Searching

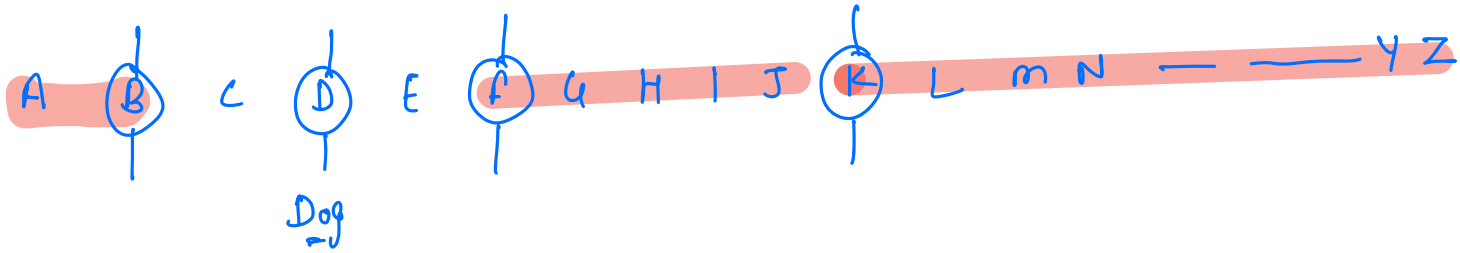
target → what to search?

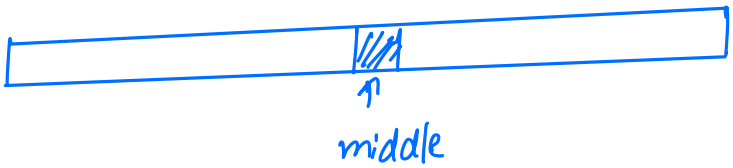
search-space → where to search/find it?

Word.	Newspaper	dictionary
contact	Phone diary	Directory / Phone contacts
Book.	Book shelf	library.

Arranged/organized in a particular order.

Dog.





Binary Search → reducing the search space by half in every single iteration based on some condition.

Q: Given a sorted array (ascending). find the idx of the element K. Return -1 if element K doesn't exist.

arr → [3 6 9 12 14 19 20 23 25 27]. K = 20

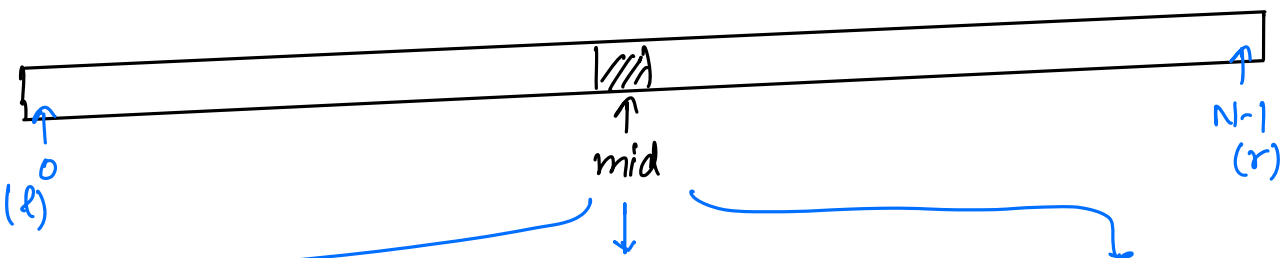
0 1 2 3 4 5 6 7 8 9

ans = 6.

idea-1. Linear Search (iterating on all elements) T.C → O(N)

idea-2 target → K

search-space → whole array [0, N-1]



arr[mid] < K

//discard left portion

[mid+1, N-1]

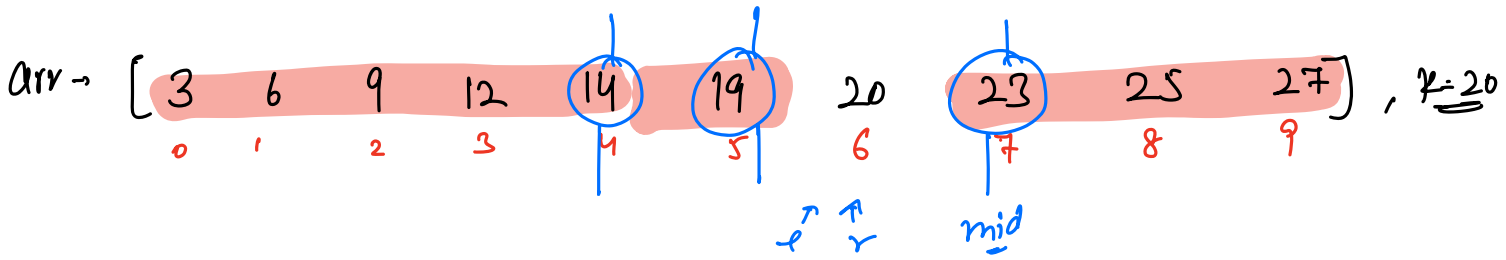
arr[mid] == K.

return mid

arr[mid] > K

//discard right portion

[0, mid-1]



l	r	mid	
0	9	$\frac{0+9}{2} = 4$	// Discard left portion → l = mid+1
5	9	$\frac{5+9}{2} = 7$	// Discard right portion → r = mid-1;
5	6	$\frac{5+6}{2} = 5$	// Discard left portion → l = mid+1
6	6	$\frac{6+6}{2} = 6$	return mid

code

```
l = 0, r = N-1
while( l <= r){
    mid = (l+r)/2;
    if (arr[mid] == k){
        return mid;
    }
    else if (arr[mid] < k){
        l = mid+1;
    }
    else{
        r = mid-1;
    }
}
return -1;
```

N
↓
N/2
↓
N/4
↓
N/8
↓
|
1

T.C → $O(\log_2 N)$
S.C → $O(1)$

$\Rightarrow O(\log_2 (\text{search-space}))$

Q1 Given a sorted array containing duplicates as well. Find freq of an element k. k=5.

	-5	-5	-3	0	0	1	1	1	5	5	5	5	5	5	5	8	10	15	15
arr[] →	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18

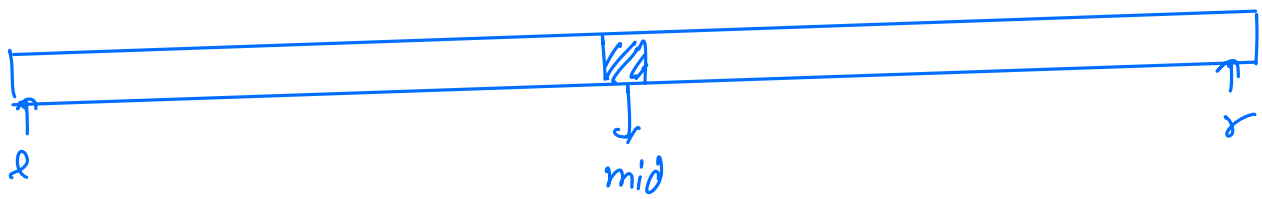
ans = 7

idea-1. Linear search and update count if arr[i] == K. [T.C → O(N), S.C → O(1)]

idea-2. [freq = last occ - first occ + 1]

target → first occurence of K.

search-space → [0, N-1]

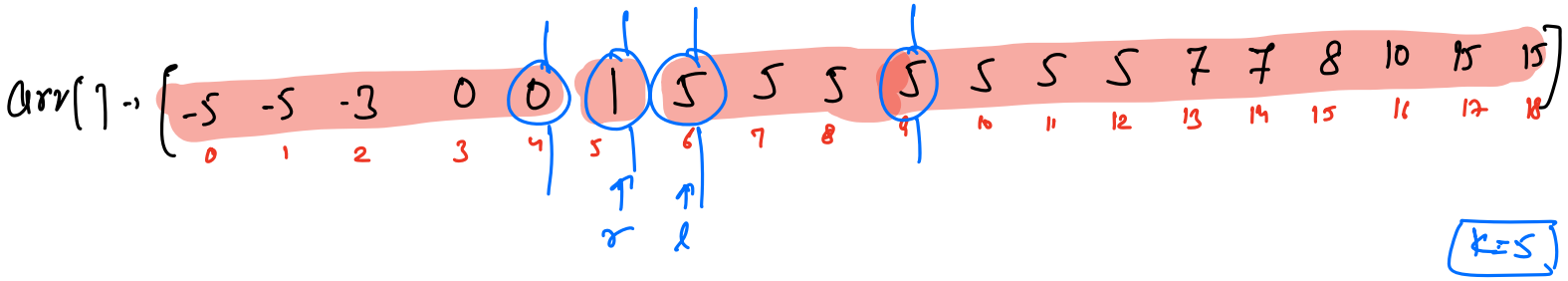


first occurence

arr[mid] < K	arr[mid] == K	arr[mid] > K
l = mid + 1	ans = mid r = mid - 1	r = mid - 1

last occurence

arr[mid] < K	arr[mid] == K	arr[mid] > K
l = mid + 1	ans = mid l = mid + 1	r = mid - 1



K=5

l	r	mid	
0	18	$\frac{0+18}{2} = 9$	arr[mid] == K, ans = 9, r = mid-1
0	8	$\frac{0+8}{2} = 4$	arr[mid] < K, l = mid+1
5	8	$\frac{5+8}{2} = 6$	arr[mid] == K, ans = 6, r = mid-1
5	5	$\frac{5+5}{2} = 5$	arr[mid] < K, l = mid+1
6	5	return ans.	

code → # first occurence

```
l = 0, r = N-1, fo = -1
while( l <= r){
        mid = (l+r)/2;
        if( arr[mid] == K){
                fo = mid;
                r = mid-1;
        }
        else if( arr[mid] < K){
                l = mid+1;
        }else{
                r = mid-1;
        }
}
```

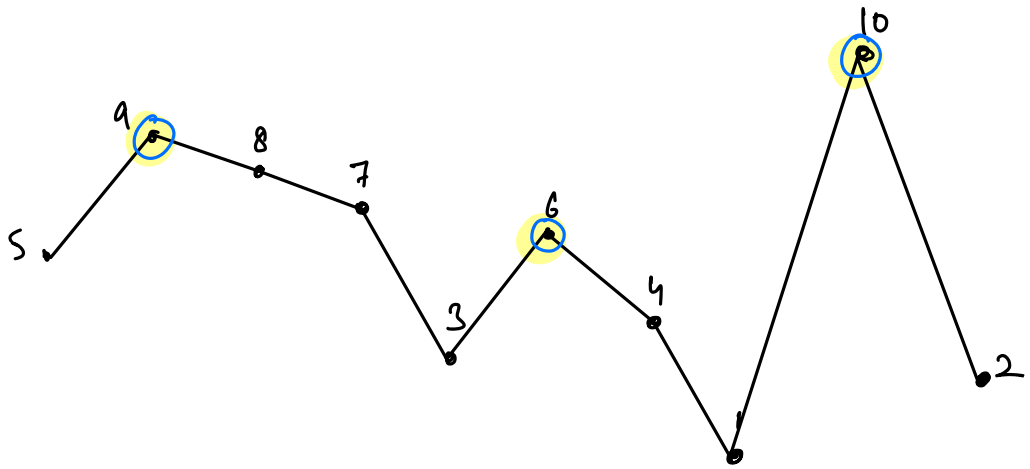
T.C → O(log N)
S.C → O(1)

Q1 Given an unsorted array of N distinct elements.
Find any one local maxima.

↓

[element which is not smaller than its adjacent elements]

arr[] → [5 9 8 7 3 6 4 1 10 2]



arr[] → [7 7 7 7]

idea-1. → Find the max element. ⇒ Linear Search T.C → O(N)

→ Iterate on all the elements & check

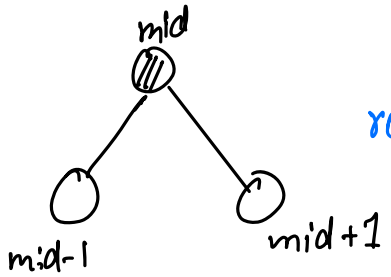
if arr[i-1] ≤ arr[i] && arr[i+1] ≤ arr[i]

T.C → O(N)

target → any local maxima

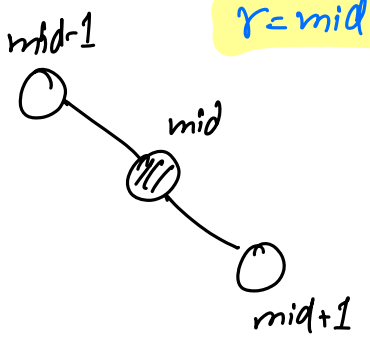
search space → [0, N-1]

①



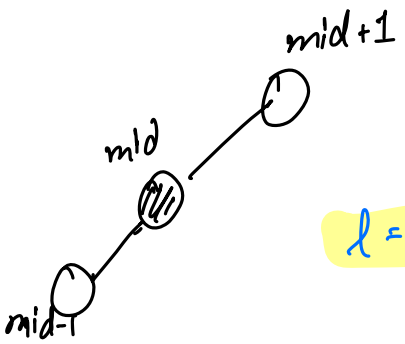
return arr[mid]

②



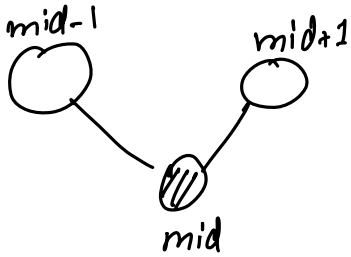
r = mid-1.

③



l = mid+1

④



Go either way

↓

r = mid-1

#code→

```
if (N==1) { return arr[0] }
if ( arr[0] ≥ arr[1] ) { return arr[0] }
if (arr[N-1] ≥ arr(N-2)) { return arr[N-1] }

l = 1, r = N-2
while( l ≤ r){
    mid = (l+r)/2;
    if (arr[mid] ≥ arr[mid-1] && arr[mid] ≥ arr[mid+1]){
        return arr[mid];          //Case-1
    }
    else if ( arr[mid] ≥ arr[mid-1]){
        l = mid+1                 //Case-3
    }
    else{
        r = mid-1                 //Case-2 & 4
    }
}
return -1;
```

T.C → $O(\log_2 N)$
S.C → $O(1)$

Q) Every element occurs twice except for one.
Find unique element.

Google

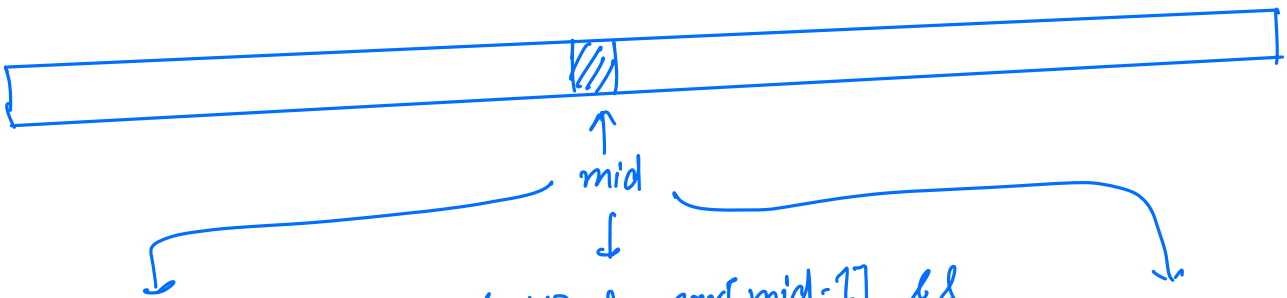
Note → Duplicates are adjacent to each other

ans = 9.

	3	3	1	1	8	8	10	10	9	6	6	2	2	4	4
idx	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14

target → unique element

Search-space → [0, N-1]



arr[mid] != arr[mid-1] && arr[mid] != arr[mid+1]

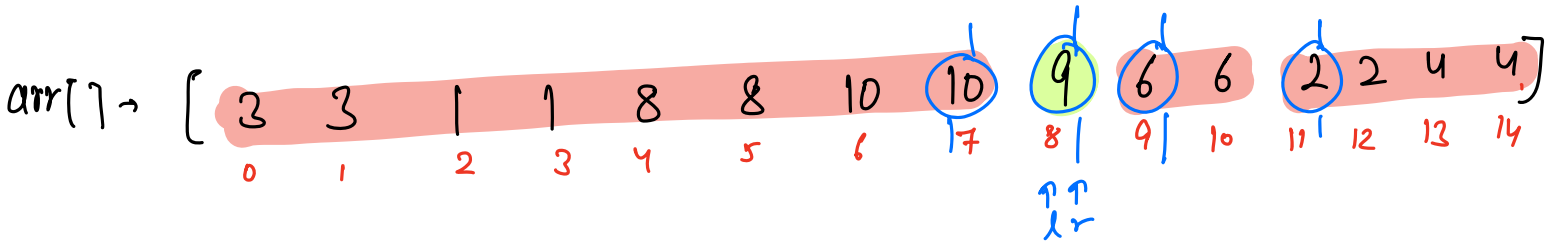
return arr[mid]

first occurence of arr[mid] ⇓ even idx

l = fo+2.

first occurence of arr[mid] ⇓ odd idx

r = fo-1



l	r	mid
0	14	$\frac{(0+14)}{2} = 7$
8	14	$\frac{8+14}{2} = 11$
8	10	$\frac{8+10}{2} = 9$
8	8	$\frac{8+8}{2} = 8$

f.o of 10 is present at 6 (even) idx.
$l = fo + 2$

f.o of 2 is present at 11 (odd) idx.
$r = fo - 1$

f.o. of 6 is present at 9 (odd) idx.
$r = fo - 1$

(arr[mid] is unique element
return arr[mid])

#code:→

```
if (N==1) { return arr[0] }

if (arr[0] != arr[1]) { return arr[0] }

if (arr[N-1] != arr[N-2]) { return arr[N-1] }

l=2, r= N-3
while(l <= r){
    mid = (l+r)/2;
    if ( arr[mid] != arr[mid-1] && arr[mid] != arr[mid+1]){
        return arr[mid];
    }
    fo = mid;
    if (arr[mid-1] == arr[mid]){
        fo = mid-1;
    }
    if ( fo % 2 == 0){
        l = fo+2;
    }
    else{
        r = fo-1;
    }
}
```

T.C → O(log N)
S.C → O(1)

- target →
- search-space →
- Condition on the basis of which search space reduced by half.

⇓

Binary Search