target==0)
        return 1;
    if(n==0 && target!=0)
        return 0;
    if(n!=0 && target==0)
        return 0;

    int ans = 0;
    for(int i=1; i<=K; i++){
        ans += SolveUsingRecursion(n-1, K, target-i);
    }
    return ans;
}
```

• Minimum Swaps to make sequence increasing → strictly increasing

i/p → nums1 → [1, 3, 5, 4] → 1, 3, 5, 7
nums2 → [1, 2, 3, 7] → 1, 2, 3, 4

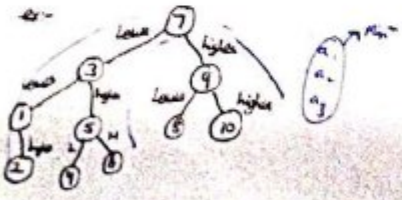
★ Guess Number Higher or Lower II

(Q-Smallest Imp-h Take Ba)

→ Logic ⇒ [Max^m Penality is the min^m no. required]

(Worst case) (What Some Cases have been solved)

Q We are playing the Guessing game. The game will work as follows:
I pick a number between 1 and n.
You guess a number.
If you guess the right number, you win the game.
If you guess the wrong number, then I will tell you whether the number I picked is higher or lower, and you will continue guessing.
Every time you guess a wrong number x, you will pay x dollars. If you run out of money, you loose the game.
Given a particular n, returns the minimum amount of money you need to guarantee a win regardless of what number I picked.



[1, 10]
6 → correct
wrong
[1, 5] [7, 10]
₹6 Penality

[1, 5] take max of this
3 → wrong
[1, 2] [4, 5]
2 → wrong 5 → wrong
1 ✓ 4 ✓
Ans = 3+2 = ₹5 Ans = 3+5

(₹8 Se Jitna ki (Same high) — Safe h)

- **Recursion**

```
int SolveUsingRecursion (int start, int end) {
    if (start >= end)
        return 0;
    int ans = INT_MAX;
    for (int i = start; i <= end; i++) {
        ans = min(ans, i + max(SolveUsingRecursion(start, i-1), SolveUsingRecursion(i+1, end)));
    }
    return ans;
}

int getMoneyAmount (int n) {
    int ans = SolveUsingRecursion(1, n);
    return ans;
}
```

(L) (R)

LAKSHAY BHARDWAJ APPROACH

Main has position le swap, ne swap karke dekh lunga aur min nikal lunga

- Swap table Hainge Jab Kaa Lagenge

$P_1 < nums2[i]$ & $P_2 < nums1[i]$

Main Sahi Me swap nhi Kar rahe, bas dekh le rahe h count++

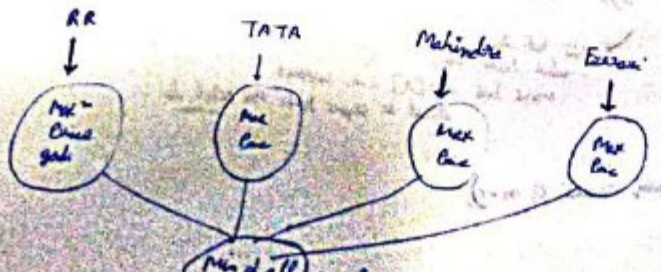
⇒ Check Leetcode → 2 JULY or 15 JUNE

- **Top-Down** → (2D DP)

```
int SolveUsingMemo (int start, int end, vector<vector<int>> & dp) {
    if (start >= end)
        return 0;
    if (dp[start][end] != -1) {
        return dp[start][end];
    }
    int ans = INT_MAX;
    for (int i = start; i <= end; i++) {
        ans = min(ans, i + max(SolveUsingMemo(start, i-1, dp), SolveUsingMemo(i+1, end, dp)));
    }
    return dp[start][end] = ans;
}
int getMoneyAmount (int n) {
    vector<vector<int>> dp(n+1, vector<int>(n+1, -1));
    int ans = SolveUsingMemo(1, n, dp);
    return ans;
}
```

Logic :- (Real life)

1 Top Model → Min Price



Minimum Cost Tree from Leaf Values

(Tree banane ki disha me Min Cost Jaa)

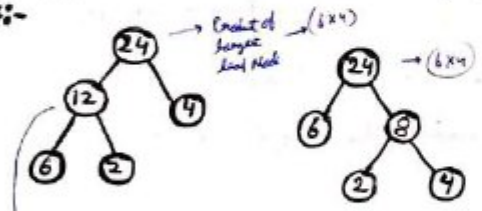
Given an array arr of positive integers, consider all binary trees such that:

- Each node has either 0 or 2 children
- The value of arr correspond to the value of each leaf in an in-order traversal of the tree.
- The value of each non-leaf node is equal to the product of the largest leaf value in its left and right subtree, respectively.

Among all possible binary trees considered, return the smallest possible sum of the values of each non leaf node. It is guaranteed this sum fits into a 32-bit Integer.

A node is a leaf if and only if it has zero children.

ex:-



6×2

24+12 = 36

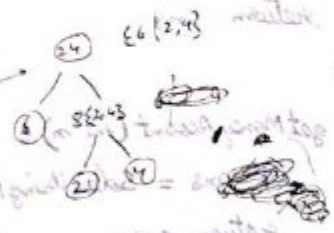
24+8 = 32

Sum of Non-leaf → Min = 32 Ans,

i/p → [6, 2, 4]

o/p → 32

→ Min Possible Sum of Non-leaf nodes in tree

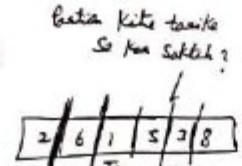


(Array-type)

i/p → | 2 | 6 | 1 | 5 | 3 | 8 |

left | right

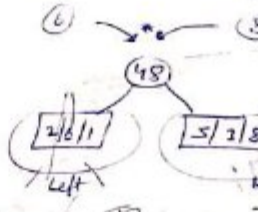
Kaise kaatu ki Sabkaa?



Without So Tree has effect

Min

Ans



recursion

max[left tree] * max[right tree]

+ Solve (left tree)

+ Solve (right tree)

Might take Min answer change (of leaf sum)

Map Se

1 - 3 → 6

4 - 6 → 8

index (key)

Highest value (value)

(Store the large value)

- **Recursion**

int minPreorderValues(vector<int> &arr) {

map<pair<int, int>, int> maxi;

// pre-computation

for (int i=0; i<arr.size(); i++) {

maxi[{i,i}] = arr[i];

for (int j = i+1; j<arr.size(); j++) {

max[{i,j}] = max(arr[j], maxi[{i, j-1}]);

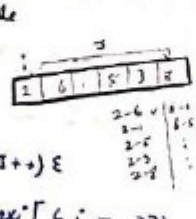
}

}

int n = arr.size()

int ans = SolveUsingRecursion(arr, maxi, 0, n-1);

return ans;



DS has type O(1), to order 2-Range to order I Max value h, Use strong header h

pair<int, int> p;

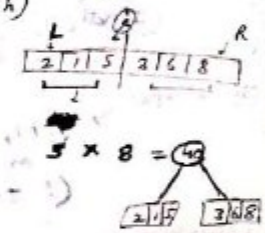
p.first = i;

p.second = i;

maxi[p] = arr[i];

```
int SolveUsingRecursion ( vector < int > & arr, map < Pair < int, int >, int & maxi, int left, int right) {
    //base case
    if (left == right)
        return 0;

    int ans = INT_MAX;
    for (int i = left ; i < right ; i++) {
        ans = min ( ans,
                maxi [{ left, i }] * maxi [{ i+1 , right }]
                + SolveUsingRecursion (arr, maxi, left, i)
                + SolveUsingRecursion (arr, maxi, i+1, right));
    }
    return ans;
}
```

(non-leaf ki value add karni h)



- **Top-Down**

```
int SolveUsingMem( vector < int > & arr, map < Pair < int, int >, int > & maxi, int left, int right,
    vector < vector < int >> & dp) {
    //base case
    if ( left == right)
        return 0;
    if ( dp[left][right] != -1) {
        return dp[left][right];
    }
    int ans = INT_MAX;
    for (int i = left; i < right; i++) {
        ans = min( ans,
                maxi [{ left, i }] * maxi [{ i+1, right }]
                + SolveUsingMem (arr, maxi, left, i, dp)
                + SolveUsingMem( arr, maxi, i+1, right, dp));
    }
    dp[left][right] = ans;
    return dp[left][right];
}
vector <vector < int > > dp(n+1, vector < int > (n+1, -1));
int ans = SolveUsingMem (arr, maxi, 0, n-1, dp);
return ans;
```

★ DP on Strings

(Order Maintain hona chahiye)

LCS → (YYYY) (Same to Same ek ghuma ke 'Yahi' Pick dunge)

→ LONGEST COMMON SUBSEQUENCE

Top Q
- LCS
- Coin change
- Longest Increase Subsequence

i/p → abcde
ace

o/p → 3

→ (using 2 Pointer Approach)

1 a b c d e
2 a c e

If Match [i] == [j] → recc(i+1, j+1)

If not equal → (i+1, j) or (i, j+1)

★ **Longest Palindromic Subsequence**

Str → "bbhab"
Rev. Str → "bahbb"
→ LCS → 4

abc → a, b, c, ab, bc, ac, abc

acd → a, c, d, ac, cd, ad, acb

2 A

- **Recursion**

```
int SolveUsingRecursion (string a, string b, int i, int j) {
    if (i == a.length()) {
        return 0;
    }
    if (j == b.length()) {
        return 0;
    }
    if (a[i] == b[j])
        return 1 + SolveUsingRecursion (a, b, i+1, j+1);
    else {
        return 0 + max ( SolveUsingRecursion (a, b, i+1, j) , SolveUsingRecursion (a, b, i, j+1))
    }
```

OHH !!!

'Agar DP use Karne Ke baad bhi agar TLE aaye to Sabko '&' by reference kar Kardo'

- **Top-Down**

```
int SolveUsingMem ( string &a, string &b, int i, int j, vector<vector<int>> &dp) {
    if (i == a.length()) {
        return 0;
    }
    if (j == b.length()) {
        return 0;
    }
    if (dp[i][j] != -1)
        return dp[i][j];
    int ans = 0;
    if (a[i] == b[j])
        ans = 1 + SolveUsingMem (a, b, i+1, j+1, dp);
    else {
        ans = 0 + max(SolveUsingMem (a, b, i+1, j, dp), SolveUsingMem (a, b, i, j+1, dp));
    }
    dp[i][j] = ans;
    return dp[i][j];
}
```

★ Longest Increasing Subsequence

→ binary Search se v hoyegya
→ hm DP se karenge

arr → [5, 8, 3, 2, 1, 9, 7]

inc/exc

[5, 8, 9]

i/p → nums = [10, 9, 2, 5, 3, 7, 101, 18]

o/p → 4

[2, 3, 7, 101]

include → at first (initial)
→ if curr > prev

exclude → if curr < prev

logic to find out the concept → "at start agar like wale start se hoke h to nai include karlenge"

[Bas yahi change aata h]

• **Recursion**

```
int SolveUsingRecursion (vector<int>& arr, int curr, int prev) {
    if (curr >= arr.size()) {
        return 0;
    }

// include
int include = 0;
if (prev == -1 || arr[curr] > arr[prev])
    include = 1 + SolveUsingRecursion (arr, curr+1, curr);

// exclude
    int exclude = 0 + SolveUsingRecursion (arr, curr+1, prev);
int ans = max( include, exclude);
    return ans;
}

int lengthOfLIS(vector<int>& nums)
    int prev = -1;
    int curr = 0;
    int ans = SolveUsingRecursion (nums, curr, prev);
    return ans;
}
```

Ye update wali logic, yha galti ke chances h

• **Top-Down**

(prev+1 index kiske liyaki prev = -1 th index)

```
int SolveUsingMem ( vector<int>& arr, int curr,
int prev, vector<vector<int>>& dp) {
    if (curr >= arr.size()) {
        return 0;
    }
    if (dp[curr][prev+1] != -1)
        return dp[curr][prev+1];

// include
int include = 0
if (prev == -1 || arr[curr] > arr[prev])
    include = 1 + SolveUsingMem ( arr, curr+1, curr, dp);
// exclude
int exclude = 0 + SolveUsingMem (arr, curr+1, prev, dp);
dp[curr][prev+1] = max(include, exclude);
return dp[curr][prev+1];
}

int lengthOfLis( vector<int>& nums) {
    int curr = 0;
    int prev = -1;
    int n = nums.size();
    vector<vector<int>> dp (n, vector<int>(n+1, -1));
    int ans = SolveUsingMem (nums, curr, prev, dp);
    return ans;
```

• .

★ DP with Binary Search (optimal Sol^n)

→ 'Agar bas dekh lo, agar tum has Top-down Method se hi kaaraga ?' ←

arr → [0, 1, 0, 3, 2, 3]

[0, 1, 2, 3]
[0, 2, 3]
[0, 1, 3]
[2, 3]
→ ans = 4

approach

[10, 9, 2, 5, 3, 7, 101, 18]

arr → [10, 9, 2, 5, 3, 7, 101, 18]

length = 4

Logic

Jab 1 to upset hoda
aur dusre wale se bura 1 to
wale index ko dekh do

```
int solveoptimal (vector<int> &arr) {
    if(arr.size()==0)
        return 0;
    vector<int> ans;
    ans.push_back(arr[0]);
    for (int i = 1; i < arr.size() ; i++) {
        if (arr[i] > ans.back()) {
            // include
            ans.push_back(arr[i]);
        }
        else {
            int index = lower_bound (ans.begin(), ans.end(), arr[i]) - ans.begin();
            ans[index] = arr[i]
        }
    }
    return ans.size();
}
```

→ O(n · log n)

★ HW ⇒ Russian Doll Envelopes → LIS with length

★ LIS + Sort — using 1st

(Sa dekh ne order me Pura wale weight)

height aur width dono hai Challenge

i/p → {[5,4], [6,4], [6,7], [2,3]}

(width ke according increasing order me sort karde)

→ height ke according sort karenge !!

∴ [2,3], [5,4], [6,7], [6,4] → LIS → 3

o/p → 3

i/p → [1,1], [1,1], [1,1]
o/p → 1

Both sides by may check kiskaise
Latest Solution

(279)

★ Questions

- Perfect Square
- Min cost for Tickets
- Best Time to buy & Sell stock
- House Robber-III
- Reduce the Minimum

- Word break-1
- Target Sum
- Ones and Zeroes
- Min Swaps to make Sequence increasing
- Longest Palindromic Substring
- Min ASCII delete Sum for two strings

- Reducing Dishes
- Unique BST-2
- Word break-2
- Distinct Subsequence
- Interleaving string
- Min Insertion Steps to make

- Min no. of Numbers make monotonic array
- Russian Doll Envelopes
- Make array strictly increasing
- Burst Balloons
- Stone Game

height ke Dekh ke backward (with Sort)

after Sort

if (prev == -1) || envelopes[i][0] > envelopes[prev][0] && envelopes[i][1] > envelopes[prev][1]

85/87 → only test case Pass

try to go with DP with binary Search

if (a[0] == b[0]) return a[1] > b[1] // 'a' table wale aur 'b' wale match then
return a[0] < b[0] // 'a' table wale aur 'b' table wale match then

SORT

→ 'agar abhi wala kisi criteria se Pura wale ke depend kar sakta h'

[use concept of LIS]

★ LIS → sort

• Max height by Stacking Cuboid

Sort + LIS

i.e. ith steps wise match

i/p → [w, l, h]
[50, 45, 20]
[95, 37, 53]
[45, 23, 12]

For max height

Sort → [l, w, h]
[20, 45, 50]
[37, 53, 95]
[12, 23, 45]

Sort acc. to width,

[12, 23, 45]
[20, 45, 50]
[37, 53, 95]

⇒ ab wahi upar LIS lagao or DONE

LIS

[Code me tabulation wale use karna h. Main Memo (Top-down) me Unique Kaise Tak Kaaraga]

Buy & Sell Stock - 2

Buy & Sell Stock - 2

i/p → 7, 1, 5, 3, 6, 4

o/p → 1 → buy, 5 → sell, 3 → buy, 6 → sell → 4+3 = 7 ✓

- Buy | Sell | Ignore
- Sell only if Buy
- Profit is sell - buy

```
int Solve(vector<int> & Prices, int i, bool buy, vector<vector<int>> & dp) {
    if (i >= Prices.size()) {
        return 0;
    }
    if (dp[i][buy] != -1) {
        return dp[i][buy];
    }

    int Profit = 0;
    if(buy) {
        int BuyProfit = -Prices[i] + Solve(Prices, i+1, 0, dp);
        int ignore = Solve(Prices, i+1, 1, dp)
        Profit = max(Buyprofit, ignore);
    }
    else {
        int SellProfit = Prices[i] + Solve(Prices, i+1, 1, dp);
        int ignore = Solve(Prices, i+1, 0, dp);
        Profit = max(Sellprofit, ignore);
    }
    dp[i][buy] = Profit;
    return dp[i][buy];
}
int maxProfit(vector<int> & Prices) {
    vector<vector<int>> dp(Prices.size() + 1, vector<int>(2,-1));
    return Solve(Prices, 0, 1, dp);
}
```

Buy & Sell Stock - 3

(at most 2 transaction allowed)

3D-DP

ek aur chij add karenge limit = 2 → aur kam hoti jaayegi limit -- karenge

ek base case add karenge

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

→ Solve(Prices, i+1, 0, dp, limit--)

```
vector<vector<vector<int>>> dp(Prices.size(),
    vector<vector<int>>(2, vector<int>(3,-1)));
```

Buy & Sell Stock - 4

(at Most K transaction)

bas modify one thing

limit = K karlo

"bagon ki baat"

▸ agar Test case Pass hogya aur submit ki error aa rha h, so check:-

(↳ & in fact →

↳ dp declaration me limit+1 karlo ... ok)

- BS - 1 → only 1 transaction
- BS - 2 → ∞ no. transaction
- BS - 3 → at most 2 transaction
- BS - 4 → at most K transaction
- BS - 5 → (BS-2) - fee

Buy and Sell Stock - 5

(with transaction fee)

i/p → Price = [1, 3, 7, 8, 4, 9], fee = 2

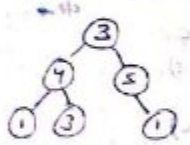
∴ [(8-1) - 2] + [(9-4) - 2]

= 8 ✓

[bas jab Sell kar rahe h, uska (Price - fee) kharcha hoke sab sand h]

House Robber - III

(inclusion / exclusion)



∴ Total Amount = 4+5

= 9 ✓

```
int Solve(TreeNode * root) {
    if (!root) {
        return 0;
    }
    int rob = 0;
    int notrob = 0;
    // include
    rob = rob + root -> val;
    if (root -> left) {
        rob = rob + Solve(root -> left -> left) + Solve(root -> left -> right);
    }
    if (root -> right) {
        rob = rob + Solve(root -> right -> left) + Solve(root -> right -> right);
    }
    // exclude
    notrob = notrob + Solve(root -> right) + Solve(root -> left);
    return max(rob, notrob);
```

```
if (dp.find(root) != dp.end()) {
    return dp[root];
}
```

For DP → unordered_map < TreeNode *, int > dp

Predict the Winner

→ TRUE if Player 1 wins or draw
else FALSE

i/p → [1, 5, 2]

Player 1 → choose 1 → Player 2 → 5 (max) → Player 1 choose 2 (5 > 2)
Player 1 → choose 2 → Player 2 → 5 (max) → Player 1 choose 1 (5 > 3)

- Player 1 way to win,
  - Focus on diff of score → $\boxed{S_{P_1} - S_{P_2} \geq 0}$ → true

```
int Solve(vector<int> &nums, int start, int end) {
    if(start == end) return nums[start];

    // diff P1 - P2
    int diffbyStart = nums[start] - solve(nums, start+1, end);
    int diffbyEnd = nums[end] - solve(nums, start, end-1);

    return max(diffbyStart, diffbyEnd);
}
```

→ 5:2 4 Min

Target Sum

→ either ⊕ or ⊖

Inclusion/exclusion

i/p → nums = [1,1,1,1,1], target = 3
o/p → 5

```
int Solve(vector<int> &nums, int target, int i) {
    if(i == nums.size()) {
        return target == 0 ? 1 : 0
    }

    int Plus = Solve(nums, target - nums[i], i+1) // (+)
    int minus = Solve(nums, target + nums[i], i+1) // (-)

    return Plus + minus;
}
```

Word Break

i/p → s = "leetcode", dict = [leet, code]
o/p → True

s = "catsandog", dict = [cats, dog, sand, and, cat]
- cat, sand, dog
- cats, and, dog → op → false
- cat, dog, dog

```
bool check(vector<String> &wordDict, string &s) {
    for(auto i: wordDict) {
        if(s == i)
            return true;
    }
    return false;
}
bool Solve(string &s, vector<string> &wordDict, int start) {
    if(start == s.size())
        return true;

    // one case
    string word = "";
    bool flag = false;
    for(int i = start; i < s.size(); i++) {
        word += s[i];
        if(check(wordDict, word)) {
            flag = flag || Solve(s, wordDict, i+1);
        }
    }
    return flag;
}
```

(call for both cases)
cat sand dog
cat dog dog

Longest Palindromic Substring

↔ Diff from longest Palindromic Subsequence

(• 2 for-loop → find all substring
• check Palindrome → by recursion)

```
int maxLen = 1, start = 0;
bool Solve(string &s, int i, int j) {
    if(i >= j) {
        return true;
    }
    bool flag = false;
    if (s[i] == s[j]) {
        flag = Solve(s, i+1, j-1);
    }
    if(flag) {
        int curLen = j - i + 1;
        if(curLen > maxLen) {
            maxLen = curLen;
            start = i;
        }
    }
    return flag;
}

string longestPalindrome(string s) {
    int n = s.size();
    for(int i = 0; i < n; i++) {
        for(int j = i; j < n; j++) {
            bool t = Solve(s, i, j);
        }
    }
    return s.substr(start, maxLen);
}
};
```

Ones and Zeros

i/p → strs = ["10", "0001", "111001", "1", "0"], m = 5, n = 3
o/p → 4

(atmost) → subset

→ 10, 1, 0 → (5-2, 3-1) (accepted)
→ 0001, 1 → (5-4, 3-1) (accepted)

Solve/Subset with some condition

No. of 0's & 1's is to count the length of the array

Pick vector from subset, take max length count the length of subset as the ans

[10, 0001]
vector[<1,1>, <3,1>]

will have to find backpack

```
void convertStrToNumStr(vector<string> &strs, vector<pair<int, int>> &numStrs) {
    for(auto str: strs) {
        int zeros = 0, ones = 0;
        for(auto ch: str)
            if(ch == '0')
                zero++;
            else
                ones++;
        numStrs.push_back(Ezeros, ones});
    }
}
int Solve(vector<pair<int, int>> &numStrs, int i, int m, int n) {
    if(i == numStrs.size())
        return 0;
    int zeros = numStrs[i].first;
    int ones = numStrs[i].second;
    int include = 0, exclude = 0;
    if(m - zeros >= 0 && n - ones >= 0)
        include = 1 + Solve(numStrs, i+1, m-zeros, n-ones);
    exclude = Solve(numStrs, i+1, m, n);
    return max(include, exclude);
}
```

→ Have my hidden logic
Back in vectors
→ JUNE

Min ASCII Delete Sum for two Strings

Sol/to
s / t
cat/ca sat/ca
a / t s / t
ca/ca ca/ca sat/sa sat/sa
a / t a / t
ca/ca ca/ca
+

→ Min-cost

add ASCII

```
int cost = 0;
// Base case
if (i == s.size()) {
    for(int j = j; j < s2.size(); j++)
        cost = cost + s2[j];
    return cost;
}
if(j == s2.size())
    for(int j = i; j < s1.size(); j++)
        cost = cost + s1[j];
    return cost;

// cost
if(s1[i] == s2[j])  cost = Solve(...)
else if cost = min(Solve(...), Solve(...))
    cost = cost + s1[i] + Solve(...) + s2[j]
```

→ Min Swap to make Sequence monotonic → by self

→ not equal by both ways

Reducing Dishes

- It can discard as well
- time × satisfaction
- can not change order

i/p → [-1, -8, 0, 5, -9]

(-1×1) + (0×2) + (5×3)

o/p → 14

```
int Solve (vector < int > & sat, int i, int time) {
    if (i == sat.size()) {
        return 0;
    }
    int include = time × sat[i] + Solve (sat, i+1, time+1);
    int exclude = Solve (sat, i+1, time);
    return max(include, exclude);
}

int maxSatisfaction (vector < int > & satisfaction) {
    sort (satisfaction.begin(), satisfaction.end());
    return Solve (satisfaction, 0, 1);
}
```

Unique BST-2

BST → right node, left node

N = 3

i/p → 1, 2, 3, 4, 5, 6, 7

```
vector < TreeNode* > allSolution(int s, int e) {
    if (s > e) return {0};
    if (s == e) return { new TreeNode(s) };
    vector < TreeNode* > ans;
    for (int i = s; i <= e; i++) {
        vector < TreeNode* > left = allSolution(s, i-1);
        vector < TreeNode* > right = allSolution(i+1, e);
        for (int j = 0, j < left.size(); j++) {
            for (int k = 0, k < right.size(); k++) {
                TreeNode* root = new TreeNode(i);
                root → left = left[j];
                root → right = right[k];
                ans.push_back(root);
            }
        }
    }
    return ans;
}
```

```
if (n == 0)
    return {};
return allSolution(1, n);
```

Distinct Subsequence

M1 → all subsequence & then compare with 't' → if equal → count++ } → TLE

M2

s = abbbit

t = abbit

abbbit

abbit

```
if (s[i] == t[j]) {
    Solve (i+1, j+1)
}
Solve (i+1, j)
```

try both include & exclude

```
if (j == t.size()) {
    return 1;
}
if (i == s.size()) {
    return 0;
}
int ans = 0;
if (s[i] == t[j]) {
    ans = ans + Solve (s, t, i+1, j+1);
}
ans = ans + Solve (s, t, i+1, j);
return ans;
```

Word Break-2

i/p → s = "catsanddog"

wordDict = ["cat", "cats", "and", "sand", "dog"]

o/p → ["cats and dog", "cat sand dog"]

Approach

1. Make a valid word
2. word + " " + (all right valid sentence) by recursion

cat + " " + (sanddog) → recursion

cats + " " + (and)

```
vector < string > Solve (string & s, unordered_map < string, bool > & dict, int i) {
    if (i == s.size())
        return {" "};
    vector < string > ans;
    string word;
    for (int j = i; j < s.size(); j++) {
        word.push_back(s[j]);   // (word = word + s[j];)
        if (dict[word].find(word) == dict.end())
            continue;
    }
    // found a valid word
    auto right = solve (s, dict, j+1);
    for (auto eachRightPart : right) {
        string subpart;
        if (eachRightPart.size() > 0) {
            subpart = " " + eachRightPart;
        }
        ans.push_back(word + subpart);
    }
    return ans;
}

vector < string > wordBreak (string s, vector < string > & wordDict) {
    unordered_map < string, bool > dict;
    for (auto word : wordDict)
        dict[word] = 1;
    return Solve (s, dict, 0);
}
```

"word" to a new valid string, to paste side

Pine apple Pen apple

→ Pine + " " + ...

→ Pineapple + " " + ...

(see diagram in the div)

Interleaving Strings

$s_1 + s_2 + s_3 + \dots s_n$

$t_1 + t_2 + \dots t_n$

$s_3 \rightarrow s_1 + t_1 + s_2 + t_2 + t_3 - s_n + t_n$

or

$t_1 + s_1 + t_2 + s_2 + \dots t_n$

M1

s_1 → find all subsets

s_2 → find all subsets

$s_n + t_n + s_{n-1} + t_{n-1} \dots = s_3$?

M2

$[m \le 1] \ni x$

s_1: aabcc

s_2: dbbca

aodbbc bcac

match with 1st & then search to s_2 & match till s_3

```
if (i == s1.size() && j == s2.size() && k == s3.size()) {
    return true;
}
bool flag = false;
if (i < s1.size() && s1[i] == s3[k]) {
    flag = flag || Solve(s1, s2, s3, i+1, j, k+1);
}
if (j < s2.size() && s2[j] == s3[k]) {
    flag = flag || Solve (s1, s2, s3, i, j+1, k+1);
}
return flag
```

Min^m insertion steps to make a string Palindrome

i/p → s = mbadm

Reality - LPS

Longest Palindrome Subsequence

max → 3

∴ total length = 3

⇒ s - 3 = 2 ans

Min^m no. of Removals to Make Mountain Array

Make Array Strictly Increasing

Stone game

Stone game - II

Burst Balloons

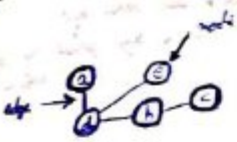
```
int Solve(vector<int> & nums, int start, int end){
    if(start > end) return 0;

    int coins = INT_MIN;
    for(int i = start; i <= end; i++){
        coins = max(coins, nums[start-1]*nums[i]*nums[end+1]
                + Solve(nums, start, i-1)
                + Solve(nums, i+1, end));
    }
    return coins;
}

int main(vector<int> & nums){
    nums.insert(nums.begin(), 1);
    nums.insert(nums.end(), 1);
    return Solve(nums, 1, nums.size() - 2);
}
```

Stone game - III

```
int Solve(vector<int> & sv, int i){
    if(i == sv.size()) return 0;

    int ans = INT_MIN;
    int total = 0;
    for(int x = 1; x <= 3; x++){
        if(i + x - 1 >= sv.size()) break;
        total += sv[i+x-1];
        ans = max(ans, total - Solve(sv, i+x));
    }
    return ans;
}
```

```
int Solve(vector<int> & Piles, int i, int M, int alice){
    if(i == piles.size())
        return 0;
    int ans = alice ? INT_MAX : INT_MIN;
    int total = 0;
    for(int x = 1; x <= 2*M; x++){
        if(i+x-1 >= Piles.size()) {
            break;
        }
        total += Piles[i+x-1];
        if(alice){
            ans = max(ans, total + solve(Piles, i+x, max(x, M), !alice));
        }
        else {
            ans = min(ans, solve(piles, i+x, max(x,M), !alice));
        }
    }
    return ans;
}
```

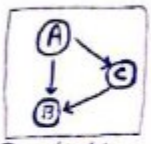
14- GRAPH



★ Remember 10 Real world application of graph

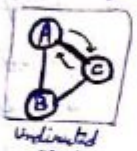
Types :-

- → Directed graph
- → Undirected graph



directed graph

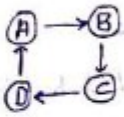
Edges :-
A→C
C→B
A→B



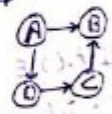
Undirected graph

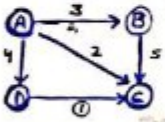
Edges :-
A-C
C-A
C-B
B-C
A-B
B-A

★ Cyclic graph



★ Acyclic graph



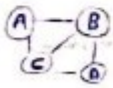


Edge E1 → Weight → 3

Weighted directed graph

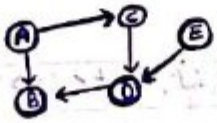
• by default weight → 1

Degree



Degree(B) = 3

▶ Indegree/outdegree



Indegree(A) = 0
Indegree(D) = 2
outdegree(D) = 1

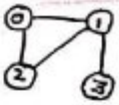
Path → 'kon node kon sa kisi kon node tak'
eg → A-C-D-B ✓
A-C-D-B-A ✗

• Graph Implementation :-

- → Adjacency Matrix → 2D array
- → Adjacency List → map/vector

⊛ Adjacency Matrix

	0	1	2	3
0	0	1	1	0
1	1	0	1	1
2	1	1	0	0
3	0	1	0	0



data implementation on VS

↓

basically hum input me diya hoga

0-1
1-0
2-0
0-2
1-2
2-1

→ hum kon
→ U=0, V=1
∴ adj[u][v] = 1

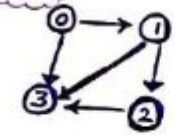
Kaise
aaj ke Pata kon
2D matrix kon sa

⊛ Adjacency List

Neighbour node

0 → {1, 3}
1 → {3, 2}
2 → {3}
3 → {3}

For undirected kon sa node kisi kon hoga



List of edges :-

[0→1]
[1→3]
[1→2]
[2→3]
[0→3]

• (Pair)

0 → {(1, 3), (3, 8)} (node, dist)
1 → {(2, 5)}
2 → {3}
3 → {(1, 6)}

```
int n;
cout << "Enter the no of nodes:" << endl;
cin >> n;
vector<vector<int>> adj(n, vector<int>(n,0));
int e;
cout << "Enter the no of edges" << endl;
cin >> e;

for(int i=0; i<e; i++){
    int u, v;
    cin >> u >> v;

    adj[u][v] = 1;
}

// Now the print the matrix
```

→ I/P :-

5
6
0 1
1 0
1 2
2 1
0 2
2 0

O/P →
0 1 1
1 0 1
1 1 0

Note → agar distance wali adj Matrix me karnie ho, Jisse ab tk 0 aur 1 kar rahe the, ab 0 aur distance mark karenge directly

T.C → O(n²)

T.C → Array → O(V+E)
Lo worst → O(V²)

```
* unordered_map < int, list < int >> adjlist;
```

HARD - P1

```
void addEdge ( int u, int v, bool direction) {
// direction = 0 -> undirected graph
// direction = 1 -> directed graph
}

adjlist [u]. push_back(v);   // create an edge from u to v
if ( direction == 0) {
    // undirected edge
    adjlist[v]. push_back(u);   // create an edge from v to u
}
}

void printAdjacencylist () {
    for (auto node : adjlist) {
        cout << node.first << "->" ;
        for (auto neighbour : node.second) {
            cout << neighbour ;
        }
    }
}
```

int, list < pair < int, int >>

→ 0 → {(1,5), (2,8)}

```
int main () {
    graph g ;
    g. addEdge (0,1,0);   (undirected edge)
    g. addEdge (1,2,0);
    g. addEdge (0,2,0);
    g. printAdjacencylist ();
}
```

→ For weighted graph → VS

YEPEE

```
map < int, list < int > > adj ;
```

0 → {1,2}
1 → {2}
2 → {3}

If weight is there

src node, list of nodes, dest node, weight

```
map < int, list < pair < int, int >>>
```

0 → {(1,5) (2,8)}
1 → {(2,6)}
2 → {3}

```
adjlist [u]. push_back ({v, weight})
```

```
template < typename T >
unordered_map <T, list < T >> adjlist;
void addEdge (T u, T v, ...)
```

→ used for generic char/int → make type v store kaun ka ?

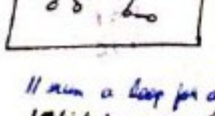
```
int main () {
    graph < char > g
    g. addEdge ('a', 'b', 0);
}
```

★ Traversal

- BFS → (Breadth first search)
- DFS → (Depth first search)

- queue
- adj list
- visited

Special case



→ graph

```
// run a loop for all nodes
for (int i = 0; i < n ; i++) {
    if (! visited [i], visited) {
        bfs();
```

BFS

```
void bfs(int src) {
    queue < int > q;
    unordered_map < int, bool > visited
    q. push (src);
    visited [src] = true;
while ( !q. empty () ) {
    int frontNode = q. front();
    q. pop();
    cout << frontNode;
    // insert neighbours
    for (auto neighbour : adjlist [frontNode]) {
        if (! visited [neighbour]) {
            q. push (neighbour);
            visited [neighbour] = true;
        }
    }
}
```

DFS

I → Print
II → visited Mark
III → neighbour → recu call

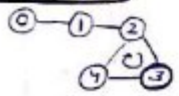
```
void dfs ( int src, unordered_map < int, bool > & visited) {
    cout << src ;  → cout
    visited [src] = true;  → visited

  for(auto neighbour : adjlist [src]) {
      if (! visited [neighbour]) {
        dfs (neighbour , visited);  → Recurse
      }
    }
  }
```

★ **Cycle Detection** (V.Imp) (Use → Solve all 4 ways) → [Pehchan enough, but 100 me 90% Dikkat ek time Language (Kyunki Practice ek Mode & Log)]

- Directed
  - BFS [adjlist & [mark-mark] → Implicit → ...]
  - DFS (visited & rec + ...)
- Un-directed
  - BFS
  - DFS

(Same logic) → (already visited & nbr != Parent)

Un-directed, using BFS



→ condition: already visited & Not Parent

3 → 4

0	1	2	3	4

0 → true
1 → true
2 → true
3 → true
4 → true

Parent

0 → -1
1 → 0
⋮

- bool checkCycleUsingBFS (int src, unordered_map< int , bool > & visited) {
    queue < int > q;
    unordered_map < int, int > Parent;

    q.push (src);
    visited [src] = true;
    Parent [src] = -1;

    while (!q.empty()) {
        int frontNode = q.front();
        q.Pop();

        for(auto neighbour : adjlist [frontNode]) {
            if (! visited [neighbour]) {
                q.push (neighbour);
                visited [neighbour] = true;
                Parent [neighbour] = frontNode;
            }
            else {
                // already visited
                if (neighbour != Parent[frontNode]) {
                    // cycle Present
                    return true;
                }
            }
        }
    }
    return false;
}

- <u>Cycle Detection of undirected using DFS</u>

- bool checkCyclicUsingDFS (int src, unordered_map<int, bool> & visited, int Parent) {

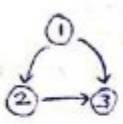
```
    visited [src] = true;
    for (auto nbr : adjlist [src]) {
        if (! visited [nbr]) {
            bool checkAgeKaAns = checkCyclicUsingDFS (nbr, visited, src);
            if (checkAgeKaAns == true)
                return true;
        }
        if (visited [nbr] && nbr != Parent) {
            // cycle Present
            return
        }
    }
    return false;
}
```

Very Easy & Very Important

Topological Sort :-

↳ Linear ordering of vertices such that for every edge $U \to V$, U comes before V in that order.

E.g.



$1 \to 2$
$1 \to 3$
$2 \to 3$

∴ Valid TS :-

$1 \to 2 \to 3$ ✓ (valid)

$2 \to 1 \to 3$ ✗ (Invalid)

- T.S → condition ⇒ Directed Acyclic graph (DAG) → "cycle me ordering hi nahi hoga"

DFS & BFS

Bridge in a graph (agar ek edge hta do to wo graph 2 me divide ho jaaye)

Shortest time to rot all oranges (1 sec. 1 rot)

o/p → 6 sec

[Ab DFS/BFS se hi hoga lekin bas Bahar bna ke Rich dikh]

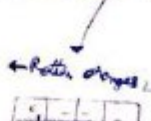
→ [Rot hoenge = count++]

No of cells

• Topic Limited but Q-Rich Ke Samajhne hoga Kisne Kaun Sa Concept Lagaya ??

Example

EX-

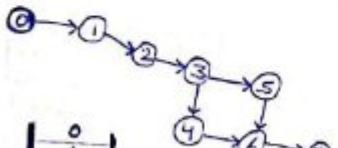


→ no of components?

1 wale Jitne h wo, Island bnte nhi h, to total Kitne island h?

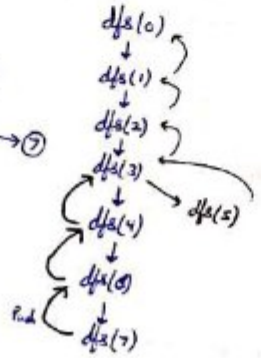
o/p → 3

Approach



0
1
2
3
5
4
6
7

Stack



→ T.S valid ⇒ $\boxed{0, 1, 2, 3, 5, 4, 6, 7}$ Ans

→ (Kuch nhi h, bas DFS call karna, aur wapas jate waqt stack me dal do)

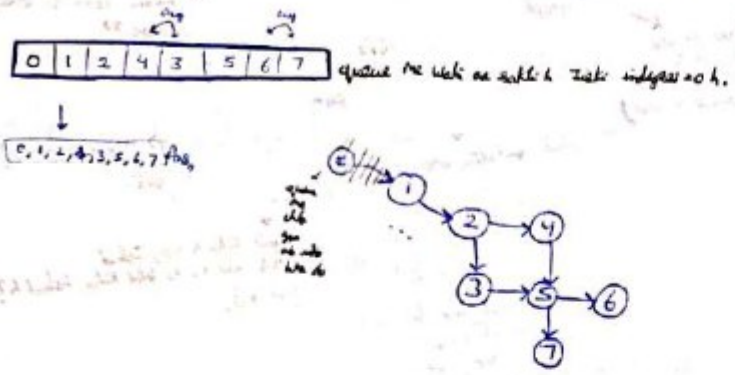
• Void topoSortDFS(int src, unordered_map<int, bool> & visited, stack<int> & ans) {

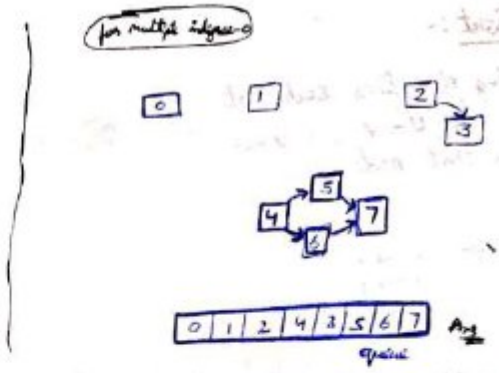
```
    visited[src] = true;
    for(auto neighbour : adjList[src]) {
        if (!visited[neighbour]) {
            topoSortDFS(neighbour, visited, ans
        }
    }
    // while returning, store the node in stack
    ans.push(src);
}
```

Approach

```
Unordered_map<int, bool> visited;
Stack<int> ans;
for (int i=0; i<n; i++) {
    if (!visited[i]) {
        g.topoSortDfs(i, visited, ans);
    }
}
while(!ans.empty()) {
    cout << ans.top();
    ans.pop
```

ab, isi ko BFS se karte h (also known as Kahn's algo)





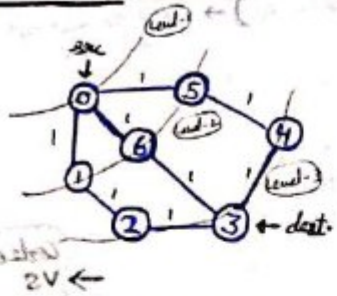

```
• void topoSortBfs (int n, vector < int > & ans) {
    queue < int > q;
    Unordered_map < int, int > indegree;
    // indegree calculate
    for (auto i: adjlist) {
        int src = i.first;
        for (auto nbr: i.second) {
            indegree[nbr]++;
        }
    }
    // put all nodes inside queue, which has indegree = 0
    for (int i = 0; i < n; i++) {
        if (indegree[i] == 0) {
            q.push(i);
        }
    }
    // bfs logic
    while (!q.empty()) {
        int fNode = q.front();
        q.pop();
        ans.push_back(fNode);
        for (auto nbr: adjlist[fNode]) {
            indegree[nbr]--;
            // check for zero again
            if (indegree[nbr] == 0) {
                q.push(nbr);
            }
        }
```

• Graph
↳ cycle exist or not
↳ using BFS
↓
Bre-respects (Kahn's algo)

Invalid T.S → cycle present → T.S ki length ≠ n

Valid TS → ∴ DAG (No cycle)

Shortest Path :-



Shortest Path :-

$0 \to 5 \to 4 \to 3$

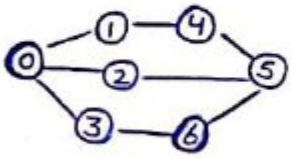
$\boxed{0 \to 6 \to 3}$ ✓

$0 \to 1 \to 2 \to 3$

$\boxed{\text{Let be weight} = 1}$ (for all)

- BFS Me Jab Pehli baar us Par Pahunchte h, to wahi wali shortest path hoti h
(Jab Same weight)

- Approach



0	1	2	3	4	5	6

queue

Parent
0 → -1
1 → 0
2 → 0
3 → 0
4 → 1
5 → 2
6 → 3

Visited
0 → ~~F~~ T
1 → ~~F~~ T
2 → ~~F~~ T
3 → ~~F~~ T
4 → ~~F~~ T
5 → ~~F~~ T
6 → ~~F~~ T

→ (Mark in VS)

↓

Hum bas yaad rakhlenge,

↓

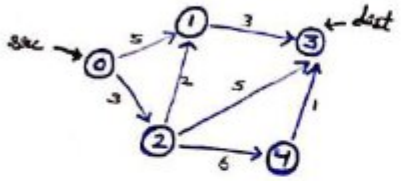
Parent ko array Banlenge,

	0	1	2	3	4	5	6
Parent →	-1	0	0	0	1	2	3

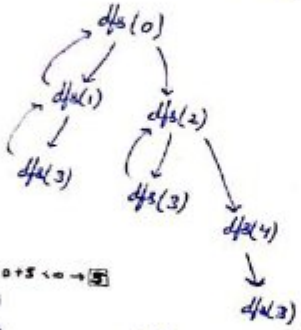
src → 0
dest = 5

5 → 2 → 0 → Now, Just reverse
dest src

- For diff weight



"• src - Same one explore kare, Phir min Nikal le"

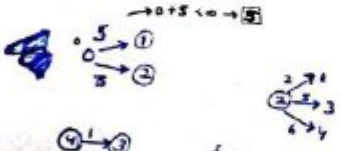


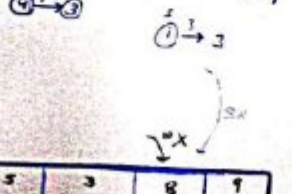
⇒ ~~the find min()~~

① find Topological sorting order
↓
Linear order

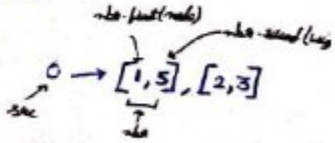
0
2
4
1
3

Stack





adjlist



	0	1	2	3	4
dist →	0	5	3	8	9
	0	1	2	3	4

(Dist) 0-0 0-1 0-2 0-3 0-4

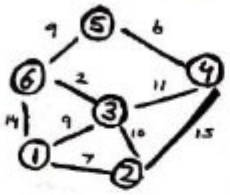
Dijkstra Algo

↳ Shortest dist

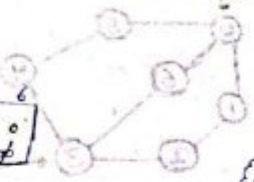
→ basically bfs based h,

(Set ya Priority Queue) → to find the min dist vertex Node

src → 6
dest → 4



	0	1	2	3	4	5	6
dist.	∞	~~14~~ 11 ϕ	12 ϕ	2 ϕ	13 ϕ	9 ϕ	0 ϕ



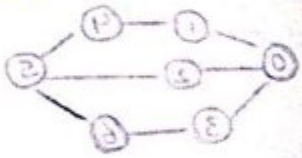
→ Codeon VS

{(11,1)
(9,5)
(2,3)

! set
↳ elet → (0,6)
dist node

(13,2)
(13,4)
(9,5)
(11,1)
(2,3)
(0,6)

{ ∞, 11, 12, 2, 13, 9, 0 }



(Leetcode - 547)

→ Matrix se list bna ke kar lage (Ichal ka)

#5 Q- Number of Provinces

1 — 2
1
3

(Indirectly - no. of components mang rha h)

MSB ---- LSB

Bitwise operator

- & – and
- | – or
- ^ – xor
- ~ – not
- << – left shift → (1 ko ek baar left shift karunga to wo '10' bn jayega binary ke andar) → ∴ multiply by 2
- >> – right shift → divide by 2 ho jata h

Sirf h, dimag me sochne ki Jaroorat nhi

Decimal to Binary

$n = 10$

	Rem
$\frac{10}{2} = 5$	0
$\frac{5}{2} = 2$	1
$\frac{2}{2} = 1$	0
$\frac{1}{2} = 0$	1

∴ 1010 Ans

1. divide by 2
2. store remainder in an array
3. repeat above 2 till $n \neq 0$
4. Reverse the answer

OR

$n = 5$

```
while(n!=0) {
    bit = n%1;
    n = n>>1;
}
```

→ answer me store karte jao ek bit ko

```
int ans=0;
int i=0;
while (n!=0) {
    int bit = n%1;
    ans = (bit × Pow(10,i)) + ans;
    n = n>>1;
    i = i+1;
}
cout << ans;
```

$(10^i \times bit) + ans$

2^i

%10

/10

bas ye yad rakho, baki sab hojayegi chija

Binary to Decimal

10101 → ?

$2^4 + 2^2 + 2^0 = 16 + 4 + 1 = 21$

```
int i=0, ans=0;
while(n!=0) {
    int digit = n%10;
    if (digit==1) {
        ans = ans + Pow(2, i);
    }
    n = n/10;
    i++;
}
cout << ans;
```

Kisi kaam o karna h to 2.0 karta hu

-1 = 1111 1111

1 << i → 000100

0 by default

BITWISE OPERATORS

a	b	|	&	^
0	0	0	0	0
0	1	0	1	1
1	0	0	1	1
1	1	1	1	0

eg. $<<i$ → $\times 2^i$ → multiply with 2 power i

• get i^{th} bit

decimal no. → $n=10$, $i=3^{rd}$

Binary representation

0 0 0 0 0 0 0 1 0 1 0

& 0 0 0 0 0 0 1 0 0 0 → Mask was generated

& → 1 → bit-1

& → 0 → bit-0

```
★ int getIthBit(int n, int i) {
    int mask = 1 << i;
    int ans = n & mask;
    if (ans == 0) {
        return 0;
    else
        return 1;
    }
```

• int a = 0 → shortcut

~a = -1 → 32 bit ki bat karega to sab assign

• **and/odd**

```
★ int main() {
    int n = 5;
    if(n & 1) {
        cout << "n is odd" << endl;
    }
    else {
        cout << "n is even" << endl;
    }
```

3 = 11
5 = 101
7 = 111

LSB ko '&' karo 1 ke sath agar 1 aaya to odd nhi to 0

• **Set i^{th} bit**

→ $n=10$, $i=2$ → o/p → 14

00000 1 0 1 0

1 0 0 0 0 1 0 0 → Mask

0 0 0 0 1 1 1 0

```
★ int mask = 1 << i;
  int ans = n | mask;
  return ans;
```

• **clear i^{th} bit**

$n=10$, $i=1$

00000 0 1 0 1 0

& 1 1 1 1 1 1 1 1 1 1 0 1

0 0 0 0 0 0 0 1 0 0 0

```
int mask = ~(1<<i);
int ans = n & mask;
return ans;
```

(bacaudize code)

```
★ void clearIthBit(int & n, int i) {
    int mask = ~(1<<i);
    n = n & mask;
  }

  void updateIthBit(int & n, int i, int target) {
    clearIthBit(n, i);
    int mask = target << i;
    n = n | mask
    cout << n

  int n = 10;
  updateIthBit(n, 0, 1);
```

→ (ho ki 0th bit ko 1 karda)

$n = 1^0$?

n = 0000010

~(1<<i)

• last 4 bit 0 karna h, to mask create karna sikh lo, (Wahi Dimag lagaye bas)

```
  00000 11111111111
$ 11111 111111 0000 → Mask
  111111 0000
```

★
```
void clearLastIthBits(int n, int i) {
    int mask = (-1<<i);
    n = n & mask;
    cout << n;
```

• check the Power of 2,

16 → 100000
8 → 10000
7 → 0111

∴ logic → (8 & 7 = 0)

★
```
if (n & (n-1) == 0)
    return true;
else
    return false;
```

• Count the no. of Set bits

★
```
int count = 0
while (n != 0) {
    int lastBit = n & 1;
    if (lastBit) {
        count ++;
    }
    // right shift
    n = n>>1;
}
return count;
```

(Dekhne ka)

→ logn

OR

★
```
int count = 0
while (n != 0) {
    // remove last set bit
    n = (n & (n-1));
    count ++;
}
return count;
```

i/p → 1110 1110 1111 (i, j)

clear

o/p → 1110 0000 0000 1111

clear bits in a range

mask a → -1 << (i+1) → 1111 0000 0000

b → (1<<j) - 1 → 0000 1111

(0000 1000) - 1 = 0000 0111

as we know, -1 = 1111 1111 1111 with 1's since left shift ✓

```
★ int a = (-1<<(i+1));
  int b = (1<<j)-1;
  int mask = a|b
  n = n & mask
  cout << n;
```

n = 1110 0111 0111 (i, j)

1110 0000 0000 1111 ← mask

(-1<<i+1) ← a = 1111 0000 0000 0000

b = 0000 0000 0000 1111

(1<<j) - 1

a|b → mask

mask & n ⇒ answer

Main step - 2^n me le sakta hu bas, find min no. of steps to reach opponent?

```
while (n!=0){
    n & (n-1)
    count ++
}
```

Greedy Algorithm (Easii Topic but IMPORTANT)

Q.1) **Minimum Platform**

→ Find the minimum no. of Platforms required for the railway station so that no train is kept waiting.

i/p → n = 6, arr[] = {0900, 0940, 0950, 1100, 1500, 1800}
dep[] = {0910, 1200, 1120, 1130, 1900, 2000}

o/p → 3

```
int solve( int n, int arr[], int dept[]){
    vector <Pair <it, it>> data;
    for(int i=0; i<n; i++){
        data.push_back({arr[i], dept[i]});
    }
    sort (data.begin(), data.end(), cmp);

    int count = 1;
    int lastDept = data[0].second;
    for(int i=1; i<n; i++){
        if (data[i].first >= lastDept){
            count ++;
            lastDept = data[i].second;
        }
    }
    return count;
}
```

```
bool cmp(Pair <int, int> a, Pair <int, int> b){
    return a.second < b.second;
}
```

→ (dep ke according data ko bada sort)

Q - 1 Platform

1 day me kitni train pass ho sakti h

(Ye top level wala Q nhi h)

sort both array → {0900, 0940, 0950, 1100, 1500, 1800}
{0910, 1120, 1130, 1200, 1900, 2000}

max = 3

```
int solve (int arr[], int dep[], int n){
    int count = 1;
    int Platform = 1;
    int i = 1;
    int j = 0;
    while (i<n && j<n){
        if (arr[i]<=dep[j]){
            count ++;
            i++;
        }
        else {
            count--;
            j++;
        }
        Platform = max(Platform, count);
    }
    return Platform;
}

int find (int arr[], int dep[], int n){
    sort (arr, arr+n);
    sort (dep, dep+n);
    return Solve (arr,dep,n)
}
```

a) Buy 2 lollipop and get 1 free, ~~but free only if must be less price than the buying lollipop~~

(min^m of 2 lollies you buy) → Isse kaise than

(Min^m Price Khach Kate Sasti lollipop kaisi)

▷ Sort

[2, 2, 5, 6, 7, 9]

Free, Buy, Buy, free

i/p → [6, 5, 7, 9, 2, 2]

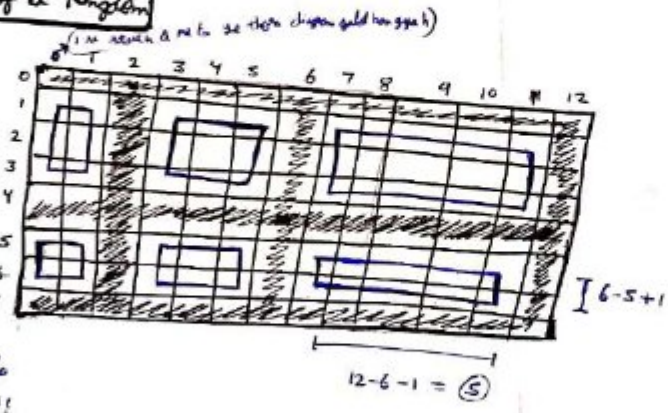
o/p → 7+9+5+2 = 23 Ans.

b) DEFKIN - Defense of a Kingdom

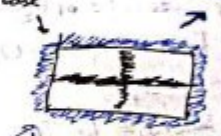
Not Defense?

(Row Ka Max^m diff nikal lo Col Ka Max diff nikal lo Dono Ko multiply Karlo)

Diff Ko Pehle, Row Ko Sort Karlo & Col Ko sort Karlo!!



edge case



boundary Ko Ki hai dunga

(-1 hai do)

-1 → Put 2 imaginary towers taki boundary se v area ka Pata!!

(Then I based indexing handle h to '0' Ko Check Karlo)

x → width
y → height
m → Tower

n → width
h → hei. of Tower

```
▷ int x, y, m;
  cin >> x >> y >> m;
  int t = m;
  vector<int> rsvC;
  vector<int> cdC;

  rsvC.push_back(0);
  cdC.push_back(0);

  while(t--) {
    int a, b;
    cin >> a >> b;
    rsvC.push_back(a);
    cdC.push_back(b);
  }

  rsvC.push_back(x+1);
  cdC.push_back(y+1);

  sort(rsvC.begin(), rsvC.end());
  sort(cdC.begin(), cdC.end());
```

```
int maxlen = INT_MIN;
for (int i=1; i< rsvC.size(); i++) {
    int a = rsvC[i-1];
    int b = rsvC[i];

    maxlen = max(maxlen, b-a-1)
}

int maxWidth = INT_MIN;
for (int i = 1; i< cdC.size(); i++) {
    int a = cdC[i-1];
    int b = cdC[i];

    maxWidth = max(maxWidth, b-a-1);
}

cout << MaxWidth x MaxLen;
```

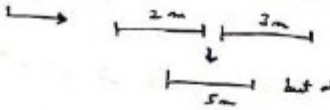
i/p

1
15 8 3
3 8
11 2
8 6

o/p

12

9) Min cost of ropes



but at cost (2+3) = 5

▷ To 2 chote h unko
Jodte Jaate zao → $\boxed{\text{heap}}$

{4, 3, 2, 6} 5₹
↓
{4, 5, 6} 9₹
{9, 6} 15₹
{15} ⊕
29₹

```
Priority-queue < int, vector < int >, greater < int >> Pq;
    for(int i=0; i<n; i++) {
        Pq. Push (arr [i]);
    }
int cost = 0;
while (Pq. size () != 0) {
    int first = Pq. top ();
    Pq. Pop ();
    int second = Pq. top ();
    Pq. Pop ();

cost += first + second;
int newlength = first + second;
Pq. Push (newlength);
}
return cost;
```

[redacted]

9) Fractional KnapSack

↳ (fraction me bhi le sakte h)

▷ $\frac{\text{Val}}{\text{weight}}$ ko sort (desc. order)

▷ one-by-one Pick
↳ agar capacity → take full
↳ else → fraction lot

▷ (1.0) multiply for value i double

```
bool cmp (Pair < int, int > a, Pair < int, int > b) {
    double ratio1 = ((1.0)*a. first) / a. second;
    double ratio2 = ((1.0)*b. first / b. second;
    return ratio1 > ratio2
}

int main () {
    int Val [] = {60, 100, 120};
    int wt [] = {10, 20, 30};
    int n = 3;
    int capacity = 50;

vector < Pair < int, int >> data;
for (int i=0; i<n; i++) {
    data. Push_back ({ Val [i], wt [i]});
}
sort (data. begin(), data. end(), cmp);
int totalValue = 0;
// check each items k entire item lele ya fraction lu
for(int i=0; i<n; i++) {
    Pair < int, int > item = data [i];
    int itemValue = item. first;
    int itemWeight = item. second;
// entire inclusion wala case
    if (itemWeight <= capacity) {
        totalValue += itemValue;
        capacity = capacity - itemWeight;
    }
    else {
        // fraction include karo
        double ratio = ((1.0)* itemValue / itemWeight;
        int valueToAdd = ratio * capacity;
        TotalValue += valueToAdd;
        capacity = 0.
    }
```

4) Chocolate Distribution Problem

$N = 8$, $M = 5$ → students

students

arr = {3, 4, 1, 9, 56, 7, 9, 12}

→ o/p → 6

↳ no. of chocolates

▷ Distribute chocolates among M students
- ↳ exactly 1 packet
- ↳ diff b/w max. no of chocolates given to a student & min no. of chocolates given to a student is minimum

▷ Sort

1, 3, 4, 7, 9, 9, 12, 56

∴ $\boxed{Min = 6}$

2 pointer approach do hoga (ek se) hoga Jayega

```
Sort (arr, arr + n);
//check diff in m-size window
int i = 0;
int j = m-1;
int diff = INT_MAX;
while(j<n) {
    int newDiff = arr[j] - arr[i];
    diff = min(diff, newDiff);
    j++;
    i++;
}
return diff;
```

★★ 4 More Questions

5) N-Meeting in one Room (or activity Selection)

↳ Some a → love babbar a → 1 platform how many train

6) Job Sequencing Problem

↳ Max^m Profit (& before deadline)

each job takes 1 unit to complete

N = 4

jobs = {(1, 4, 20), (2, 1, 10), (3, 1, 40), (4, 1, 30)}

id, deadline, profit

▷ Sort accord to Profit

& check for 'd' as deadline !!

EXTRA EFFORT [PSG - Special]

Q → Remove Duplicate from sorted array

i/p → [0,0,1,1,1,2,2,3,3,4]

o/p → 5 , nums = [0,1,2,3,4, ---]

```
int Duplicate (vector<int>& nums)
    int i=0;
    for(int j=1; j< nums.size(); j++) {
        if (nums[i] != nums[j]) {
            nums[i+1] = nums[j];
            i++;
        }
    }
    return i+1;
};
```

Q Find all duplicates in an array

i/p → nums = [4,3,2,7,8,2,3,1]

o/p → [2,3]

(Pre)

```
unordered_map<int, int> hash;
for(int i=0; i< nums.size(); i++){
    hash[nums[i]]++;
}
vector<int> ans;
for(auto i : hash){
    if (i.second > 1){
        ans.push_back(i.first);
    }
}
return ans;
```

(Pro) (Efficient)

```
vector<int> ans;
for (int i=0; i< nums.size(); i++){
    int x = abs(nums[i]);
    if (nums[x-1] < 0) {
        ans.push_back(x);
    }
    nums[x-1] = nums[x-1] * (-1);
}
return ans;
```

▶ Make array Strictly Increasing

```
int Solve(vector<int>& arr1, vector<int>& arr2, int prev, int i) {
    if (i == arr1.size()) {
        return 0;
    }
    int op1 = INT_MAX;
    if (prev < arr1[i]) {
        op1 = solve(arr1, arr2, arr[i], i+1);   //no-operation
    }

    int op2 = INT_MAX;
    auto it = upper_bound(arr2.begin(), arr2.end(), prev);
    if (it != arr2.end()) {
        int index = it - arr2.begin();

        arr1[i] = arr2[index]

        op2 = 1 + solve(arr1, arr2, arr2[index], i+1)

    }
    return min(op1, op2)

int makearray( ---- ) {
    sort(arr2.begin(), arr2.end);
    return solve(arr1, arr2, -1, 0);
```

prev so that hum element hum data

pick index k

→ so line se first array ki modified hogyi.

Prev wala element ki bhai data h as prev and count++ hoga

▶ Min^m no. of Removals to Make Mountain array

↳ Just VS (very easy if all conditions are clear)

$\boxed{s = \text{int}}$ ∴

MERGE SORT

→ based on Divide & conquer

- ⓐ Mid find
- ⓑ break 2 half
- ⓒ Recursion se 2 half ko sort karlige
- ⓓ Main se 2 half ko Merge karlige

→ ye kaise karoge?

S → merge 2 sorted array

i/p → arr1 = [2|4|6]

arr2 = [3|5|7|9|10]

2 Ptr: ek ek ko compare kar ke merge

```
void merge(int* arr, int s, int e){
    int mid = (s+e)/2;
    int len1 = mid-s+1;
    int len2 = e-mid;

    int* left = new int[len1];
    int* right = new int[len2];

//copy values
    int k=s;
    for(int i=0; i<len1; i++){
        left[i] = arr[k];
        k++;
    }
    k = mid+1;
    for(int i=0; i<len2; i++){
        right[i] = arr[k];
        k++;
    }

//merge logic
    int leftIndex=0;
    int rightIndex=0;
    int mainArrayIndex=s;
    while(leftIndex<len1 && rightIndex<len2){
        if(left[leftIndex]<right[rightIndex]){
            arr[mainArrayIndex++] = left[leftIndex];
        }
        else{
            arr[mainArrayIndex++] = right[rightIndex++];
        }
    }
//copy logic for leftArray
    while(leftIndex<len1){
        arr[mainArrayIndex++] = left[leftIndex++];
    }
//copy logic for rightArray
    while(rightIndex<len2){
        arr[mainArrayIndex++] = right[rightIndex++];

    Delete left[];
    Delete right[];
```

```
void mergeSort(int* arr, int s, int e){
//Base case
    if(s>=e)
        return;

    int mid = (s+e)/2;
//left part sort karne recursion
    mergeSort(arr, s, mid);
//right part sort karne recursion
    mergeSort(arr, mid+1, e);

//Now merge 2 sorted array
    merge(arr, s, e);
}

int main(){
    int arr[] = {4,5,13,2,12};
    int n = 5;
    int s=0;
    int e = n-1;
    mergeSort(arr,s,e);

    for(int i=0; i<n; i++){
        cout << arr[i];
    }
    return 0;
}
```

→ using vector – on VS

using array

DSA → Chapter 7 → Run

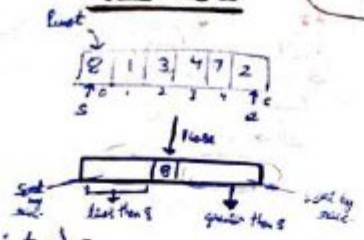
② ...

ⓑ Part 2 ⓒ

```
void merge(vector<int>&v, vector<int>&temp, int start,
    int mid, int end){
    int i = start, j = mid+1, k = start;
    while(i<=mid && j<=end){
        if(v[i]<=v[j]) temp[k++] = v[i++];
        else
            temp[k++]=v[j++];
    }
    while(i<=mid) temp[k++]= v[i++];
    while(j<=end) temp[k++] =v[j++];
    while(start<=end){
        v[start] = temp[start],
        start++;
    }
}
void mergeSort(vector<int>&v, vector<int>&temp,
    int start, int end){
    if(start >= end) return;
    int mid = (start+end)>>1;
    mergeSort(v, temp, start, mid);
    mergeSort(v, temp, mid+1, end);
    merge(v, temp, start, mid, end);
}
```

Quick Sort

(Partition logic + recursion)



① has I no. ko sahi Jaga le Pahuchna da
② Fir ki recursion Sambhal lega

using Partition logic (us no. ke left side usse chote hange aur us no. ke right side usse bade hange)

Pivot element ko sahi Jaga dete

(Kaun v ho sakta h)

```
int Partition(int arr[], int s, int e) {
    // Step1: choose PivotElement
    int PivotIndex = s;
    int PivotElement = arr[s];

    // Step2: find right Position for Pivot element and place it there
    int count = 0;
    for(int i = s+1 ; i<=e; i++) {
        if(arr[i] <= PivotElement) {
            count++;
        }
    }
    // Jab main loop se bahar aaya, to mere pas Pivot ki right Position
    ka index ready h
    int rightIndex = s+ count;
    Swap(arr[PivotIndex], arr[rightIndex]);
    PivotIndex = rightIndex;    // update PivotIndex

    // Step3: left me chote aur right me bade
    int i = s;
    int j = e;
    while(i< PivotIndex && j> PivotIndex) {
        while(arr[i] <=PivotElement) {
            i++;
        }
        while(arr[j] > PivotElement) {
            j--;
        }
    // Ab 2 case ho sakta h
    // A → you found the elements to swap
    // B → no need to swap
        if( i< PivotIndex && j> PivotIndex) {
            Swap(arr[i], arr[j]);
        }
    }
    return PivotIndex;
```

(count kaun sahi ka ho sakta hai, sahi kitne chote element h ttl)

```
→ void quickSort(int arr[], int s, int e) {
    // base case
    if(s>=e)
        return;
    // Partition logic, return PivotIndex
    int p = Partition(arr, s, e);
```

index milegya

```
    // Recursive calls
    // Pivot element → left
        quickSort (arr, s, p-1);

    // Pivot element → right
        quickSort (arr, p+1, e);

→ int main() {
    int arr[] = {8,1,3,4,20,50,30};
    int n = 7;

    int s=0;
    int e = n-1;
    quickSort (arr, s, e);
    for(int i=0; i<n; i++) {
        cout << arr[i];
    }
    return 0;
}
```

★ Selection Sort

```
for(int i=0; i<n-1; i++) {
    int minIndex = i;
    for(int j= i+1; j<n; j++) {
        if (arr[j] < arr[minIndex]) {
            minIndex = j;
        }
    }
    Swap (arr[minIndex], arr[i]);
}
```

best → $O(n^2)$
↑
worst

★ Insertion Sort

```
for(int i=1; i<n; i++) {
    int temp = arr[i];
    int j = i-1;
    for ( ; j>=0; j--) {
        if (arr[j] > temp) {
            // shift
            arr[j+1] = arr[j];
        }
        else {
            break;
        }
    }
```

worst → $O(n^2)$
↘
best → $O(n)$

★ Bubble Sort

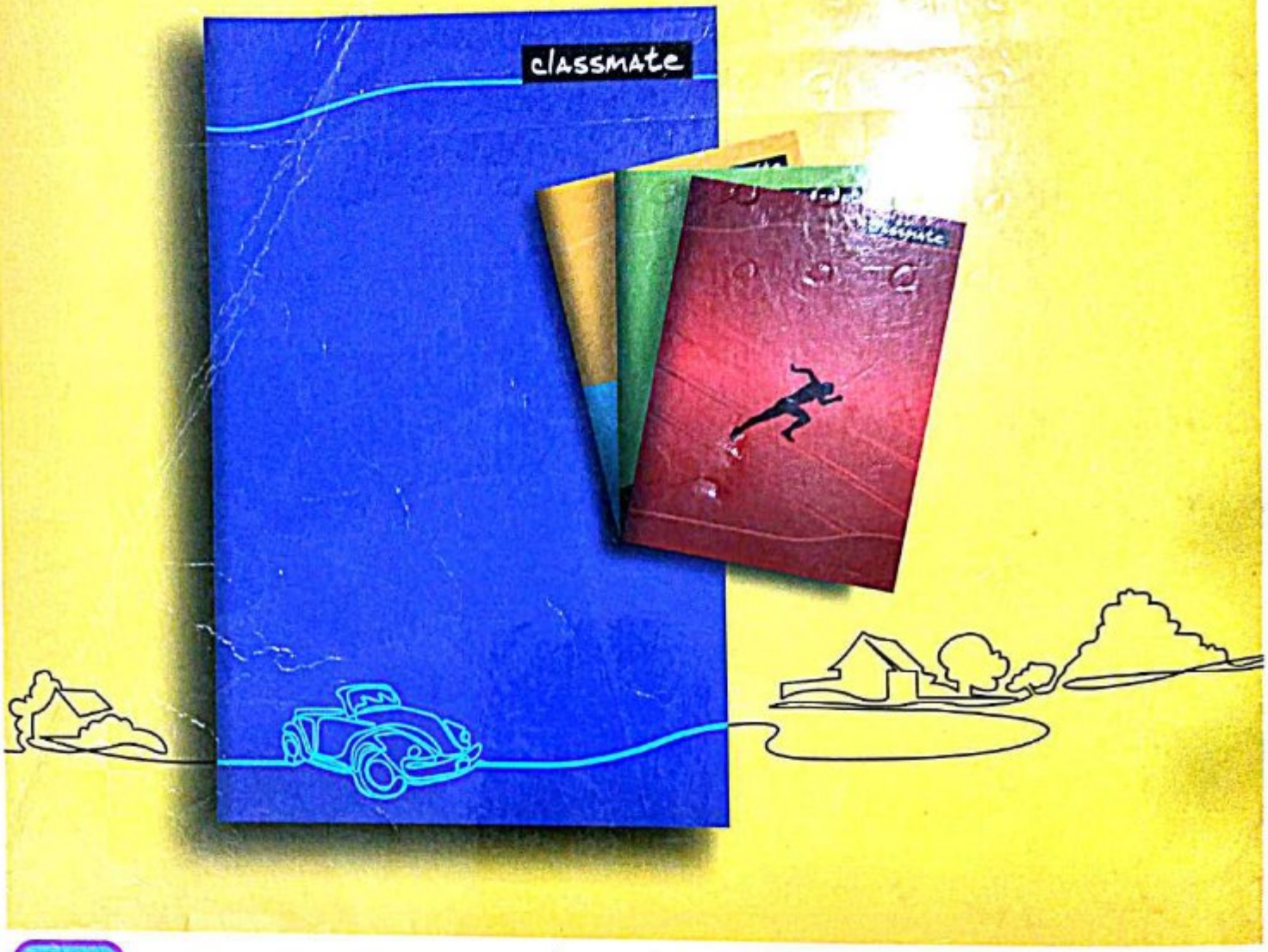
```
for(int i=1; i<n; i++) {
    for(int j=0; j<n-1; j++) {
        if(arr[j] > arr[j+1]) {
            Swap (arr[j], arr[j+1])
        }
    }
}
```

worst → $O(n^2)$
↘
best → $O(n)$

Enduring Value

classmate
ENJOY LEARNING

NEAT DESIGNS.
FOR THE SMART ONES.



my classmate app

Learn while you play story-based skill development games & experience AR based concepts

Download the App now by scanning the QR code

ITC's Primary Education Programme is designed to provide children from weaker sections access to education with focus on learning outcomes and retention. The interventions focus on universal access and retention, bridging of gender and social category gaps in elementary education and improving the quality of learning. Operational in 13 states, the programme has benefited 8,64,000 children till March 2022.

Enduring Value

Let's put India first

Customize your notebook covers at **www.classmateshop.com**

classmate ALL ROUNDER
National Level Multi-Skill Competition

FEEDBACK ? SUGGESTIONS ?
Quality Manager, ITC Ltd.- ESPB, ITC Centre, 5th Floor, 760, Anna Salai, Chennai. 600 002.
classmate@itc.in 18004253242

A quality product marketed by

Enduring Value

Exercise Book
140 Pages
(Total Pages Include Index & Printed Information)
Size : 29.7 x 21 cm
MRP Rs. 63.00
Inclusive of all taxes

Batch: D/AG/JC
02000200
8902519002006
©ITC Limited

Type of Ruling :
Unruled