SEARCHING && SORTING

LEVEL-2

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

Q Find Pivot element. [It's a sub part of search in a rotated & sorted array (33 Leetcode)]

Sorted

2	4	6	8	10	12	14	16

Pivot [max. no] in this case

Rotated & Sorted

12	14	16	2	4	6	8	10

Increasing ORDER — INCREASING ORDER

∴ Some elements are rotated here

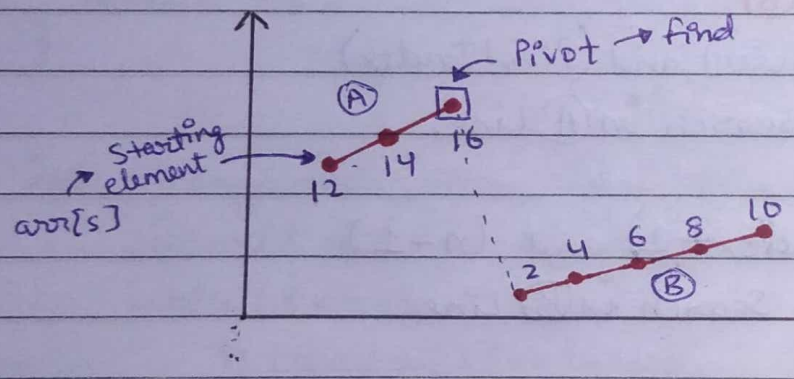
∵ Definition of pivit element is different in different resources.

Ex:-

Increasing, Decrease, Increasing

arr →

12	14	16	2	4	6	8	10
0	1	2	3	4	5	6	7



Approaches

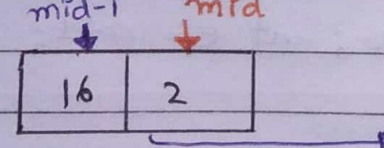
#1 → Linear Search → O(n)

#2 → Sort (Asc/Desc) → O(n log n)

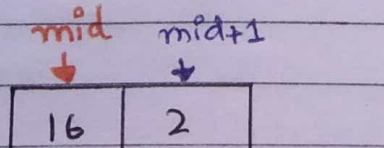
#3 → BS → O(log n)

∵ We can apply Binary search in (A) and (B) but we have to handle 16 and 2 seperately.

** [mid-1] >= 0 love Bhaiya cond^n lagana bhul gye the so, remember that [mid+1] < n



(I) arr[mid] < arr[mid-1] → ans → (mid-1)



(II) arr[mid] > arr[mid+1] → ans = (mid)

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

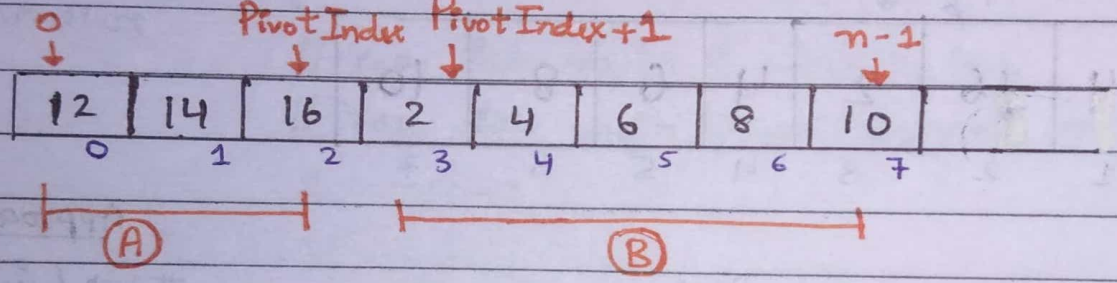
∴ Logic for (A) and (B) Line elements

(III) if arr[s] > arr[mid] → (B) Line
↳ left me jao

(IV) else → (A) Line
↳ right me jao

∴ Corner case s==e i.e when array have single element.
↳ return e bcz its the pivot element here.

↳ where to apply Binary search? in (A) Line or in (B) Line.



∴ If target is between (start index) and (PivotIndex) then apply Binary search in (A) Line.

If target is between (PivotIndex+1) and (n-1) then apply Binary search in (B) Line

Code:-

```
int findPivotIndex (vector<int> & arr) {
    int n = arr.size(); int s=0; int e = n-1;
    int mid = s+(e-s)/2;
    while ( s<=e) {
    // Corner Case
    if ( s ==e ) {
            // single element
            return e;
    }
```

Spir

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

```
    else if (mid-1 >=0 && arr[mid] < arr[mid-1]){
            return mid-1;
    }
    elseif (mid+1 <n && arr[mid] > arr[mid+1]) {
            return mid;
    }
    else if (arr[s] > arr[mid]){  // On (B) Line
            e = mid - 1;          // Left me jao
    }
    else {                        // On (A) Line
        s = mid +1;               // Right me jao
    }
    // yaha galti ni Karni so, update mid
    mid = s + (e- s)/2;
}
  return -1;
}

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

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

```
int search(vector<int>& nums, int target) {
    int n = nums.size();
    int pivotIndex = findPivotIndex(nums);
    int ans = -1;

    // search in (A)
    if (target >= nums[0] && target <= nums[pivotIndex]) {
        ans = binarySearch(nums, 0, pivotIndex, target);
    }
    else {
        ans = binarysearch(nums, pivotIndex+1, n-1, target);
    }
    return ans;
}
```

3) Sqrt(x), (69 leetcode)

i/p → Number = x

o/p → $\sqrt{x}$ → ans

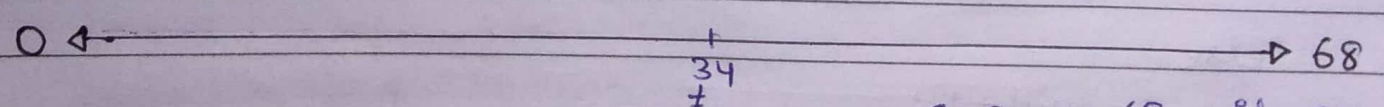
Ex:- i/p → 25 , 36 [Find it in O(log n) T.C]

o/p → $\sqrt{25} = 5$, $\sqrt{36} = 6$

Search space minimisation :-

Possibility of answers within a range.

Ex:- x = 68 , possible answers of $\sqrt{68}$ are its search space.



s = 0, e = 68, mid = 34

Predicate func^n

34 x 34 > 68 → left me jao

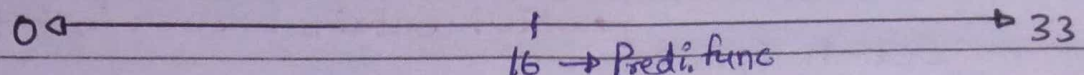
e = mid - 1

e = 34 - 1 = 33

Step I :- Search space choose

Step II :- Predicate function :- Check any condition and return T or F.

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

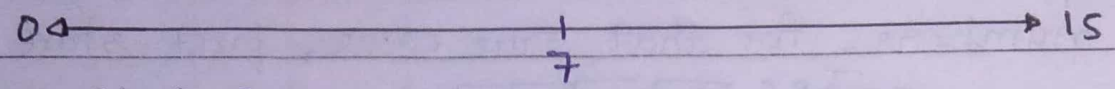


S=0, e=33, mid = 16

16 × 16 → 256 == 68 → F

256 > 68 → Left me jao → e = mid - 1

e = 16 - 1 = 15



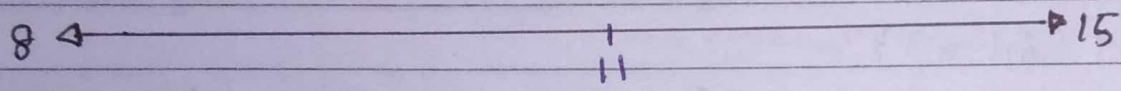
S=0, e=15, mid = $\frac{0+15}{2}$ = 7

7 × 7 = 49 == 68 → F

49 < 68 → Right me jao

Store ans ↙ ans = 7

s = mid + 1 = 7 + 1 = 8

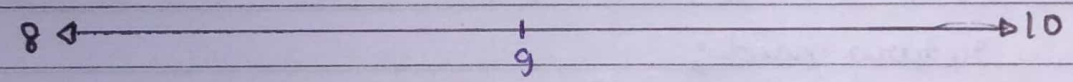


S=8, e = 15, mid = $\frac{8+15}{2} = \frac{23}{2}$ = 11

11 × 11 == 68 → F

121 > 68 → Left me jao → e = mid - 1

e = 11 - 1 = 10

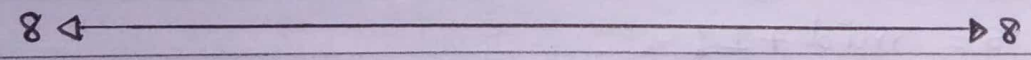


S=8, e=10, mid = 9

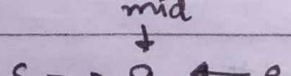
9 × 9 == 68 → F

81 > 68 → Left me jao → e = mid - 1

e = 9 - 1 = 8



S=8, e=8, mid=8



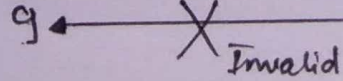
8 × 8 == 68 → F

8 × 8 > 68 → F

64 < 68 → store the ans in the case of '<' less than

ans = 8

Right me jao, s = mid = 1 ⇒ 8 + 1 = 9

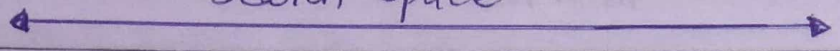


→ Rukjao

Spiral

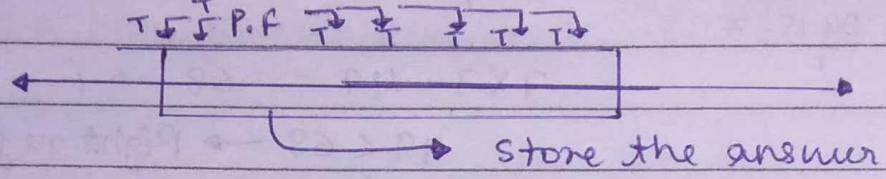
Date ..../..../.....

(A) Search Space → which contains answer in between



(B) Predicate function → To check any condⁿ and return True or false

(C) It may happen that, Predicate function return True for a range of numbers. for that true cases, first store the answer.



Code :-

```
int mySqrt (int x) {
    int s=0; int e=x ; int mid = s +(e - s)/2;
    int ans = -1;
    while (s <= e) {
    // Kya mid hi toh answer nahi
    if ( mid * mid == x)
            return mid;
    else if ( mid*mid < x )
            // store the ans
            //and move towards right
            ans = mid;
            s = mid +1;
    else
            //move towards Left
            e = mid - 1;
    //update mid
     mid = s + (e -s)/2;
    }
    return ans;
}
```

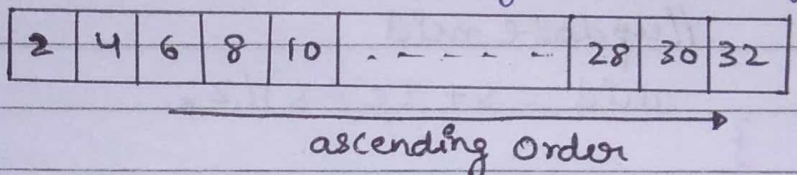
Date ..../..../.....

Binary Search in 2-D Array:-

	0	1	2	3
0	2	4	6	8
1	10	12	14	16
2	18	20	22	24
3	26	28	30	32

∴ (i) Every row is sorted in increasing order
(ii) Starting element of each row is greater than the last element of previous row.

→ It is stored in the memory as 1'D array.



↳ formula for 2D to 1D conversion — $(* i + j$
No. of Columns ← row index → col index

* ↳ formula for 1D to 2D conversion — $i = \frac{mid}{c}$, $j = mid \% c$
row Index ← No. of columns → column Index

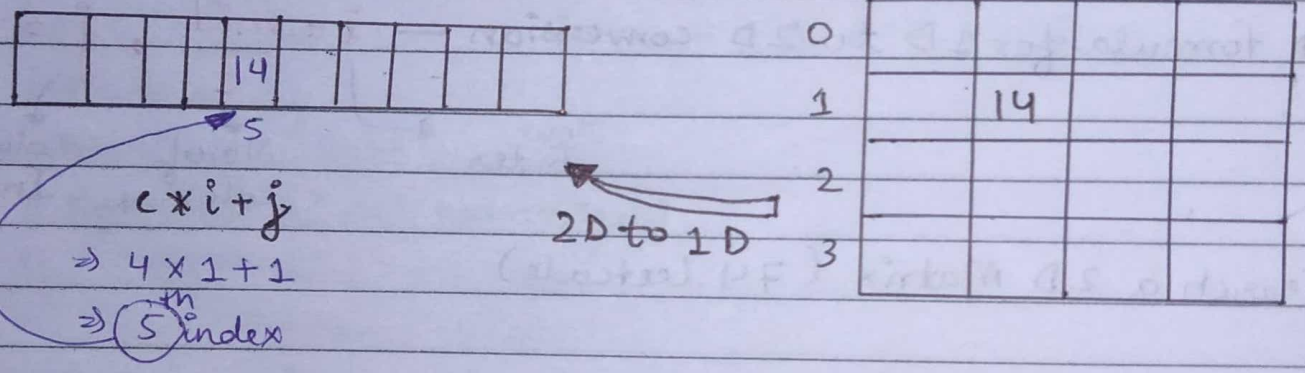
Q Search a 2D Matrix (74. Leetcode)

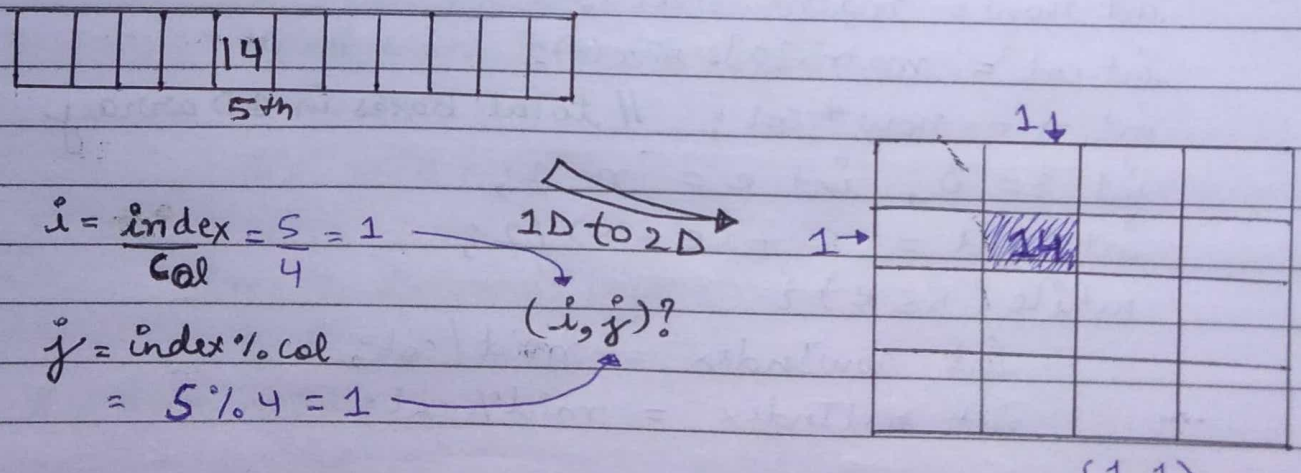
```
bool searchMatrix(vector<vector<int>>& matrix, int target) {
    int row = matrix.size();
    int col = matrix[0].size();
    int n = row * col;  // total boxes in 1D array
    int s = 0, int e = n-1;
    int mid = s + (e-s)/2;
    while(s<=e){
        int rowIndex = mid/col;
        int colIndex = mid%col;

        int currentNumber = matrix[rowIndex][colIndex]
```

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

```
if (currentNumber == target)
        return true;
else if (currentNumber > target)
        // Left
        e = mid - 1;
else
        // right
        s = mid + 1;
//update mid
 mid = s + (e - s)/2;
}
return false;
}
```

Total column = 4





Spiral