Linear search

10	20	30	40	50	60	70
0	1	2	3	4	5	6

target = 70

```
for (i=0 ; i<n; i++){
        if (arr[i] == target){
            return true
        }
        return false
}
```

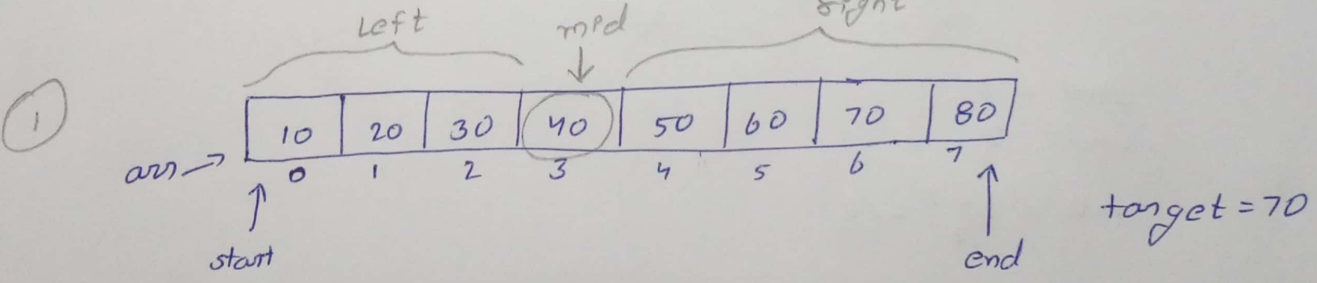
$T.C \Rightarrow O(n)$
↳ size of array

Binary search

↳ array → sorted → asc
                  → des

array → Monotomic function

1

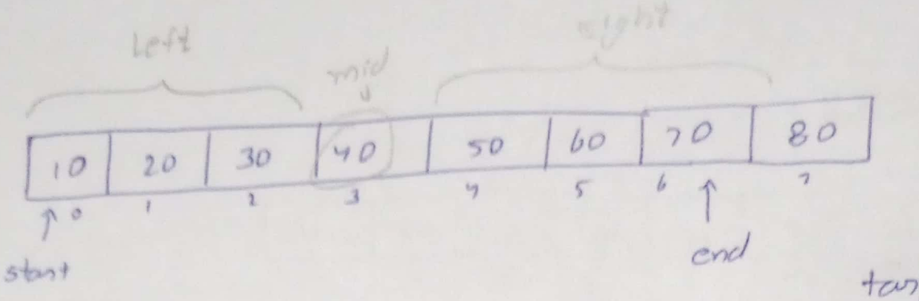


target = 70

start = 0 , end = 7

$mid = \frac{(s+e)}{2} = \frac{0+7}{2} = \boxed{3}$

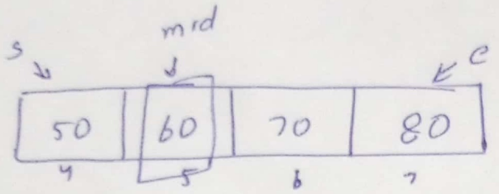
$\boxed{mid = s + (e-s)/2}$



target = 70

40 == 70 → F ⇒ 70 > 40 → right

(2)



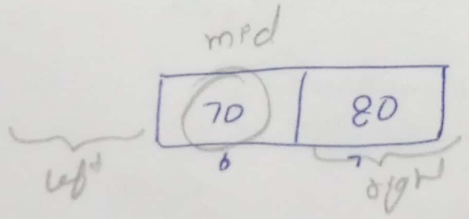
start = 4 , end = 7

mid = $\frac{4+7}{2}$ = 5

60 == 70 → F

70 > 60 → Right

(3)



start = 6

end = 7

mid = $\frac{6+7}{2} = \boxed{6}$

70 == 70 → Yes

→ return mid

mid

10	20	30	40	50	60	70	80
0	1	2	3	4	5	6	7

n

mid

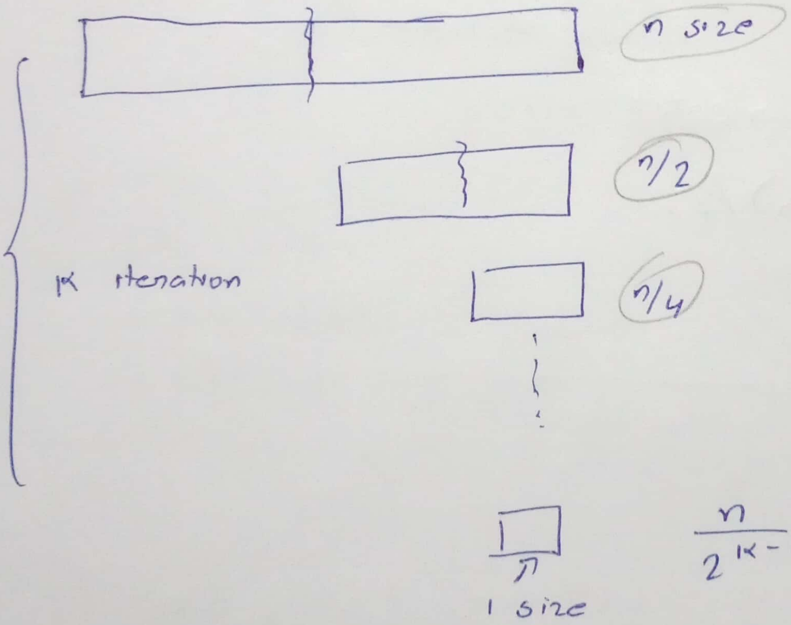
50	60	70	80
4	5	6	7

n/2

70	80
6	7

n/4

with every iteration
size reduce by half



$\frac{n}{2^0} = n$

$\frac{n}{2^1} = \frac{n}{2}$

$\frac{n}{2^2} = \frac{n}{4}$

$\frac{n}{2^{K-1}} \Rightarrow \frac{n}{2^K}$

$\frac{n}{2^K} = 1$

$n = 2^K$

$\boxed{\log_2 n = K}$ →

**

$\boxed{T.C \Rightarrow O(\log n)}$

① found

```
if (arr[mid] = = target)
        return mid

if (arr[mid] < target)
        s = mid + 1

if (target < arr[mid])
        e = mid - 1
```

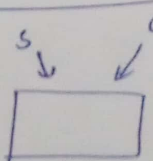
$\underbrace{\text{while}(s <= e)}_{\text{condition}}$

```
// mid update → catch
mid = s + (e - s) / 2;
```

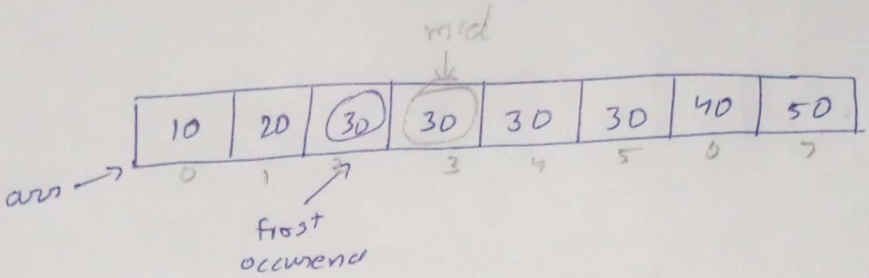
② Not found
↳ return -1

* Point to remember

start = = end → single element ko represent karta h



Find first occurence of a number in a sorted array



target = 30

ans = 2

① s=0 , e=7 , mid=3

30== 30 → true

↑ found

→ ans ko store kr lo

ans =3

→ left me chle jao

e=mid+1

found

→ ans store

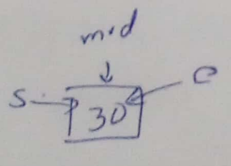
→ left me chle jana

② s=0 , e= 2 , mid = 1

20 = =30 → F

30 > 20 → right me jao

③

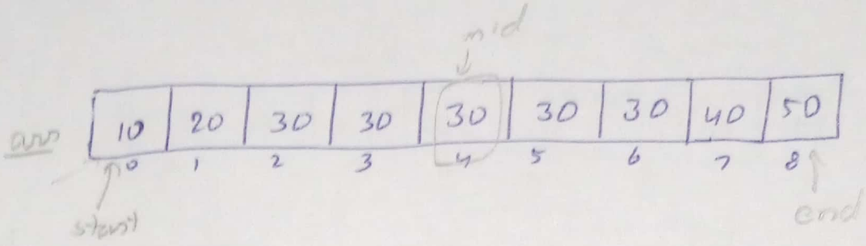


s = 2 , e = 2 , mid= 2

arr [mid] = = target

30==30 → found → ans → 2

Find last occurence in sorted array



s = 0
e = 8
mid = 4

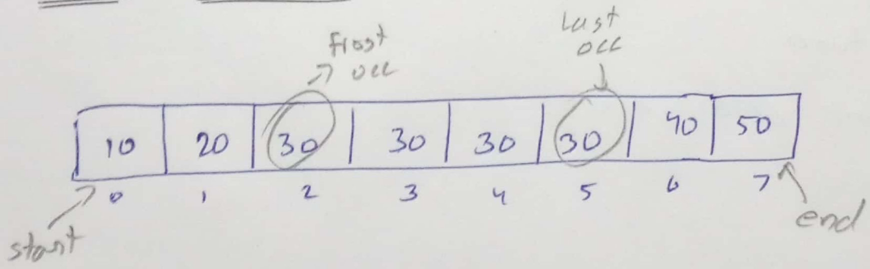
arr[mid] = = target

30 = = 30 → True
↓
found

→ ans store

→ right me chale jao

Find total occurence

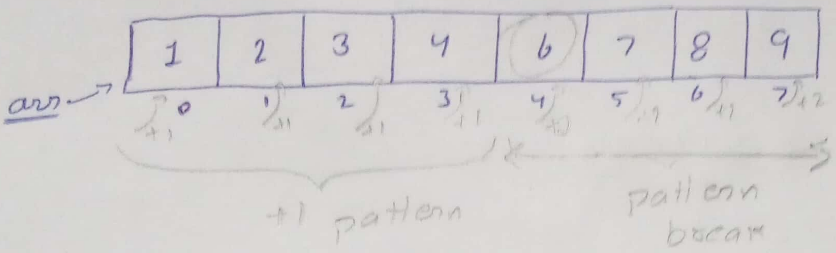


first occ = 2

last occ = 2

$$\boxed{\text{Total occurence} \Rightarrow \text{Last occ} - \text{first occ} + 1}$$

Find missing element in a sorted array



ans = 5

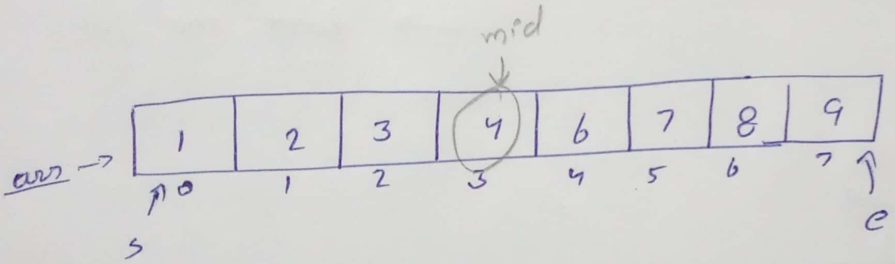
missing

Approach

binary search apply karnge $\boxed{O(\log n)}$ solve krna h

cond^n → sorted h

1



$s=0$ $e=7$ > $mid = \frac{0+7}{2} = 3$

diff = arr[mid] − mid

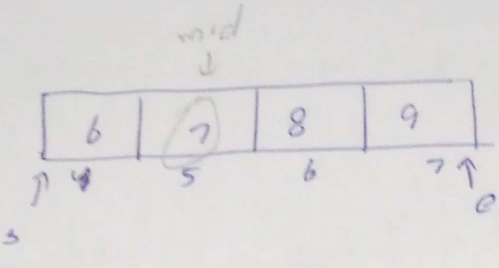
diff = 1

$\boxed{\text{diff} = 1}$ → right me ja → s = mid + 1

Base case

if (ans + 1 = = 0)
 return n+1

(2)



s = 4
e = 7 > mid = $\frac{4+7}{2}$ = (5)

diff = arr[mid] - mid

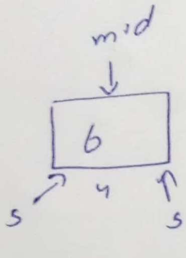
7-5

$\boxed{\text{diff} = 2}$

diff == 1 → false → ans store
ans = 7

→ Left me jao

(3)



s = 4

e = 4

mid = 4

diff = arr[mid] - mid

diff = 2

diff == 1 → No → ans = 4

Lef me ja (e=mid-1)

(4)

s = 4
e = 3 → $\overset{\text{cond}^n}{(s > e)}$ → ouk jao

$\boxed{\text{final Ans} = \text{ans} + 1}$

1 → n

1 → 9

1 2 3 4 5 6 7 8 9

0	1	2	3	4	5	6	7
1	2	3	4	5	6	7	8

(n-1) integer

s <= e

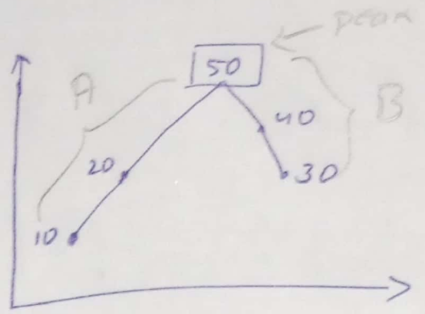
infinite loop

s < e

Imp?

Peak element in a mountain array

i/p => array[] = {10, 20, 50, 40, 30}



ans = 50

↳ o/p

Observation

Ⓐ

↓

arr[i] < arr[i+1] ← Peak right me exist karti h

Peak point

↓

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

arr[i] > arr[i+1]

arr[i] > arr[i-1]

Ⓑ

↓

arr[i] > arr[i+1]

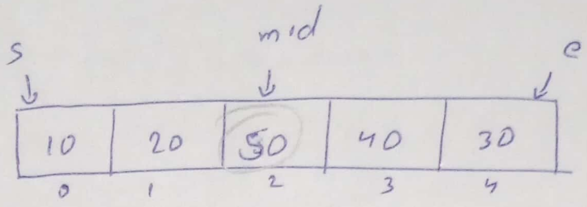
arr[i] < arr[i+1] → Ⓐ

↓

B + Peak

return start/end

1



$$\left.\begin{matrix} s = 0 \\ e = 4 \end{matrix}\right\rangle \; mid = \frac{0+4}{2} = 2$$

arr [mid] < arr [mid+1]

50 < 40 → false → B or peak

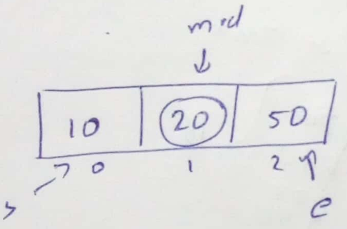
↓

Left me jao

$\boxed{e = mid}$

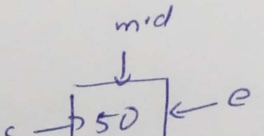
catch

2



$$\left.\begin{matrix} s = 0 \\ e = 2 \end{matrix}\right\rangle \; mid = \frac{0+2}{2} = 1$$

3



→ single element

→ Peak element

while (s<=e)

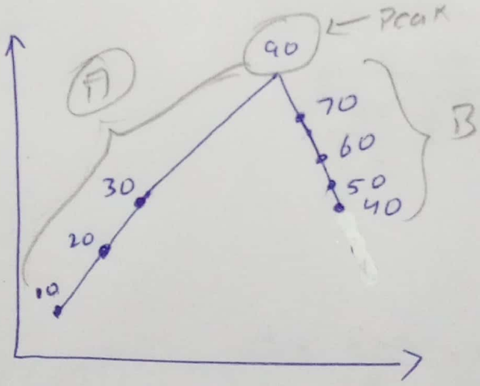
s = mid+1
↓
s = mid

↓

infinite loop ← e = mid-1
↓
e = mid

Dry run

arr	10	20	30	90	70	60	50	40
	0	1	2	3	4	5	6	7



True

A → $\boxed{arr[i] < arr[i+1]}$

B → $\boxed{arr[i] > arr[i+1]}$

True

Peak

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

(A) element pe khada hunga
↳ right me jaunga

(B) → left

(Peak) → vahi pe rahta

```
if (arr[mid] < arr[mid+1]){ → true → A
        // right
        s = mid+1
}
else {
        // left
        e = mid ;   ← catch
}
```

①

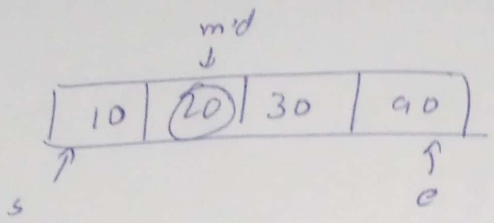
s			mid				e
↓			↓				↓
10	20	30	90	70	60	50	40
0	1	2	3	4	5	6	7

s = 0
e = 7 > mid = $\frac{0+7}{2}$ = 3

arr[mid] < arr[mid+1]

90 < 70 → false → e = mid

(2)

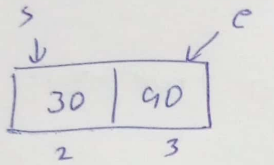


s = 0
e = 3 > mid = 1

arr[mid] < arr[mid+1]

20 < 90 → true → right

(3)

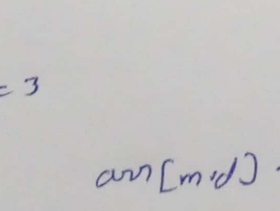


s = 2
e = 3 > mid = 2

arr[mid] < arr[mid+1]

30 < 90 → true
↳ right

(4)



s = 3

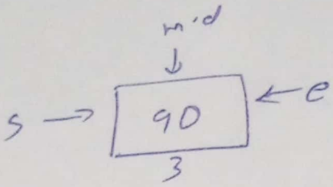
e = 3

mid = 3

arr[mid] < arr[mid+1]

90 < 70 → false
↳

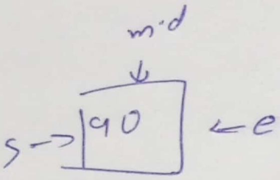
5



s = 3
e = 3
mid = 3

90 < 70 → T → e = mid

6



s = 3
e = 3
mid = 3

90 < 70 → True
↳ e = mid

Infinite Loop me fas gaye

condition to overcome this infinite Loop

while (s < e)

H/w → find pivot element

✓ return s/e