

1.in a company, those empoloyee salary has more than10000. They will get 2000 bonus otherwise will get 1000 bonus.

1.start

2.enter salary

3. if(salary>10000)

Output “Salary+20000”

Else{

Output”salary+10000”}

4.exit

If (salary>10000)

Enter the value of salary

(Yes)

Salary+ 2000

(no)

Salary +1000

Output Salary

2. Check the number prime or not

Num%c==0

C<num

Input num, c=2

=

C=c+1

no

Not prime number

yes Yes

Prime number

no

Suppose 36

1\*36=36(Simply we take sqrt for looing input

2\*18=36

3\*12=36 only this part requed for looping

4\*9=36

6\*6=36

9\*4=36

12\*3=36 repeated step (not equired)

18\*2=36

36\*1=36

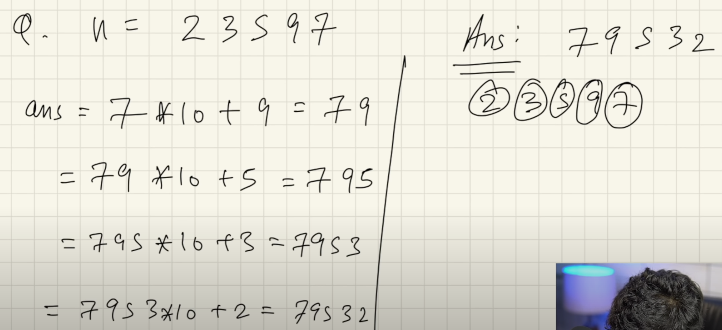
**Pseudo code**

1. **Start**
2. **Input the value of num and declar c=2;**
3. **While c<num**

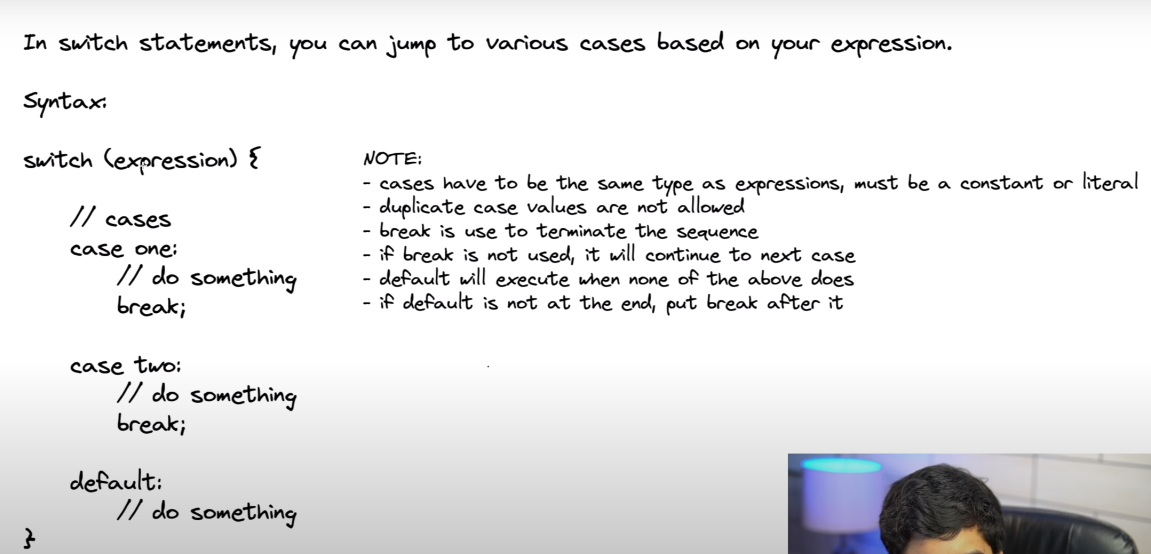
* **If (num%c==0){**
* **Print (it’s not a prime number)**
* **Beak;**
* **}**
* **Else{ (c=c+1)**

**4.end while loop**

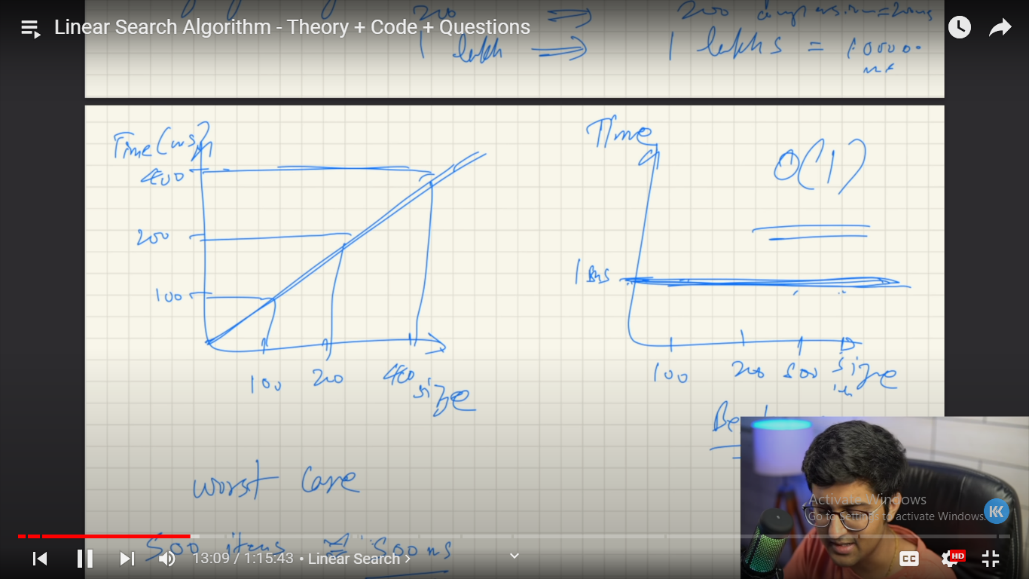
1. **Print it’s a prime number 5. exit**
2. **Reverse Number**



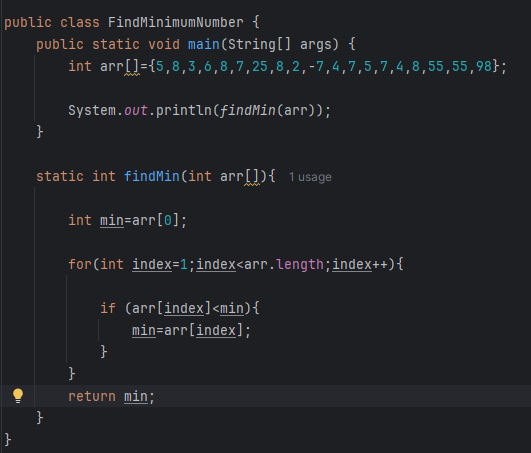
1. Switch Case



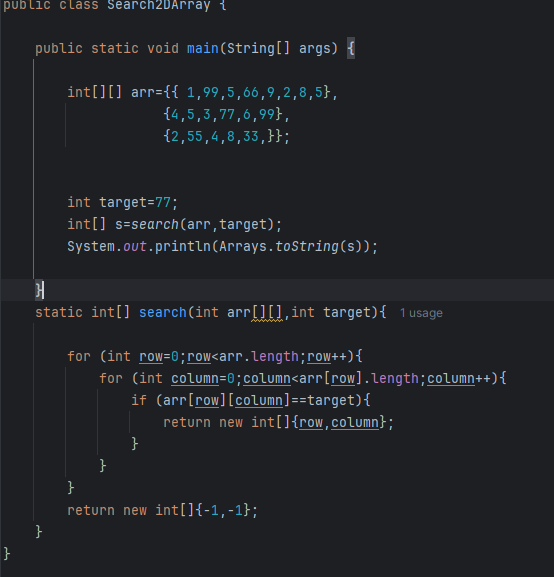
**1.LINEAR\_SEARCH\_ALGORITHM**



Q.Find the minimum number in array



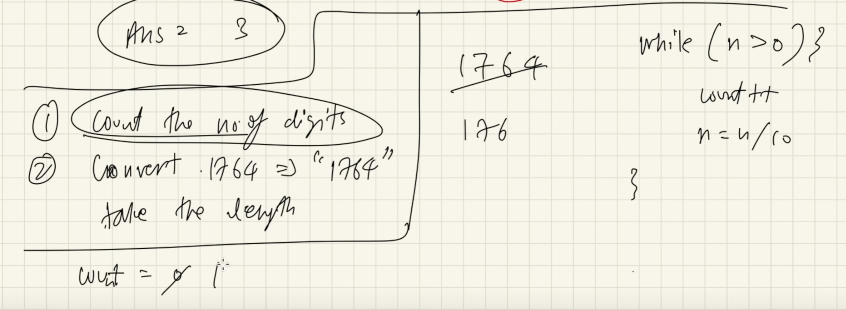
Q.2D Array find the number



**1.**

**Leet Code :**

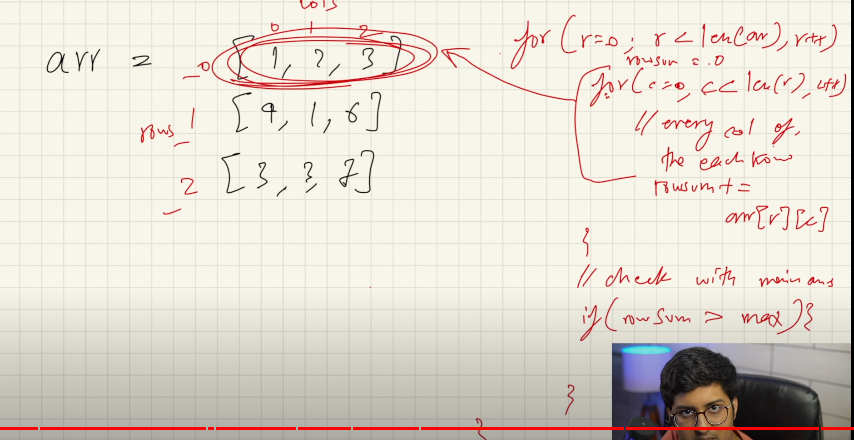
[**https://leetcode.com/problems/find-numbers-with-even-number-of-digits/submissions/1285192492/**](https://leetcode.com/problems/find-numbers-with-even-number-of-digits/submissions/1285192492/)



public class FindEvenNumDigit {  
 public static void main(String[] args) {  
  
 int arr[]={1425,25792,73145,25,84,8,8888,99,3336,78962};  
  
 int evenCount=*evenFind*(arr);  
  
 System.*out*.println(evenCount);  
  
 }  
  
 static int evenFind(int arr[])  
 {  
 int evenNum=0;  
 for (int i=0;i<arr.length;i++){  
 int k=arr[i];  
 if(*even*(k)){  
 evenNum++;  
 }  
 }  
  
 return evenNum;  
 }  
  
  
 static boolean even(int k){  
 int count=0;  
 while (k>0){  
 k=k/10;  
 count++;  
 }  
  
 if (count%2==0){  
 return true;  
 }  
 else {  
 return false;  
 }  
 }  
}

[**1672. Richest Customer Wealth**](https://leetcode.com/problems/richest-customer-wealth/)

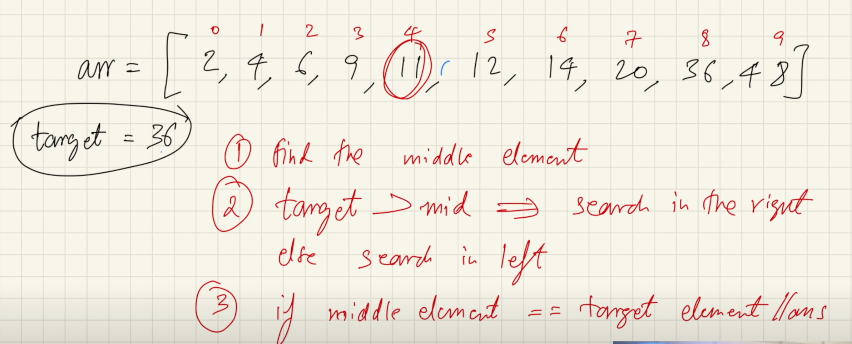
<https://leetcode.com/problems/richest-customer-wealth/>



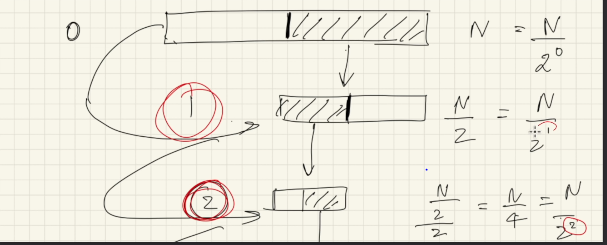
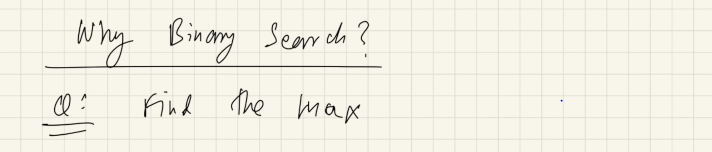
public class RichestCustomerWealth {  
  
 public static void main(String[] args) {  
 int[][] arr={  
  
 {2,8,7},{7,1,3},{1,9,5}  
 };  
  
 int total=*maxValue*(arr);  
  
 System.*out*.println(total);  
 }  
  
static int maxValue(int[][] arr){  
   
int max=0;  
  
 for (int row=0;row<arr.length;row++){  
   
 int sum=0;  
   
 for (int col=0;col<arr[row].length;col++){  
  
 sum=sum+arr[row][col];  
  
 }  
 if(sum>max){  
   
 max=sum;  
 }  
 }  
 return max;   
 }  
}

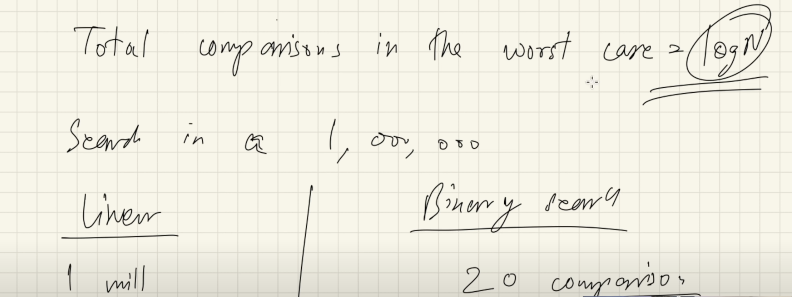
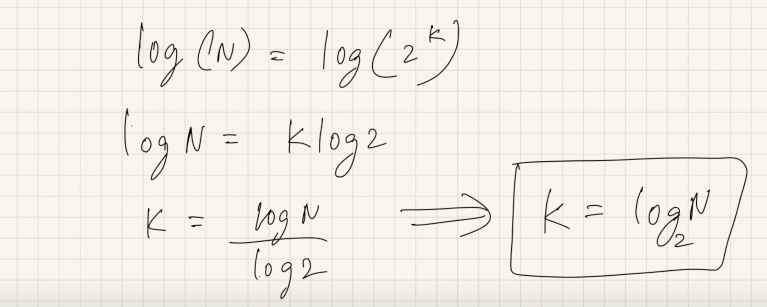
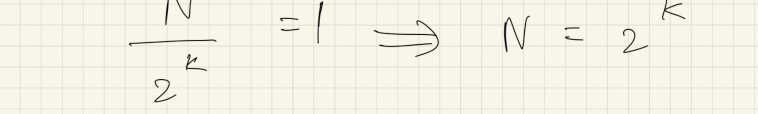
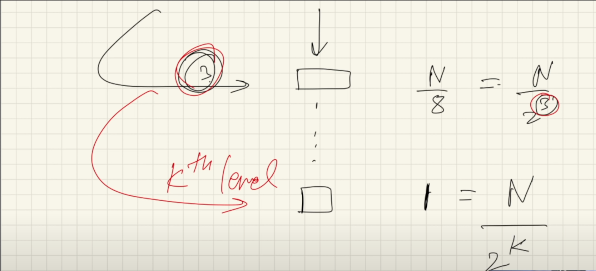
**2.BINARY SEARCH ALGORITHM**

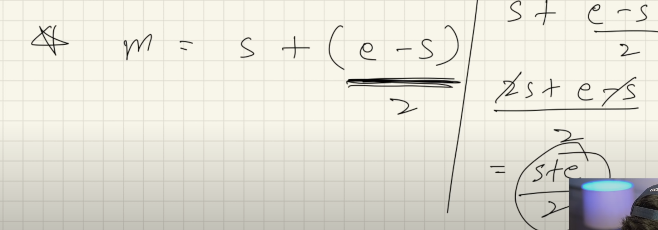
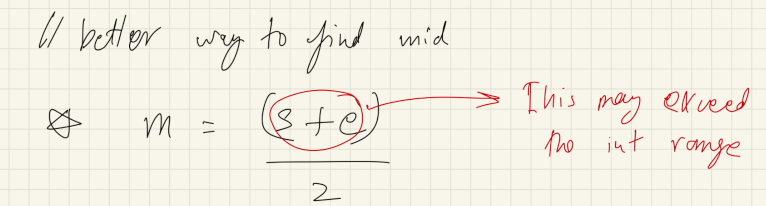
STEPS WE FOLLOWED











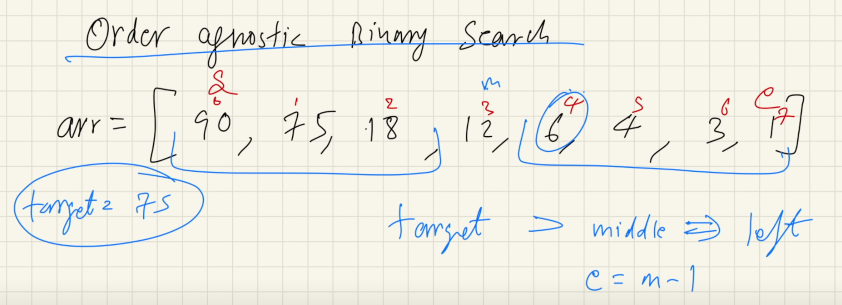
**1.find the number using binary Search Algorithm**

package BinarySearchAlgo;  
  
public class SearchTheNumber {  
  
 public static void main(String[] args) {  
  
 int[] arr={2,3,4,5,6,7,8,9,10,11,12,13};  
  
 int target=9;  
 int num= *binarySearch*(arr,target);  
 System.*out*.println(num);  
 }  
  
 static int binarySearch(int[] arr,int target){  
 int start=0;  
 int end=arr.length-1;  
  
 while(start<=end){  
  
 int mid=start+(end-start)/2;  
 if(target<arr[mid]){  
 end=mid-1;  
 } else if (target>arr[mid]) {  
 start=mid+1;  
 }  
 else {  
 return mid;  
 }  
 }  
  
 return -1;  
 }  
}

**2.find the number using binary Search Algorithm from unsort array**

package BinarySearchAlgo;  
  
public class SearchNumberfromUnSortArray {  
  
 public static void main(String[] args) {  
 int[] arr={2,4,89,5,6,7,55,8,99,15,78,35,10};  
 int[] rr= *sorting*(arr);  
  
 int target=10;  
 for (int i=0;i < rr.length;i++){  
 System.*out*.println(rr[i]);  
 }  
  
 int num=*binarySearch*(rr,target);  
  
 System.*out*.println("Searching index number "+num);  
  
  
 }  
 static int[] sorting( int[] arr){  
  
 int t=0;  
 for (int i=0;i< arr.length-1;i++){  
 for(int j=i+1;j<arr.length;j++){  
  
 if (arr[i]>arr[j]){  
 t=arr[j];  
 arr[j]=arr[i];  
 arr[i]=t;  
 }  
  
 }  
 }  
 return arr;  
 }  
  
  
 static int binarySearch(int[] rr,int target){  
  
 int start=0;  
 int end=rr.length-1;  
  
 int mid=start+(end-start)/2;  
  
 while(start<=end){  
  
 if(target<rr[mid]){  
 end=mid-1;  
 }else if(rr[mid]>target) {  
 start=mid+1;  
 }  
 else {  
 return mid;  
 }  
 }  
  
 return -1;  
 }  
  
  
}

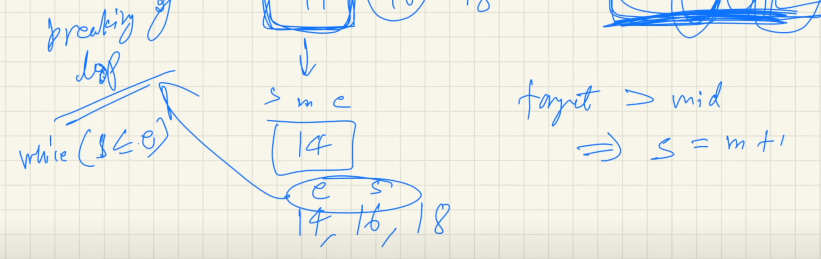
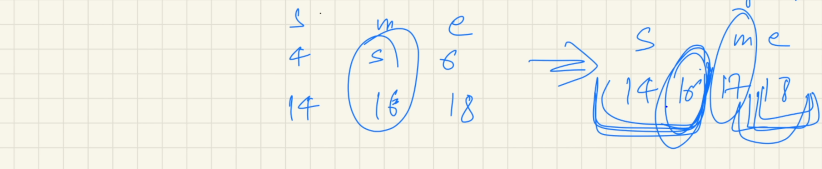
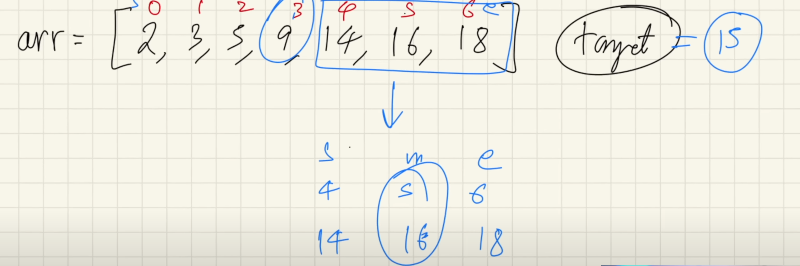
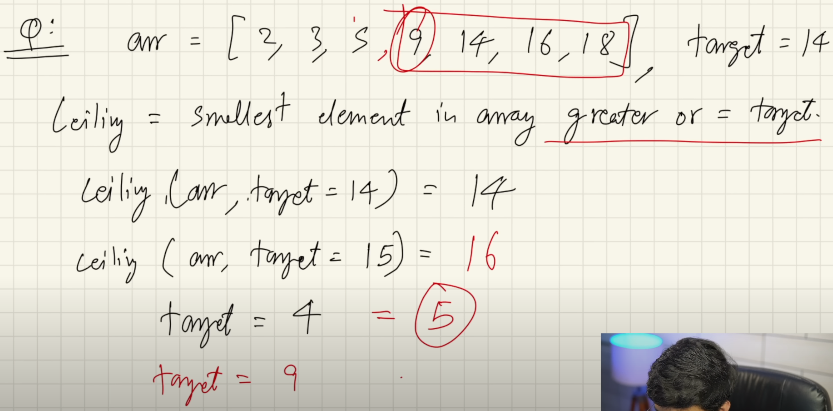
**Descending order number Search using BinarySearchAlgo**

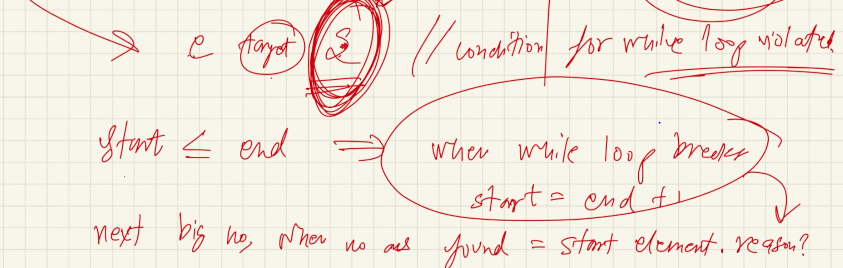
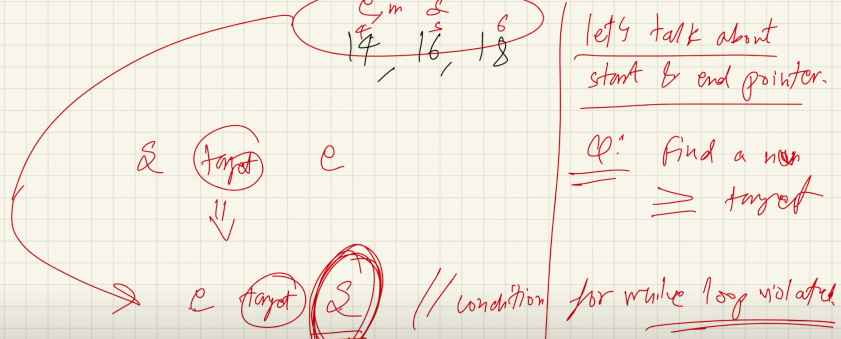


**Q.BinarySearchAlgo for OrderIgnosticsArray**

package BinarySearchAlgo;  
  
public class OderIgnosticBSOptimizedCode {  
  
 public static void main(String[] args) {  
 int[] arr = {40, 34, 32, 29, 28, 24, 23, 22, 20, 18, 15, 12, 11, 10, 9, 5, 3, 2};  
 int target = 3;  
 int num = *binarySearch*(arr, target);  
 System.*out*.println(num);  
 }  
 static int binarySearch(int[] arr,int target){  
   
 int start=0;  
 int end=arr.length-1;  
 boolean isAsyc=arr[start]<arr[end];  
   
 while(start<=end){  
   
 int mid=start+(end-start)/2;  
 if (target == arr[mid]) {  
 return mid;  
 }  
  
 if (isAsyc){  
   
 if (target<arr[mid]){  
   
 end=mid-1;  
   
 }else {  
   
 start=mid+1;  
 }  
 }else{  
   
 if(target<arr[mid]){  
   
 start=mid+1;  
   
 }else {  
   
 end=mid-1;  
 }  
 }  
 }  
 return -1;  
 }  
}

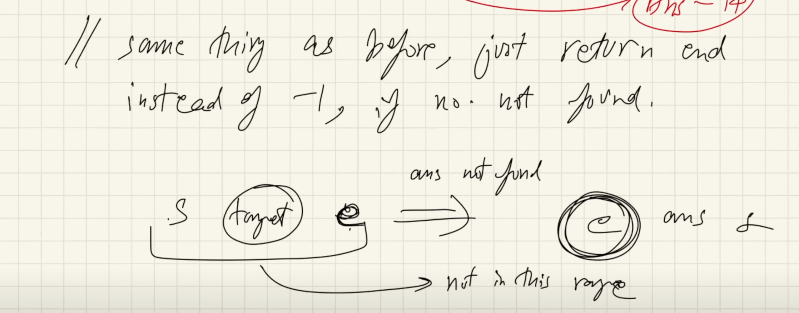
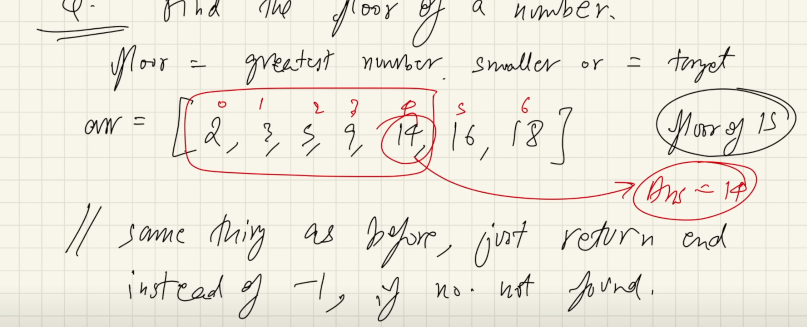
**Q.find the ceilng number of target element**





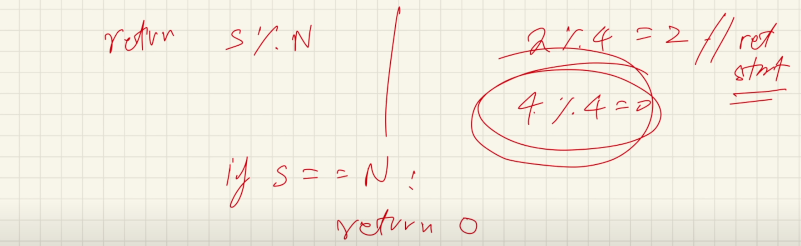
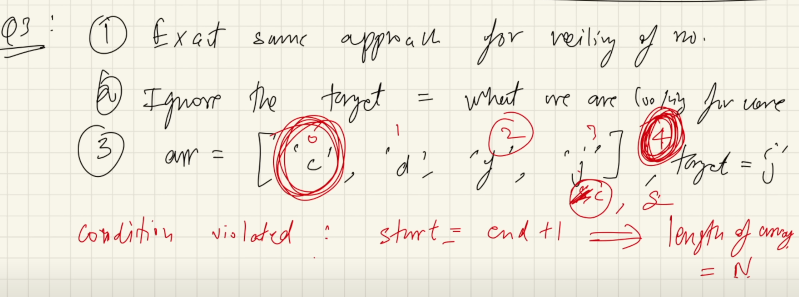
package BinarySearchAlgo.interviewQuestions;  
  
public class CeilingOfNumberUsingBS2 {  
  
 public static void main(String[] args) {  
  
 int[] arr={1,10,11,12,13,15,16,17};  
 int target=14;  
 int ceilingNum=*ceilingNum*(arr,target);  
  
 System.*out*.println(ceilingNum);  
 }  
  
   
 static int ceilingNum(int[] arr,int target){  
  
 int start=0;  
 int end=arr.length-1;  
  
 while(start<=end){  
  
 int mid=start+(end-start)/2;  
  
 if(target>arr[mid]){  
 start=mid+1;  
 } else if (target<arr[mid]) {  
 end=mid-1;  
  
 }else{  
 return mid;  
 }  
 }  
 return start;  
 }  
  
}

1. **FIND THE FLOOR NUMBER OF TARGET NUMBER USING BS ALGO**

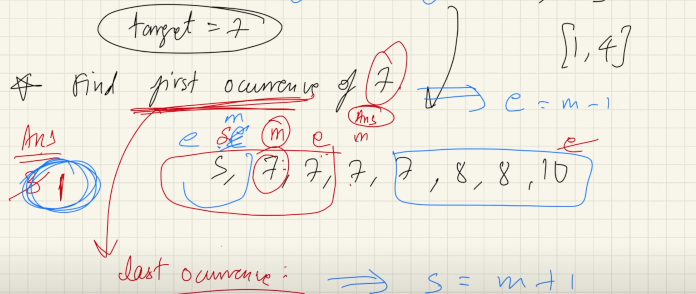


package BinarySearchAlgo.interviewQuestions;  
  
public class FloorOfTargetNumberUsingBS {  
  
 public static void main(String[] args) {  
  
 int[] arr={1,10,11,12,13,15,16,17};  
  
 int target=14;  
 int floorNum=*floorNumBS*(arr,target);  
 System.*out*.println(floorNum);  
 }  
  
 static int floorNumBS(int[] arr,int target){  
  
 int start=0;  
 int end=arr.length-1;  
  
 while (start<=end){  
  
 int mid=start+(end-start)/2;  
 if (target<arr[mid]){  
 end=mid-1;  
  
 } else if (target>arr[mid]) {  
 start=mid+1;  
  
 }else {  
 return mid;  
 }  
 }  
 return arr[end];  
 }  
}

**LeetCode:** [**https://leetcode.com/problems/find-smallest-letter-greater-than-target/description/**](https://leetcode.com/problems/find-smallest-letter-greater-than-target/description/)

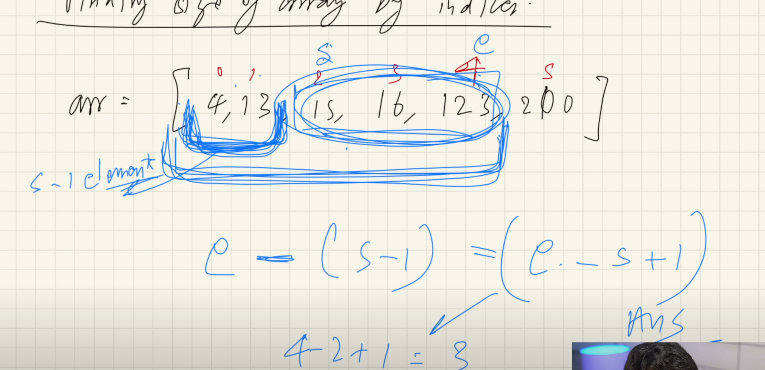
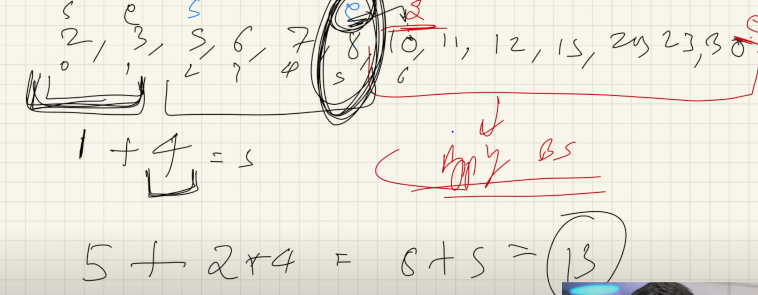
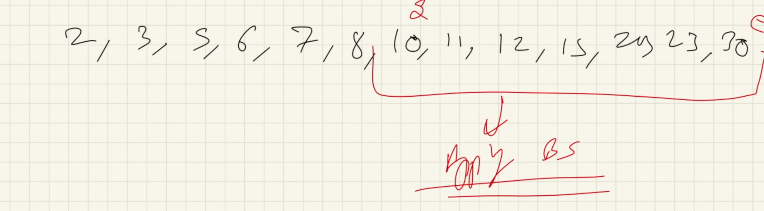
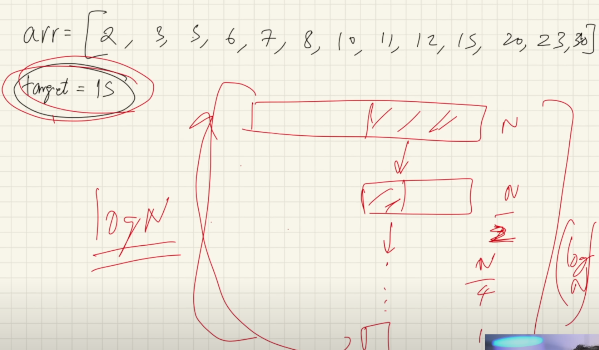


package BinarySearchAlgo.LeetCode;  
//https://leetcode.com/problems/find-smallest-letter-greater-than-target/  
public class FindSmallestLetterGreaterThanTarget {  
  
 public static void main(String[] args) {  
  
 char[] letters = {'c','f','j'}; char target = 'a';  
 char c=*nextGreatestLetter*(letters,target);  
  
 System.*out*.println(c);  
  
 }  
 static char nextGreatestLetter(char[] letters, char target) {  
  
 int start=0;  
 int end=letters.length-1;  
  
 while(start<=end){  
  
 int mid=start+(end-start)/2;  
  
 if (target<letters[mid]){  
 end=mid-1;  
 } else {  
 start=mid+1;  
 }  
 }  
  
 return letters[start % letters.length];  
 }  
}

1. **LeetCode:** [**https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/**](https://leetcode.com/problems/find-first-and-last-position-of-element-in-sorted-array/)
2. class Solution {
3. public int[] searchRange(int[] nums, int target) {
4. int start=0;
5. int end=nums.length-1;
6. int[] krr={-1,-1};
7. krr[0]=firstIndex(nums,target,true);
8. krr[1]=firstIndex(nums,target,false);
9. return krr;
10. }
12. public int firstIndex(int[] arr,int target,boolean firstIndex){
14. int start=0;
15. int end=arr.length-1;
16. int ans=-1;
17. while(start<=end){
18. int mid=start+(end-start)/2;
19. if(target<arr[mid]){
20. end=mid-1;
21. }else if(target>arr[mid]){
22. start=mid+1;
23. }else{
24. ans=mid;
25. if(firstIndex){
26. end=mid-1;
28. }else{
29. start=mid+1;
31. }
32. }
33. }
34. return ans;
35. }
36. }

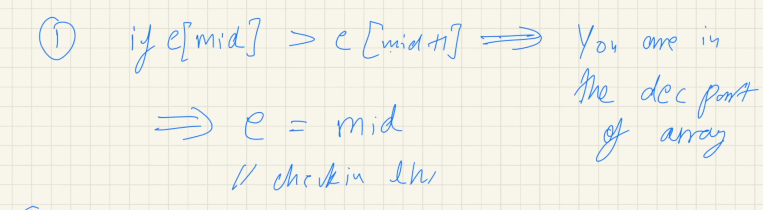
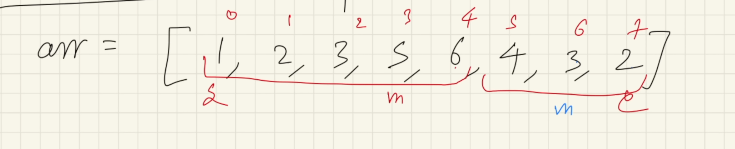
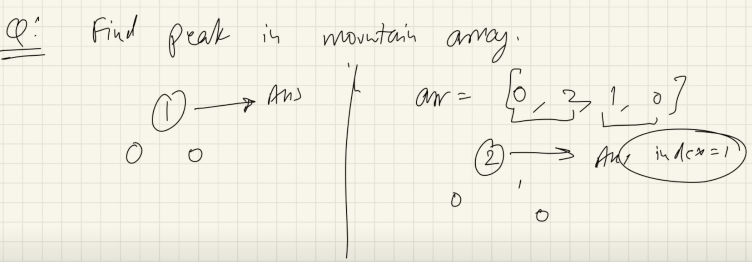
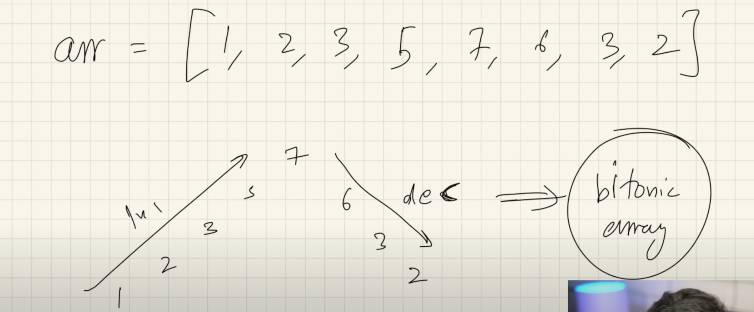
package BinarySearchAlgo.LeetCode;  
  
public class FindFirstAndLastPositionOfElementInSortedArray {  
 public static void main(String[] args) {  
  
 int[] nums = {5, 7, 7, 8, 8, 10};  
  
 int target = 8;  
 int[] krr = *searchRange*( nums, target);  
  
 System.*out*.println(krr[0]+" "+krr[1]);  
  
 }  
 public static int[] searchRange(int[] nums, int target){  
  
 int[] krr={-1,-1};  
 krr[0]=*firstndexSearch*(nums,target,true);  
 krr[1]=*firstndexSearch*(nums,target,false);  
  
 return krr;  
 }  
  
 static int firstndexSearch(int[] nums, int target, boolean b) {  
  
 int start=0;  
 int end=nums.length-1;  
 int ans=-1;  
 while(start<=end){  
 int mid=start+(end-start)/2;  
  
 if (target<nums[mid]){  
 end=mid-1;  
 } else if (target>nums[mid]) {  
 start=mid+1;  
  
 }else{  
 ans=mid;  
 if(b){  
 end=mid-1;  
 }else {  
 start=mid+1;  
 }  
 }  
 }  
  
 return ans;  
 }  
}

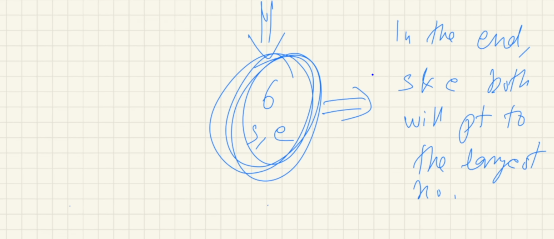
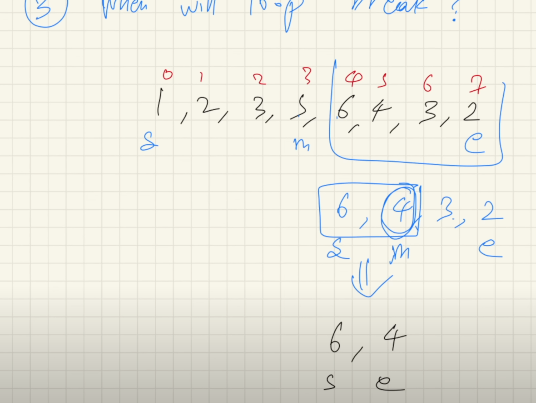
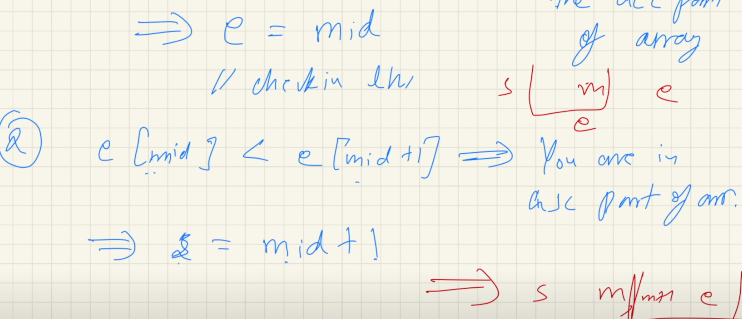
**Question: Position of element in infinite sorted array**



package BinarySearchAlgo.AmazonInterviewQuestions;  
  
public class PositionOfElementInInfiniteSortedArray {  
  
 public static void main(String[] args) {  
 int[] arr={12,54,59,60,63,69,74,75,76,79,89,90,95,96,98,100,101,103,106};  
 int target =75;  
 System.*out*.println(*ans*(arr,target));  
 }  
 static int ans(int[] arr,int target){  
  
 int start=0;  
 int end=1;  
  
 while(target>arr[end]){  
  
 int temp=end+1;  
 end=end+(end-start+1)\*2;  
 start=temp;  
 }  
int ansIndex=*binarySearch*(arr,target,start,end);  
 return ansIndex;  
 }  
  
 static int binarySearch(int[] arr,int target,int start,int end){  
  
 while(start<=end){  
 int mid=start+(end-start)/2;  
 if(target<arr[mid]){  
 end=mid-1;  
 }else if(target>arr[mid]){  
 start=mid+1;  
 }else {  
 return mid;  
 }  
 }  
 return -1;  
 }  
}

1. **LeetCode** [**https://leetcode.com/problems/peak-index-in-a-mountain-array/description/**](https://leetcode.com/problems/peak-index-in-a-mountain-array/description/)





**Code(leetcode IDE)**

class Solution {

    public int peakIndexInMountainArray(int[] arr) {

        int start=0;

        int end=arr.length-1;

        while(start<end){

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

            if(arr[mid]>arr[mid+1]){

                end=mid;

            }else if(arr[mid]<arr[mid+1]){

                start=mid+1;

            }

        }

        return start;

    }

}

package BinarySearchAlgo.LeetCode;  
//https://leetcode.com/problems/peak-index-in-a-mountain-array/description/  
public class PeakIndexInMountainArray {  
  
 public static void main(String[] args) {  
 int[] arr={1,2,3,4,4,5,6,7,8,5,6,3,2,1};  
 System.*out*.println(*peakIndex*(arr));  
 }  
 static int peakIndex(int[] arr){  
  
 int start=0;  
 int end=arr.length-1;  
  
 while (start<end){  
 int mid=start+(end-start)/2;  
 if(arr[mid]<arr[mid+1]){  
 start=mid+1;  
 } else if (arr[mid]>arr[mid+1]) {  
  
 end=mid;  
  
 }  
 }  
 return start;  
 }  
}

1. **LeetCode:** [**Find Peak Element**](https://leetcode.com/problems/find-peak-element/)

class Solution {

    public int findPeakElement(int[] nums) {

       int start=0;

       int end=nums.length-1;

       while(start<end){

        int mid=start+(end-start)/2;

        if(nums[mid]>nums[mid+1]){

            end=mid;

        }else if(nums[mid]<nums[mid+1]){

            start=mid+1;

        }

       }

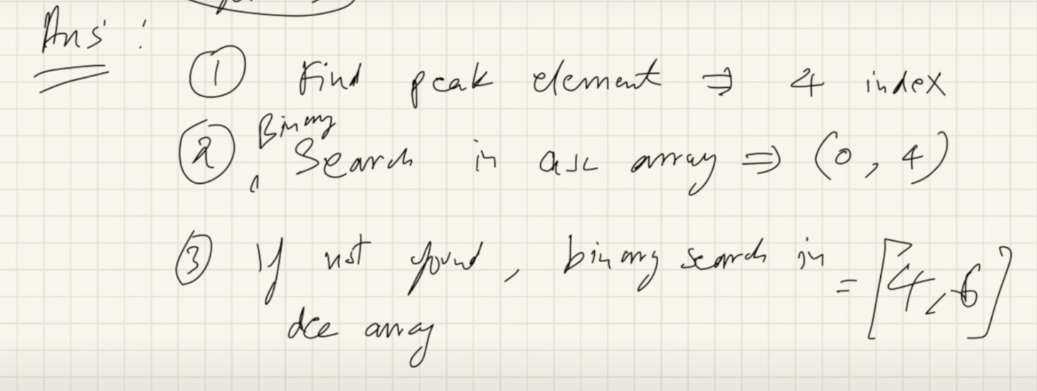
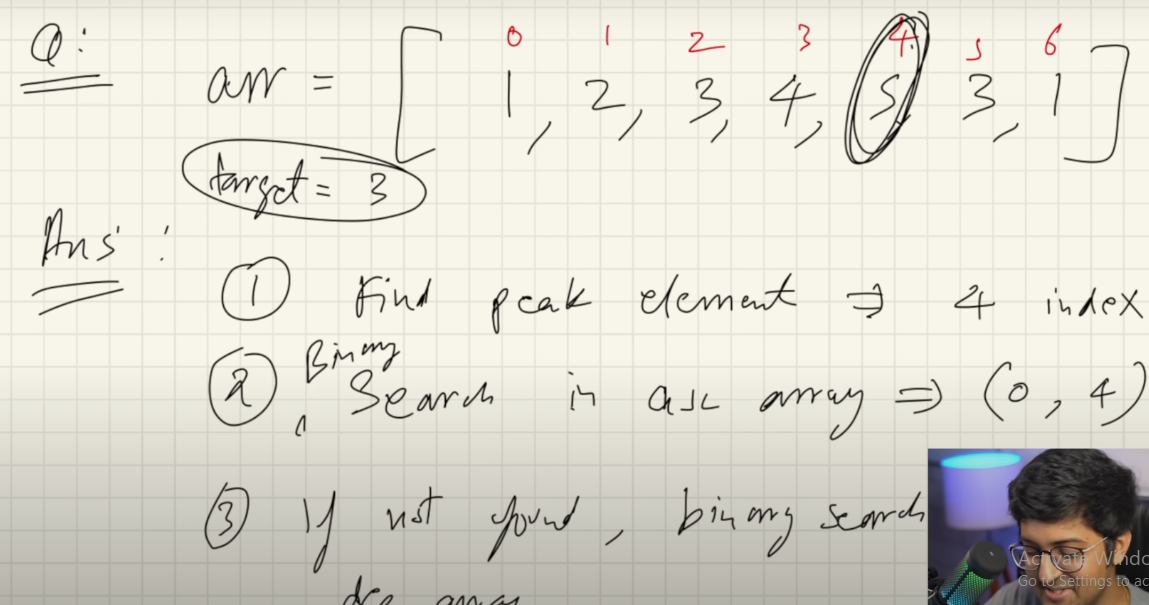
       return start;

    }

}

package BinarySearchAlgo.LeetCode;  
  
//https://leetcode.com/problems/find-peak-element/description/   
public class FindPeakElement {  
  
 public static void main(String[] args) {  
 int[] arr={1,2,1,3,5,6,4};  
  
 int ans=*findPeakElement*(arr);  
 System.*out*.println(ans);  
 }  
 static int findPeakElement(int[] arr){  
  
 int start=0;  
 int end=arr.length-1;  
  
 while(start<end){  
  
 int mid=start+(end-start)/2;  
  
 if (arr[mid]<arr[mid+1]){  
 start=mid+1;  
 } else if (arr[mid]>arr[mid+1]) {  
 end=mid;  
 }  
 }  
 return start;  
 }  
}

1. **LeetCode** [**https://leetcode.com/problems/find-in-mountain-array/**](https://leetcode.com/problems/find-in-mountain-array/)



class Solution {

    public int findInMountainArray(int target, MountainArray mountainArr) {

        int start=0;

        int end=mountainArr.length()-1;

       int peak=peakMountain(mountainArr,start,end);

       int order=orderAscending(mountainArr,start,peak, target);

        if(order!=-1){

           return order;

        }else{

            int dOrder=orderDescending(mountainArr,peak,end,target);

            return dOrder;

        }

    }

    public int peakMountain(MountainArray mountainArr, int start,int end ){

        while(start<end){

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

            if(mountainArr.get(mid)<mountainArr.get(mid+1)){

                start=mid+1;

            }else{

                end=mid;

            }

        }

        return start;

    }

    public int orderAscending(MountainArray mountainArr, int start,int peak,int target){

        int end=peak;

        while(start<=end){

        int mid=start+(end-start)/2;

            if(target<mountainArr.get(mid)){

               end=mid-1;

            }else if(target>mountainArr.get(mid)){

              start=mid+1;

            }

            else{

             return mid;

            }

        }

       return -1;

    }

    public int orderDescending(MountainArray mountainArr,int peak, int end,int target){

        int start=peak;

        while(start<=end){

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

            if(target<mountainArr.get(mid)){

                start=mid+1;

            }else if(target>mountainArr.get(mid)){

                end=mid-1;

            }else{

                return mid;

            }

        }

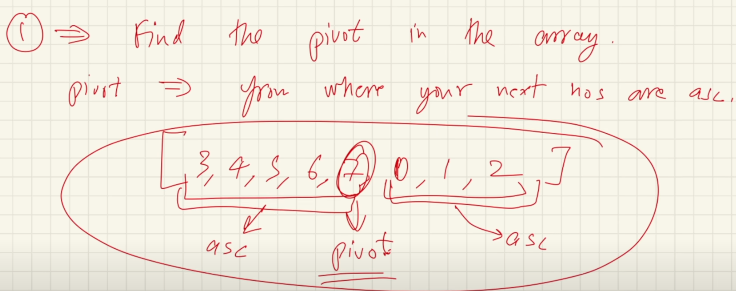
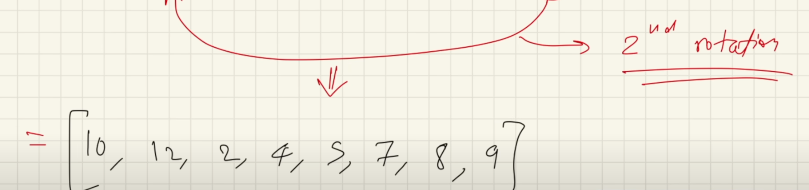
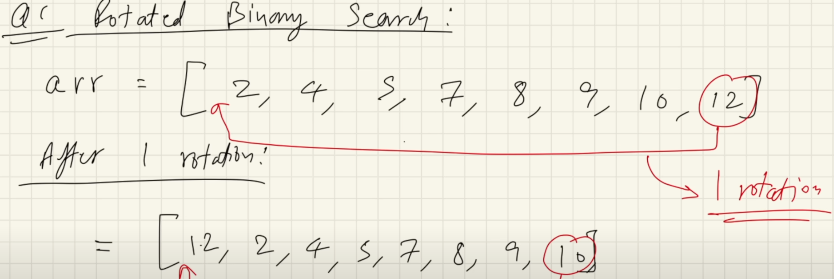
        return -1;

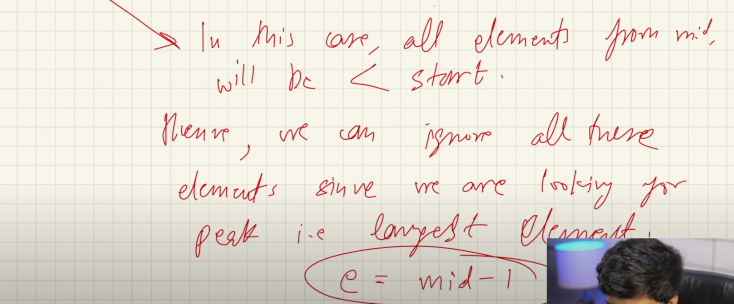
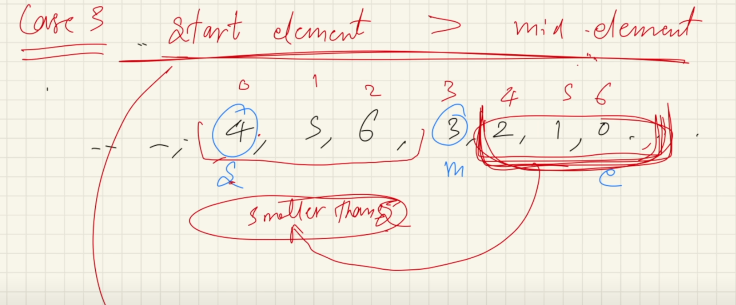
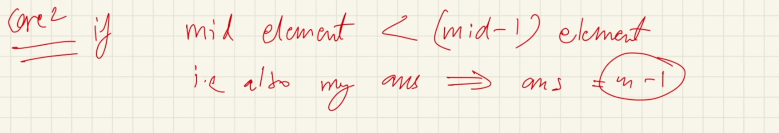
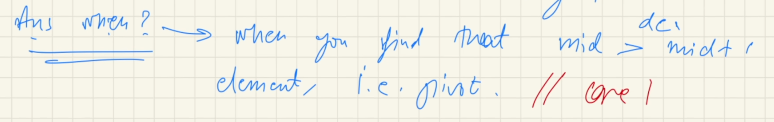
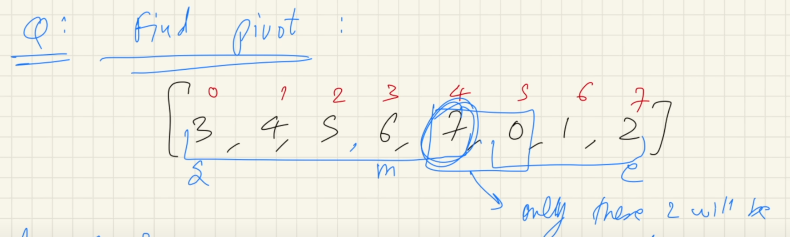
    }

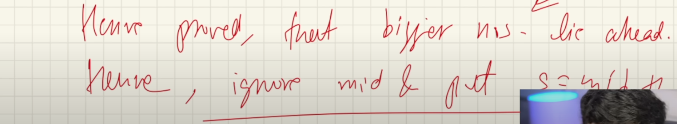
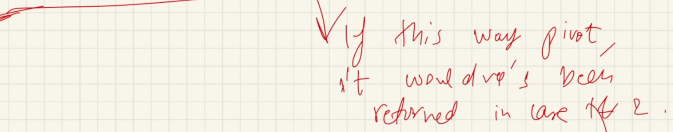
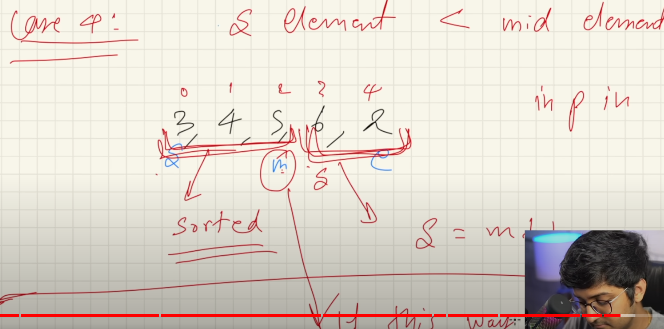
}

package BinarySearchAlgo.LeetCode;  
  
//https://leetcode.com/problems/find-in-mountain-array/  
//  
  
public class FindInMountainArray {  
 public static void main(String[] args) {  
  
 int[] arr={1,2,3,4,5,6,7,3,1};  
 int target=3;  
  
 int peak= *peakIndex*(arr,target);  
  
 int ascOrder = *ascendingBS*(arr,peak,target);//peak use as end index value  
 if (ascOrder!=-1){  
 System.*out*.println(ascOrder);  
 }else {  
 int desOrder = *descendingBS*(arr, peak, target); //peak use as start index value  
 System.*out*.println(desOrder);  
  
 }  
 }  
  
 static int peakIndex(int[] arr, int target){  
  
 int start=0;  
 int end=arr.length-1;  
  
 while(start<end) {  
 int mid = start + (end - start) / 2;  
 if (arr[mid] < arr[mid + 1]) {  
 start = mid + 1;  
 } else if (arr[mid] > arr[mid + 1]) {  
 end = mid;  
 }  
  
 }  
 return start;  
 }  
  
 static int ascendingBS(int[] arr, int end,int target){  
 int start=0;  
  
 while (start<=end){  
  
 int mid=start+(end-start)/2;  
 if (target<arr[mid]){  
 end=mid-1;  
 } else if (target>arr[mid]) {  
 start=mid+1;  
 }else {  
 return mid;  
 }  
 }  
 return -1;  
 }  
  
 static int descendingBS(int[] arr,int start,int target){  
  
 int end=arr.length-1;  
 while(start<=end){  
 int mid=start+(end-start)/2;  
 if (target<arr[mid]){  
 start=mid+1;  
 } else if (target>arr[mid]) {  
 end=mid-1;  
 }else {  
 return mid;  
 }  
  
 }  
 return -1;  
 }  
}

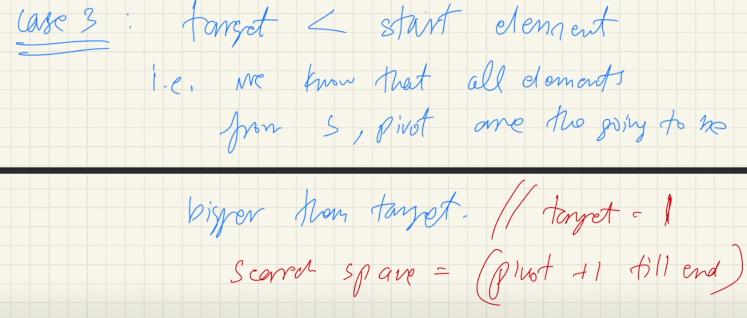
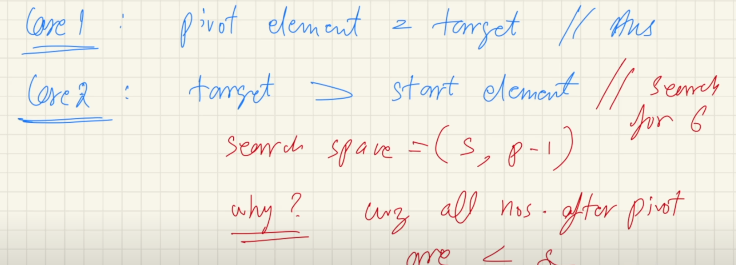
1. [**Search in Rotated Sorted Array**](https://leetcode.com/problems/search-in-rotated-sorted-array/)







->After find pivot



class Solution {

    public int search(int[] nums, int target) {

        int start=0;

        int end=nums.length-1;

        int findPeak=findiPivort(nums,start,end);

          if (findPeak == -1) {

            return binarySearch(nums, start, end, target);

        }

         if(target==nums[findPeak]){

            return findPeak;

         }

        if(target>=nums[0]){

        int fHalf=binarySearch(nums,start,findPeak-1,target); //findpeak use as a end

        return fHalf;

        }

        else{

            int lHalf=binarySearch(nums,findPeak+1,end, target); //findpeak use as a start

           return lHalf;

        }

    }

    public int findiPivort(int[] arr,int start,int end){

        while(start<=end){

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

            if(mid<end && arr[mid]>arr[mid+1]){

                return mid;

            }else if(mid>start && arr[mid]<arr[mid-1]){

                return mid-1;

            } else if(arr[mid]<=arr[start]){

                end=mid-1;

            }else{

                start=mid+1;

            }

        }

        return -1;

    }

    public int binarySearch(int[] arr,int start,int end, int target){

          while(start<=end){

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

            if(arr[mid]<target){

                start=mid+1;

            }else if(arr[mid]>target){

                end=mid-1;

            }else{

                return mid;

            }

        }

        return -1;

    }

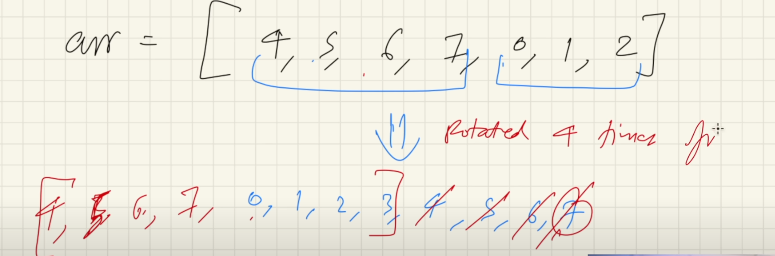
}

public class SearchInRotateArray {  
  
 public static void main(String[] args) {  
  
 int[] arr={4,5,6,7,0,1,2};  
 int target=0;  
 int start=0;  
 int end=arr.length-1;  
 int pivot=*findPivot*(arr,start,end);  
  
 if(pivot==-1){  
 int binarySearc= *binarySearch*(arr,start,end,target);  
 System.*out*.println(binarySearc);  
 } if(arr[start]==target) {  
 System.*out*.println(start);  
  
 } if (arr[start]<=target) {  
 System.*out*.println(*binarySearch*(arr,start,pivot-1,target));  
 }else {  
 System.*out*.println(*binarySearch*(arr,pivot+1,end,target));  
 }  
 }  
  
 static int findPivot(int[] arr,int start,int end){  
  
 while(start<=end){  
   
 int mid=start+(end-start)/2;  
 if(arr[mid]>arr[mid+1]){  
 return mid;  
 } else if (arr[mid-1]>arr[mid]) {  
 return mid-1;  
 } else if (arr[mid]<=arr[start]) {  
 end=mid-1;  
 }else{  
 start=mid+1;  
 }  
 }  
 return -1;  
 }  
  
 static int binarySearch(int[] arr,int start,int end,int target){  
  
 while(start<=end){  
 int mid=start+(end-start)/2;  
  
 if(arr[mid]<target){  
 start=mid+1;  
 }else if (arr[mid]>target){  
 end=mid-1;  
 }else {  
 return mid;  
 }  
 }  
  
 return -1;  
 }  
  
}

1. **Search Repeat Arrtay rotation Array**

package BinarySearchAlgo.LeetCode;  
  
public class SearchRepeatArrtayrotationArray {  
  
 public static void main(String[] args) {  
 int[] arr={2,2,9,2,2,2};  
 int target=2;  
  
 System.*out*.println(*findPivot*(arr));  
  
 }  
 static int findPivot(int[] arr){  
  
 int start=0;  
 int end=arr.length-1;  
  
 while(start<=end){  
  
 int mid=start+(end-start)/2;  
 if (arr[mid]>arr[mid+1]){  
 return mid;  
 } else if (arr[mid]<arr[mid-1]) {  
 return mid-1;  
  
 } else if (arr[mid]==arr[start] &&arr[mid]==arr[end] ) {  
 if (arr[start]>arr[start+1]){  
 return start;  
 }  
 start++;  
 if (arr[end]<arr[end-1]){  
 return end-1;  
 }  
 end--;  
  
 } else if (arr[start]<arr[mid]|| (arr[start]==arr[mid] && arr[mid]>arr[end])) {  
 start=mid+1;  
 } else if (arr[start]>arr[mid]||(arr[start]==arr[mid] && arr[mid]<arr[end])) {  
 end=mid-1;  
 }  
 }  
  
  
 return -1;  
 }  
}

1. **Find the Rotation Count in Rotated Sorted array**



package BinarySearchAlgo;  
  
public class RotationCountArray  
{  
 public static void main(String[] args) {  
 int[] arr={5,6,7,8,9,10,1,2,3,4};  
  
 int rotation=*rotationCount*(arr);  
 System.*out*.println(rotation+1);  
 }  
  
 static int rotationCount(int[] arr){  
 int start=0;  
 int end=arr.length-1;  
  
 while(start<=end){  
 int mid=start+(end-start)/2;  
  
 if(arr[mid]>arr[mid+1]){  
 return mid;  
 } else if (arr[mid-1]>arr[mid]) {  
  
 return mid-1;  
 } else if (arr[start]<arr[mid]) {  
 start=mid+1;  
 } else if (arr[start]>=arr[mid]) {  
 end=mid-1;  
 }  
 }  
  
 return -1;  
 }  
}

1. [**Split Array Largest Sum**](https://leetcode.com/problems/split-array-largest-sum/)

