Binary search

Interview problems

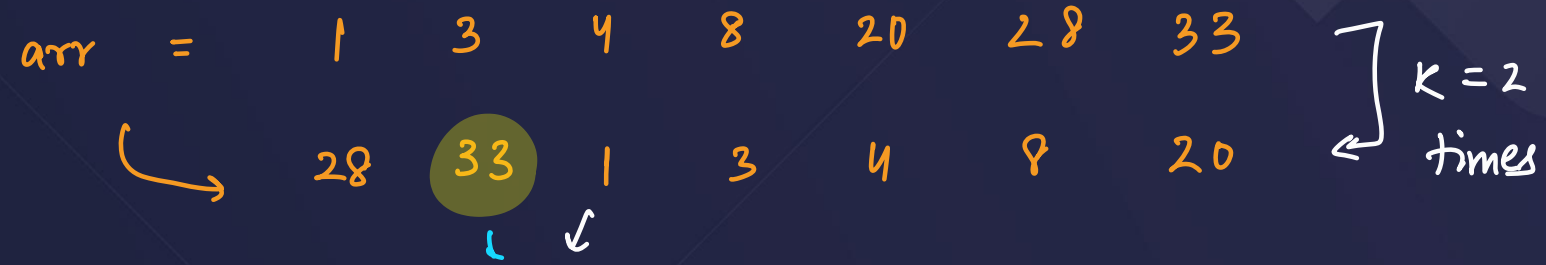
Ques : Peak index in mountain array [Leetcode 852]

O(log n)

arr[i-1] < arr[i] > arr[i+1]

```
mid = lo + (hi-lo)/2 ;
if (arr[mid] > arr[mid+1] && arr[mid] < arr[mid-1])
    return mid;
else if( arr[mid] > arr[mid+1]) hi = mid-1;
else lo = mid+1
```

Ques : Search in Rotated Sorted Array [Leetcode 33]



Task no. 1 → pivot element / idx →

```
if (arr[mid] < arr[mid+1] &&
        arr[mid] > arr[mid-1])
```

Ques : Search in Rotated Sorted Array [Leetcode 33]

0	1	2	3	4	5	6	7
6	8	20	28	33	1	3	4
				lo	mid		hi

if (arr[mid] < arr[mid-1] && arr[mid] < arr[mid+1])

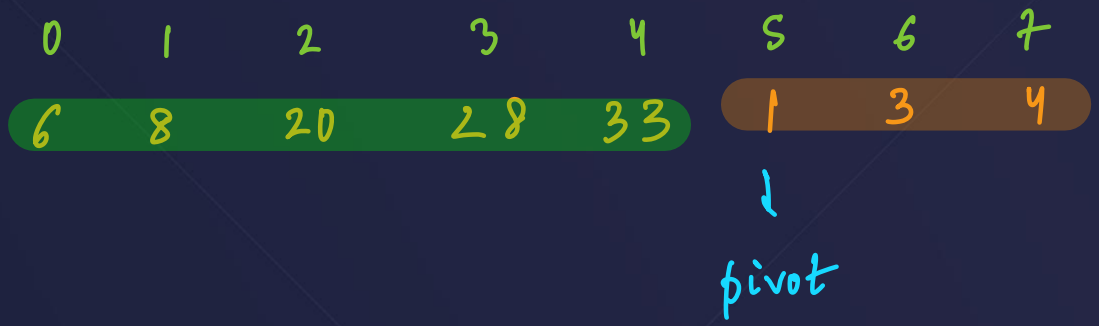
if (arr[mid] > arr[mid-1] && arr[mid] > arr[mid+1]

if (arr[mid] > arr[hi]) lo = mid+1

if (arr[mid] < arr[hi]) hi = mid-1

Ques : Search in Rotated Sorted Array [Leetcode 33]

→ O(log n)



target = 20

if (target >= nums[0] && target <= nums[pivot-1])

B.S.

```
// 4 5 6 7 0 1 2
int n = nums.size();
int lo = 0;
int hi = n-1;
// finding pivot element / index
int pivot = -1; // smallest element
while(lo<=hi){
    int mid = lo + (hi-lo)/2;
    if(nums[mid]<nums[mid+1] && nums[mid]<nums[mid-1]){
        pivot = mid;
        break;
    }
    else if(nums[mid]>nums[mid+1] && nums[mid]>nums[mid-1]){
        pivot = mid + 1;
        break;
    }
    else if(nums[mid]>nums[hi]) lo = mid + 1;
    else hi = mid - 1;
}
```

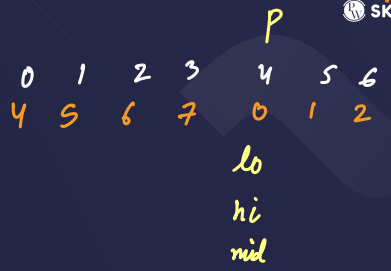
n = 1

1
lo
hi
mid

pivot = -1

```
if(target>=nums[0] && target<=nums[pivot-1]){
    lo = 0;
    hi = pivot-1;
    // normal binary search
    while(lo<=hi){
        int mid = lo + (hi-lo)/2;
        if(nums[mid]==target) return mid;
        else if(nums[mid]>target) hi = mid - 1;
        else lo = mid + 1;
    }
}
else{
    lo = pivot;
    hi = n-1;
    // normal binary search
    while(lo<=hi){
        int mid = lo + (hi-lo)/2;
        if(nums[mid]==target) return mid;
        else if(nums[mid]>target) hi = mid - 1;
        else lo = mid + 1;
    }
}
```

target = 0



arr = 1 3

hi

lo

mid

target = 0

If array was already sorted, then you wont find the pivot element

if (mid == 0) lo = mid + 1

if (mid == n-1) hi = mid - 1

arr 3 1

target = 1

hi

lo

mid

Ques : Find K Closest Elements [Leetcode 658]

Case-1 → x is present

arr 1 2 3 4 6

lb ↑ (2)

ub ↑ (4)

x = 3

k = 4

vector<int> v(k); → 3 2 4 1

↳ sort

[1 2 3 4]

Ques : Find K Closest Elements [Leetcode 658]

Case - 2 If 'x' is not present in array

arr = 1 2 3 4 6

lb ↑ (3)

ub ↑ (6)

x = 5

K = 2

vector <int> v(2) = 4 6

Ques : Find K Closest Elements

[Leetcode 658]

logn → to search/ ub, lb

T.C. → O(logn + K)

Case-3 If 'x' is not present in array

arr = 1 2 3 4 6

lb → 3

ub → 6

x = 5

K = 4

vector<int> v(k) = 4 6 3 2

Ques : Find K Closest Elements [Leetcode 658]

Case - 4

arr = 1 3 5 7 9 11

lb

ub

x = 3

k = 5

vector<int> v(k) = { 3 , 1 , 5 , 7 , 9 }

Ques : Find K Closest Elements [Leetcode 658]

Case - 5 If element is < nums[0]

arr = 1 2 3 4 5

x = -1

k = 4

ans = 1, 2, 3, 4

Ques : Find K Closest Elements [Leetcode 658]

Case-6 If x > nums[n-1]

arr = 1 2 3 4 5

x = 7

k = 3

v = { 3 4 5 }

ans = 3, 4, 5

Dry Run

```
int lo = 0;
int hi = n-1;
bool flag = false; // if x is present in arr or not
int t = 0; // representing index of ans
int mid = -1;
// binary search
while(lo<=hi){
    mid = lo + (hi-lo)/2;
    if(arr[mid]==x){
        flag = true; // present
        ans[t] = arr[mid];
        t++;
        break;
    }
    else if(arr[mid]>x) hi = mid - 1;
    else lo = mid + 1;
}
int lb = arr[hi];
int ub = arr[lo];
if(flag==true){
    lb = mid-1;
    ub = mid+1;
}
```

0	1	2	3	4	5
1	1	1	10	10	10
		hi	lo		
			mid		

$x = 9$

$x = 1$

vector<int> ans(1) = { }

Maza aa gaya

*Ques : Sum of Square Numbers [Leetcode 633]

→ c = 41

↓

$a^2+b^2 = c$

↓

$x = a^2$

$y = b^2$

a^2	b^2
0	41
1	40
2	39
3	38
4	32
⋮	
20	21
21	20
22	19
⋮	
41	0

$a^2+b^2 = c$

↓ ↓

x y

$x = b$, $y = c$

T.C. → O(c) → Gandi nai

Ques : Sum of Square Numbers [Leetcode 633]

c = 41

x = 0 ~~5~~ ~~9~~ 16

y = ~~41~~ ~~36~~ ~~32~~ 25

loop

```
if ( isPS(x) && isPS(y)) return true;
else if (!isPS(y)){
    y = sqrt(y) * sqrt(y);
    x = c - y;
}
else { // x is not perfect
    x = (sqrt(x)+1) * (sqrt(x)+1); y = c-x
}
```

Ques : Sum of Square Numbers [Leetcode 633]

Space Complexity : O(1)

Time Complexity :

M-I → $\frac{c}{2}$ no. of operations

= O(c)

M-II → $O(\sqrt{c})$

42

x	y
0	42
6	36
9	33
17	25
25	17

42

x	y
0	42
1	41
2	40
⋮	⋮
21	21
22	20

Ques : Sum of Square Numbers [Leetcode 633]

c = 7

x	y	
0	7	
1	6	
2	5	
3	4	
4	3	✓

c = 7

x	y	
0	7	
3	4	
4	3	✓

$\sqrt{c} + 1$ operations

Ques : Sum of Square Numbers [Leetcode 633]

$c = 102 \longrightarrow \sqrt{c} = 10$

x	y	
0	102	
2	100	
4	98	
21	81	
25	77	
38	64	
49	53	
53	49	✗

$O(\sqrt{c})$

Ques : Sum of Square Numbers [Leetcode 633]

Ques : Sum of Square Numbers [Leetcode 633]

1 2 3 1 1 days = 4

8x-3

ans = 4

lo	hi	mid
1	8	4
1	3	2

mid = 2

m = 4 * 2

count = 0 1