Binary Search to find X in sorted array

02 April 2025 10:18

```
public int search(int[] nums, int target)
    {
        int l=0;
        int r=nums.length-1;
        while(l<=r)
        {
            int mid=(l+r)/2;
            if(nums[mid]==target)
            {
                return mid;
            }
            else if(nums[mid]>target)
            {
                r=mid-1;
            }
            else
            {
                l=mid+1;
            }
        }
        return -1;

    }
```

Implement Lower Bound

02 April 2025 10:20

```
static int findFloor(long arr[], int n, long x)
    {
        Arrays.sort(arr);
        int l=0;
        int r=arr.length-1;
        int ans=-1;

        while(l<=r)
        {
            int mid=(l+r)/2;
            if(arr[mid]==x)
            {
                return mid;
            }
            else if(arr[mid]>x)
            {
                r=mid-1;
            }
            else
            {
                ans=mid;
                l=mid+1;
            }
        }
        return ans;
    }
```

Implement Upper Bound

02 April 2025 10:24

```
public int[] getFloorAndCeil(int x, int[] arr)
   {
      Arrays.sort(arr);
      int[]res=new int[2];
      int l=0;
      int r=arr.length-1;
      int ans=-1;

      while(l<=r)
      {
         int mid=(l+r)/2;
         if(arr[mid]==x)
         {
            ans=arr[mid];
            break;
         }
         else if(arr[mid]>x)
         {
            ans=arr[mid];
            r=mid-1;
         }
         else
         {

            l=mid+1;
         }
      }
      if(ans!=-1)
      {
         res[1]=ans;
      }
      else
      {
         res[1]=-1;
      }
      l=0;
      r=arr.length-1;
      ans=-1;

      while(l<=r)
      {
         int mid=(l+r)/2;
         if(arr[mid]==x)
         {
            ans=arr[mid];
            break;
         }
         else if(arr[mid]>x)
         {
            r=mid-1;
         }
         else
         {
            ans=arr[mid];
            l=mid+1;
         }
      }
      if(ans!=-1)
      {
```

```
        res[0]=ans;
    }
    else
    {
        res[0]=-1;
    }
    return res;
}
```

Search Insert Position

02 April 2025 10:26

```
public int searchInsert(int[] arr, int x)
{
int l=0;
int r=arr.length-1;
int ans=0;

while(l<=r)
{
int mid=(l+r)/2;
if(arr[mid]==x)
{

//ans=mid;
r=mid-1;
}
else if(arr[mid]>x)
{

r=mid-1;
}
else
{
//ans=mid;
l=mid+1;
}
}
return l;

}
```

Floor/Ceil in Sorted Array

02 April 2025 10:28

```
public int[] getFloorAndCeil(int x, int[] arr)
   {
      Arrays.sort(arr);
      int[]res=new int[2];
      int l=0;
      int r=arr.length-1;
      int ans=-1;

      while(l<=r)
      {
         int mid=(l+r)/2;
         if(arr[mid]==x)
         {
            ans=arr[mid];
            break;
         }
         else if(arr[mid]>x)
         {
            ans=arr[mid];
            r=mid-1;
         }
         else
         {

            l=mid+1;
         }
      }
      if(ans!=-1)
      {
         res[1]=ans;
      }
      else
      {
         res[1]=-1;
      }
      l=0;
      r=arr.length-1;
      ans=-1;

      while(l<=r)
      {
         int mid=(l+r)/2;
         if(arr[mid]==x)
         {
            ans=arr[mid];
            break;
         }
         else if(arr[mid]>x)
         {
            r=mid-1;
         }
         else
         {
            ans=arr[mid];
            l=mid+1;
         }
      }
      if(ans!=-1)
      {
```

```
        res[0]=ans;
    }
    else
    {
        res[0]=-1;
    }
    return res;
}
```

Find the first or last occurrence of a given number in a sorted array

02 April 2025 10:29

```
public int[] searchRange(int[] arr, int x)
{
Arrays.sort(arr);
int[]res=new int[2];
int l=0;
int r=arr.length-1;
int ans=-1;

while(l<=r)
{
int mid=(l+r)/2;
if(arr[mid]==x)
{
ans=mid;
l=mid+1;
}
else if(arr[mid]>x)
{

r=mid-1;
}
else
{

l=mid+1;
}
}
if(ans!=-1)
{
res[1]=ans;
}
else
{
res[1]=-1;
}
l=0;
r=arr.length-1;
ans=-1;

while(l<=r)
{
int mid=(l+r)/2;
if(arr[mid]==x)
{
ans=mid;
r=mid-1;
}
else if(arr[mid]>x)
{
r=mid-1;
```

```
}
else
{

l=mid+1;
}
}
if(ans!=-1)
{
res[0]=ans;
}
else
{
res[0]=-1;
}
return res;

}
```

From <https://onlinenotepad.org/notepad>

Count occurrences of a number in a sorted array with duplicates

02 April 2025 10:31

```
int count(int[] arr, int n, int x)
   {
      Arrays.sort(arr);
       int[]res=new int[2];
       int l=0;
       int r=arr.length-1;
       int ans=-1;

       while(l<=r)
       {
          int mid=(l+r)/2;
          if(arr[mid]==x)
          {
             ans=mid;
             l=mid+1;
          }
          else if(arr[mid]>x)
          {

             r=mid-1;
          }
          else
          {

             l=mid+1;
          }
       }
       if(ans!=-1)
       {
          res[1]=ans;
       }
       else
       {
          return 0;
       }
       l=0;
       r=arr.length-1;
       ans=-1;

       while(l<=r)
       {
          int mid=(l+r)/2;
          if(arr[mid]==x)
          {
             ans=mid;
             r=mid-1;
          }
          else if(arr[mid]>x)
          {
             r=mid-1;
          }
          else
          {
```

```
            l=mid+1;
        }
    }
    if(ans!=-1)
    {
        res[0]=ans;
    }
    else
    {
        return 0;
    }
    return res[1]-res[0]+1;
}
```

Search in Rotated Sorted Array I

02 April 2025 10:33

```
class Solution {
    public int search(int[] nums,int l,int r, int target)
    {

        while(l<=r)
        {
            int mid=(l+r)/2;
            if(nums[mid]==target)
            {
                return mid;
            }
            else if(nums[mid]>target)
            {
                r=mid-1;
            }
            else
            {
                l=mid+1;
            }
        }
        return -1;

    }
    public int search(int[] nums, int target)
    {
        int a=0;
        int l=0;
        int r=nums.length-1;
        if(nums.length==1)
        {
            if(nums[0]==target)
            {
                return 0;
            }
            return -1;
        }
        while(l<=r)
        {
            int mid=(l+r)/2;
            if(mid>0&&mid<nums.length-1)
            {
                if(nums[mid]<nums[mid-1]&&nums[mid]<nums[mid+1])
                {
                    a=mid;
                    break;
                }
                else if(nums[mid]>nums[0])
                {
                    l=mid+1;
                }
                else
                {
                    r=mid-1;
                }
            }
```

```
            else if(mid==0)
            {
                if(nums[0]>nums[1])
                {
                    a=1;
                    break;
                }
                else
                {
                    a=0;
                    break;
                }
            }
            else if(mid==nums.length-1)
            {
                if(nums[nums.length-1]<nums[nums.length-2])
                {
                    a=nums.length-1;
                    break;
                }
                else
                {
                    a=nums.length-2;
                    break;
                }
            }
        }
        int res1=search(nums,0,a-1,target);
        int res2=search(nums,a,nums.length-1,target);
        return Math.max(res1,res2);

    }
}
```

Search in Rotated Sorted Array II***

02 April 2025 10:46

[1,1,1,1,1,1,1,1,1,1,1,1,2,1,1,1,1,1]

```
public boolean search(int[] nums, int target)
    {
        int n=nums.length;
        int l=0;
        int h=n-1;
        while(l<=h)
        {
            int mid=(l+h)/2;
            if(nums[mid]==target)
            {
                return true;
            }
            if(nums[mid]==nums[l]&&nums[mid]==nums[h]) //Shrinking the cond
            {
                l++;
                h--;
                continue;
            }
            if(nums[l]<=nums[mid])
            {
                if(nums[l]<=target&&target<=nums[mid])
                {
                    h=mid-1;
                }
                else
                {
                    l=mid+1;
                }
            }
            else
            {
                if(nums[mid]<=target&&target<=nums[h])
                {
                    l=mid+1;
                }
                else
                {
                    h=mid-1;
                }
            }
        }
        return false;

    }
```

```cpp
class Solution {
public:


    int find(vector<int>nums)
    {
        int n = nums.size();
        int pivot = 0;
        for(int i = 0 ; i < n-1 ; i++)
        {
            // if(nums[i] == target)
            //     return true;
            if(nums[i+1] < nums[i])
            {
                pivot = i+1;
                return pivot;
            }
        }
        return 0;

    }


    int binary(vector<int>num,int l,int r,int k)
    {
        while(l<=r)
        {
            int mid=(l+r)/2;
            if(num[mid]==k)
            {
                return 1;
            }
            else if(num[mid]>k)
            {
                r=mid-1;
            }
            else
            {
                l=mid+1;
            }
        }
        return 0;
    }

    bool search(vector<int>& nums, int target)
    {
        int l=0;
        int r=nums.size()-1;
        int n=nums.size();
        if(nums[0]==target)
        {
            return true;
        }

        int mid=find(nums);

        return max(binary(nums,l,mid-1,target),binary(nums,mid,r,target));


    }
};
```

Find minimum in Rotated Sorted Array

02 April 2025 12:56

```
class Solution {
    public int findMin(int[] nums)
    {
        int a=0;
        int l=0;
        int r=nums.length-1;
        if(nums.length==1||nums[l]<nums[r])
        {
            return nums[0];
        }
        while(l<=r)
        {
            int mid=(l+r)/2;
            if(mid>0&&mid<nums.length-1)
            {
                if(nums[mid]<nums[mid-1]&&nums[mid]<nums[mid+1])
                {
                    a=mid;
                    break;
                }
                else if(nums[mid]>nums[0])
                {
                    l=mid+1;
                }
                else
                {
                    r=mid-1;
                }
            }
            else if(mid==0)
            {
                if(nums[0]>nums[1])
                {
                    a=1;
                    break;
                }
                else
                {
                    a=0;
                    break;
                }
            }
            else if(mid==nums.length-1)
            {
                if(nums[nums.length-1]<nums[nums.length-2])
                {
                    a=nums.length-1;
                    break;
                }
                else
                {
                    a=nums.length-2;
                    break;
                }
            }
```

```
        }

        return nums[a];

    }
}
```

Find out how many times has an array been rotated

02 April 2025 12:58

```
public int findKRotation(List<Integer> nums)
   {

       int a=0;
       int l=0;
       int r=nums.size()-1;
       if(nums.size()==1||nums.get(l)<nums.get(r))
       {
          return 0;
       }
while(l<=r)
       {
          int mid=(l+r)/2;
          if(mid>0&&mid<nums.size()-1)
          {
             if(nums.get(mid)<nums.get(mid-1)&&nums.get(mid)<nums.get(mid+1))
             {
                a=mid;
                break;
             }
             else if(nums.get(mid)>nums.get(0))
             {
                l=mid+1;
             }
             else
             {
                r=mid-1;
             }
          }
          else if(mid==0)
          {
             if(nums.get(0)>nums.get(1))
             {
                a=1;
                break;
             }
             else
             {
                a=0;
                break;
             }
          }
          else if(mid==nums.size()-1)
          {
             if(nums.get(nums.size()-1)<nums.get(nums.size()-2))
             {
                a=nums.size()-1;
                break;
             }
             else
             {
                a=nums.size()-2;
                break;
             }
          }
       }

       return a;

}
```

Single element in a Sorted Array

02 April 2025 12:59

```java
public int singleNonDuplicate(int[] nums)
{
    int high = nums.length-1;
    int low = 0;
    int mid;

    //Boundary cases
    if(high==0)
        return nums[0];
    else if(nums[0]!=nums[1])
        return nums[0];
    else if(nums[high]!=nums[high-1])
        return nums[high];
    while(low<=high)
    {
        mid = low + (high-low)/2;
        //Unique element condition
        if(nums[mid]!=nums[mid+1] && nums[mid]!=nums[mid-1])
            return nums[mid];

        if(((mid%2)==0 && nums[mid]==nums[mid+1])
           ||  ((mid%2)==1 && nums[mid]==nums[mid-1]))
            low = mid+1;
        else
            high = mid-1;

    }
    return -1;

}
```

```cpp
int singleNonDuplicate(vector<int>& nums)
    {
        int res=0;

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

    }
```

Find peak element

02 April 2025 13:17

```
public int findPeakElement(int[] nums)
    {
        int a=0;
        int l=0;
        int r=nums.length-1;
        if(nums.length==1)
        {

            return 0;
        }
        while(l<=r)
        {
            int mid=(l+r)/2;
            if(mid>0&&mid<nums.length-1)
            {
                if(nums[mid]>nums[mid-1]&&nums[mid]>nums[mid+1])
                {
                    a=mid;
                    break;
                }
                else if(nums[mid]>nums[mid-1])
                {
                    l=mid+1;
                }
                else
                {
                    r=mid-1;
                }
            }
            else if(mid==0)
            {
                if(nums[0]>nums[1])
                {
                    a=0;
                    break;
                }
                else
                {
                    a=1;
                    break;
                }
            }
            else if(mid==nums.length-1)
            {
                if(nums[nums.length-1]<nums[nums.length-2])
                {
                    a=nums.length-2;
                    break;
                }
                else
                {
                    a=nums.length-1;
                    break;
                }
            }
        }

        return a;


    }
```

Find square root of a number in log n

02 April 2025 13:22

```
int floorSqrt(int n) {
       // Your code here
        return (int) Math.floor(Math.sqrt(n));
   }
```

MY SOLN in logn:

```
int floorSqrt(int n) {
       // Your code here
        int low = 0;
       int high = n;
       int ans = -1;
       while (low<=high){
          int mid = low+(high-low)/2;
          int midSq = mid*mid;
          if (midSq==n){
             return mid;
          } else if (midSq < n) {
             low = mid+1;
             ans = mid;
          }else{
             high = mid-1;
          }
       }
       return ans;
   }
```

Find the Nth root of a number using binary search

02 April 2025 13:22

```
public int find(int n,int m,int mid)
  {
    long ans=1;
    for(int i=0;i<n;i++)
    {
      ans=ans*mid;
      if(ans>m)
      {
        return 2;
      }

    }

    if(ans==m)
    {
      return 1;
    }
    return 0;
  }

  public int nthRoot(int n, int m)
  {
    int l=1;
    int r=m;

    while(l<=r)
    {
      int mid=(l+r)/2;

      int ans=find(n,m,mid);
      if(ans==1)
      {
        return mid;
      }
      else if(ans==0)
      {
        l=mid+1;
      }
      else
      {
        r=mid-1;
      }
    }

    return -1;
  }
```

Koko Eating Bananas

02 April 2025 13:22

```
public int find(int[]piles, int x)
    {
        int a=0;
        for(int i=0;i<piles.length;i++)
        {
            a+=Math.ceil((double)(piles[i])/(double)(x));
        }
        return a;
    }
    public int minEatingSpeed(int[] piles, int h)
    {
        int a=1;
        int b=1;
        for(int i=0;i<piles.length;i++)
        {
            a=Math.min(a,piles[i]);
            b=Math.max(b,piles[i]);
        }
        while(a<=b)
        {
            int mid=(a+b)/2;
            int ans=find(piles,mid);
            if(ans<=h)
            {
                b=mid-1; //ye kyuki min find karna hai hame
            }
            else
            {
                a=mid+1;
            }
        }
        return a;

    }
```

Minimum days to make M bouquets

02 April 2025 13:23

```
public int find(int[]piles, int x ,int k)
    {
        int a=0;
        int maxc=0;
        for(int i=0;i<piles.length;i++)
        {
            if(piles[i]<=x)
            {
                a++;
                if(a==k)
                {
                    maxc++;
                    a=0;
                }
            }
            else
            {
                a=0;
            }
        }
        return maxc;
    }

    public int minDays(int[] bloomDay, int m, int k)
    {
        long val = (long) m * k;
        int n = bloomDay.length; // Size of the array
        if (val > n) return -1;
        int a = Integer.MAX_VALUE; // Minimum day
        int b = Integer.MIN_VALUE; // Maximum day
        for(int i=0;i<bloomDay.length;i++)
        {
            a=Math.min(a,bloomDay[i]);
            b=Math.max(b,bloomDay[i]);
        }
        while(a<=b)
        {
            int mid=(a+b)/2;
            int ans=find(bloomDay,mid,k);
            if(ans>=m)
            {
                b=mid-1; //ye kyuki min find karna hai hame
            }
            else
            {
                a=mid+1;
            }
        }
        return a;

    }
```

```
#include <iostream>
struct comp{
   int a=10;
   char b;
   float c;
   //double t;
   int arr[0]; //
   static int var;
};
int main() {
   // Write C++ code here
   std::cout << sizeof(comp)<<" ";
   comp cp;
   std::cout<<cp.a<<" ";
   std::cout<<sizeof(cp)<<" ";

   return 0;
}
```

```
// Online C++ compiler to run C++ program online
#include <iostream>
#include<cstring>

int main() {
   // Write C++ code here
   int a=0;
   char arr[10];

   std::cin>>arr;
   if(strncmp(arr,"passwd",6)==0)
   {
      a=1;
   }
   if(a)
   {
      std::cout<<"access granted";
   }
   else
   {
      std::cout<<"access denied";
   }

   return 0;
}
```

Find the smallest Divisor***

02 April 2025 13:23

```
public int find(int[]piles, int x)
    {

        int maxc=0;
        for(int i=0;i<piles.length;i++)
        {
            maxc+=Math.ceil((double)piles[i]/(double)(x));

        }
        //System.out.println(maxc);
        return maxc;
    }
    public int smallestDivisor(int[] nums, int threshold)
    {
        //long val = (long) m * k;
        int n = nums.length; // Size of the array
        //if (val > n) return -1;
        int ans=-1;
        int a = 1;//Integer.MAX_VALUE; // initialization with 1
        int b = Integer.MIN_VALUE; // Maximum day
        for(int i=0;i<nums.length;i++)
        {
            a=Math.min(a,nums[i]);
            b=Math.max(b,nums[i]);
        }
        while(a<=b)
        {
            int mid=(a+b)/2;
            int ans1=find(nums,mid);
            //System.out.println(ans1);
            if(ans1==threshold)
            {
                ans=mid;
                b=mid-1; //threshold hai ya usse kam to badhane ke liye denominator
ekam karna padega na re baba
            }
            else if(ans1>threshold)
            {
                a=mid+1;

            }
            else
            {
                ans=mid;
                b=mid-1;
            }

        }
        return ans;
    }
```

Conclusion:

By initializing a to 1, we ensure that we are considering **all possible divisors**, from the smallest (which produces the largest sum) to the largest (which will produce the smallest sum). Starting with the minimum element in nums would skip potential valid solutions and fail to find the correct smallest divisor.

Capacity to Ship Packages within D Days

02 April 2025 13:23

```
public int find(int[]piles, int x)
    {
        int a=0;
        int maxc=1;
        for(int i=0;i<piles.length;i++)
        {

            if(a+piles[i]>x)
            {
                maxc++;
                a=piles[i];

            }
            else
            {
                a+=piles[i];
            }
        }
        return maxc;
    }

    public int shipWithinDays(int[] weights, int days)
    {
        //long val = (long) m * k;
        int n = weights.length; // Size of the array
        //if (val > n) return -1;

           int a = 0; // Minimum possible capacity is the largest weight in the array
           int b = 0; // Maximum possible capacity is the sum of all weights
        int res=-1;
        for(int i=0;i<weights.length;i++)
        {
            a=Math.max(a,weights[i]);
            b+=weights[i];
        }
        while(a<=b)
        {
            int mid=(a+b)/2;
            int ans=find(weights,mid);
            if(ans>days)
            {
                a=mid+1; //ye kyuki min find karna hai hame
            }
            else
            {
                res=mid;
                b=mid-1;
            }
        }
        return res;
    }
```

Kth Missing Positive Number

02 April 2025 13:23

Aggressive Cows

02 April 2025 13:23

```
public static boolean canWePlace(int[] stalls, int dist, int cows) {
      int n = stalls.length; //size of array
      int cntCows = 1; //no. of cows placed
      int last = stalls[0]; //position of last placed cow.
      for (int i = 1; i < n; i++) {
         if (stalls[i] - last >= dist) {
            cntCows++; //place next cow.
            last = stalls[i]; //update the last location.
         }
         if (cntCows >= cows) return true;
      }
      return false;
   }
   public static int aggressiveCows(int[] stalls, int k) {
      int n = stalls.length; //size of array
      //sort the stalls[]:
      Arrays.sort(stalls);

      int low = 1, high = stalls[n - 1] - stalls[0];
      //apply binary search:
      while (low <= high) {
         int mid = (low + high) / 2;
         if (canWePlace(stalls, mid, k) == true) {
            low = mid + 1;
         } else high = mid - 1;
      }
      return high;
   }
```

Split array - Largest Sum

02 April 2025 13:23

Painter's partition

02 April 2025 13:23

```
public static int countStudents(ArrayList<Integer> arr, int pages) {
    int n = arr.size(); // size of array
    int students = 1;
    long pagesStudent = 0;
    for (int i = 0; i < n; i++) {
      if (pagesStudent + arr.get(i) <= pages) {
        // add pages to current student
        pagesStudent += arr.get(i);
      } else {
        // add pages to next student
        students++;
        pagesStudent = arr.get(i);
      }
    }
    return students;
  }

  public static int findPages(ArrayList<Integer> arr, int n, int m) {
    // book allocation impossible
    if (m > n)
      return -1;

    int low = Collections.max(arr);
    int high = arr.stream().mapToInt(Integer::intValue).sum();
    while (low <= high) {
      int mid = (low + high) / 2;
      int students = countStudents(arr, mid);
      if (students > m) {
        low = mid + 1;
      } else {
        high = mid - 1;
      }
    }
    return low;
  }
```

0jlnbr	

Minimize Max Distance to Gas Station

02 April 2025 13:23

Median of 2 sorted arrays

02 April 2025 13:24

Book Allocation Problem

02 April 2025 13:23

```
public static int countStudents(ArrayList<Integer> arr, int pages) {
 int n = arr.size(); // size of array
 int students = 1;
 long pagesStudent = 0;
 for (int i = 0; i < n; i++) {
 if (pagesStudent + arr.get(i) <= pages) {
 // add pages to current student
 pagesStudent += arr.get(i);
 } else {
 // add pages to next student
 students++;
 pagesStudent = arr.get(i);
 }
 }
 return students;
 }

 public static int findPages(ArrayList<Integer> arr, int n, int m) {
 // book allocation impossible
 if (m > n)
 return -1;

 int low = Collections.max(arr);
 int high = arr.stream().mapToInt(Integer::intValue).sum();
 while (low <= high) {
 int mid = (low + high) / 2;
 int students = countStudents(arr, mid);
 if (students > m) {
 low = mid + 1;
 } else {
 high = mid - 1;
 }
 }
 return low;
 }
```

Kth element of 2 sorted arrays

02 April 2025 13:24

Find the row with maximum number of 1's

11 April 2025 01:25

```
public static int lowerBound(ArrayList<Integer> arr, int n, int x) {
      int low = 0, high = n - 1;
      int ans = n;

      while (low <= high) {
         int mid = (low + high) / 2;
         // maybe an answer
         if (arr.get(mid) >= x) {
            ans = mid;
            // look for smaller index on the left
            high = mid - 1;
         } else {
            low = mid + 1; // look on the right
         }
      }
      return ans;
   }
   public static int rowWithMax1s(ArrayList<ArrayList<Integer>> matrix, int n, int m) {
      int cnt_max = 0;
      int index = -1;

      // traverse the rows:
      for (int i = 0; i < n; i++) {
         // get the number of 1's:
         int cnt_ones = m - lowerBound(matrix.get(i), m, 1);
         if (cnt_ones > cnt_max) {
            cnt_max = cnt_ones;
            index = i;
         }
      }
      return index;
   }
```

Search in a 2 D matrix

11 April 2025 01:26

```
public boolean searchMatrix(int[][] matrix, int target)
    {
        int i=0;
        int j=matrix[0].length-1;
        while(i<=matrix.length-1&&j>=0)
        {
            if(matrix[i][j]==target)
            {
                return true;
            }
            else if(matrix[i][j]>target)
            {
                j--;
            }
            else
            {
                i++;
            }
        }
        return false;

    }
```

Search in a row and column wise sorted matrix

11 April 2025 01:26

```
public boolean searchMatrix(int[][] matrix, int target)
  {
     int i=0;
     int j=matrix[0].length-1;
     while(i<=matrix.length-1&&j>=0)
     {
        if(matrix[i][j]==target)
        {
           return true;
        }
        else if(matrix[i][j]>target)
        {
           j--;
        }
        else
        {
           i++;
        }
     }
     return false;

  }
```

Find Peak Element (2D Matrix)

11 April 2025 01:26

Matrix Median

11 April 2025 01:26