Searching & sorting - Level-2

Q. 1) Find Pivot Element

[Search in a rotated & rotated Array]

e.g → |12|14|16| 2|4| 6|8|10|

Approach 1
Take out Maximum number using Linear search. $TC \rightarrow O(N)$

Approach 2
sort
$T.C \rightarrow O(N \log N)$
then L.S or B.S

Approach 3
Separately handle pivot part.

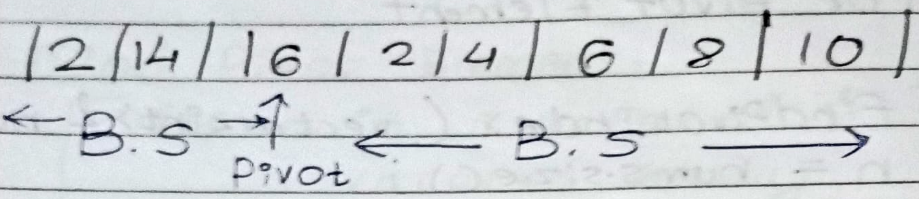
conditions ⇒
arr[mid] < arr[mid-1] → ①
arr[mid] > arr[mid+1] → ②

if arr[S] > arr[mid] → Ⓑ Part it is
↳ left maijao
else
↳ right maijao

Code for Pivot Element

```
~~int~~
 int   findPivotIndex ( vector<int>& nums){
   int n = nums.size();
    int s=0;
   int e= n-1;
    int  mid = s + (e-s)/2;

  while (s<=e){
     // corner case
     if(s<== e) {
         //single edement
       return s;
  }
  if (arr[mid] > arr[mid+ 1]) {
         return mid;
  }
  else if ( arr[mid]<arr[mid *-1])
     return mid-1;
  else if( arr [s] >arr[mid])
        e= mid-1;
  else
      s= mid+1;
  mid = s+ (e-s) /2;
 }
 return -1;
}
```



Q.2) Sqrt (x)

i/p → Number = x

O/p → $\sqrt{x}$ → ans

i/p → 25

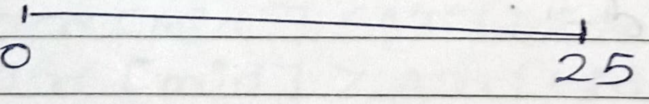
↓

$\sqrt{25}$

↓

5

① Search space



② Predigate function

③ ans store

H.W ⇒ Exact Root value.
upto 3 decimal places

Code:

```
int mySqrt (int x) {
    int s=0;
    int e=x;
  long long int mid= s+(e-s)/2;
   int ans=-1;
  while( s<=e) {
   //kya mid hi toh answer hai
   if(mid*mid ==x) {
      return mid;
  }
   else if ( mid*mid < x) {
       //ans store
      // right me jao
     ans=mid;
     s=mid+1;
  }
  else {
     //left me jana hai
     e=mid-1;
  }
  mid= s+(e-s)/2;
 }
 return ans;
}
```

Q3) Binary search on 2D Array.

Code:

```
bool searchMatrix (vector<vector<int>>
&matrix, int target) {
   int row = matrix.size();
   int col= matrix[0].size();
    int n = row * col;

   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 currNumber =matrix[rowIndex]
                         [colIndex];
  if (currNumber == target){
      return true;
  }
  else if (target >currNumber){
        //right
        s= mid+1;
  }
  else {
        // Left
        e=mid-1;
    }
   mid = s+(e-s)/2;
  }
} return false;
```

Tip:-

$$2D \xrightarrow[c * i + j]{} 1D$$

$$1D \longrightarrow 2D$$

$$i = \frac{index}{col}$$

$$j = index \% col$$