Binary Search : Searching words in a dictionary. potato

Divide into 2 parts.

	0	1	2	3	4	5
Eg	3	10	17	18	24	37
	s		mid			e

Q1) Given sorted array with distinct elements, search for K. If K is not present return -1
(return index)

0	1	2	3	4	5	6	7	8	9
3	6	9	12	14	19	20	23	25	27

Idea : Use binary search

Case 1 : ar[mid] == k return mid

Case 2 : ar[mid] < k goto right

[mid] .... k

Case 3 : ar[mid] > k goto left

0	1	2	3	4	5	6	7	8	9
3	6	9	12	14	19	20	23	25	27

$k = 20$

l	h	mid	
0	9	4	l = mid+1
5	9	7	h = mid-1
5	6	5	
6	6	6	found !!!

return 6

$N \to N/2 \to N/4 \to \cdots \to 1$ ($\log n$)

Code

```
int search( int ar[], int N, int k){
    l = 0          h = n-1
    while (l ≤ h) {
        mid = (l+h)/2
        if (ar[mid] == k)    return mid
        if (ar[mid] < k)     l = mid+1
        else                 h = mid-1
    }
    return -1
}
```

TC: $O(\log n)$

SC: $O(1)$

Q2 Sorted array, find index of first occurence of elem K.

0	1	2	3	4	5	6	7	8	9	10
-5	-5	-3	0	0	1	1	5	5	5	5

$K = 5 \Rightarrow 7$

$K = 0 \Rightarrow 3$

$K = -3 \Rightarrow 2$

Use Binary Search

Case I ar[mid] == k

kk mid kkk

ans = mid

goto = left

Case II ar[mid] < k

mid ... k

goto right

Case III ar[mid] > k

mid

goto left

0	1	2	3	4	5	6	7	8	9	10
-5	-5	-3	0	0	1	1	5	5	5	5

l	h	m	
			k = 5
0	10	5	
6	10	8	ans = 8
6	7	6	
7	7	7	ans = 7
7	6		STOP!!!!

Code

```
int search( int ar[], int N, int k){
    l = 0          h = n-1
    while (l ≤ h) {
        mid = (l+h)/2
        if (ar[mid] == k) {ans = mid   h = mid-1}
        if (ar[mid] < k)   l = mid + 1
        else   h = mid-1
    }
    return ans
}
```

TC: $O(\log n)$

SC: $O(1)$

Q3 Every element occurs twice except 1 which appears once. Find unique element. Note: Duplicates are adjacent

	0	1	2	3	4	5	6	7	8	9	10
Eg -	3	3	1	1	8	8	10	10	19	6	6

Idea :

TC:

Obs: Before unique elem → All 1'st occurences at even indexes
After unique elem → All 1'st occurences at odd indexes
Hence binary search

target ⇒

what to search ⇒ indexes of the array

Case I ar[mid] => unique return ?

Case II if (ar[mid] == ar[mid-1]) A A

0	1	2	3	4	5	6	7	8	9	10
3	3	1	1	8	8	10	10	19	6	6

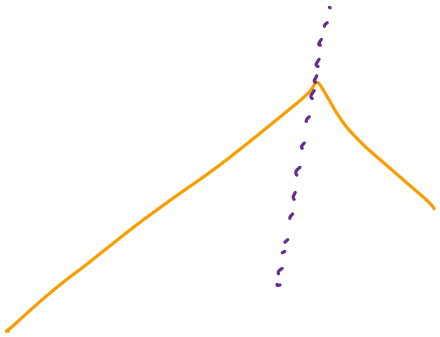
l	h	m
0	10	5
6	10	8

return a[8]

Code

```
int findUnique ( int arr [], int N) {
    l = 0        h = n-1
while (l ≤ h) {
    mid = (l+h) / 2
    if ( (mid==0 || arr [mid] != arr [mid-1] )      &&
        ((mid==n-1) || arr [mid] != arr [mid+1]) )
        return arr [mid]
    if ( (mid==0 || arr [mid] == arr [mid-1] )) {
            // second occ
        if (mid % 2 ==0)
                h = mid-1
        else        l = mid +1
    }
    else {    // first occ
        if (mid % 2 != 0)
                h = mid-1
        else        l = mid +1
    }
}
```

TC:
$O(\log n)$



Q4 Given an increasing-decreasing array. Find max elem

Eg1 1 3 5 2 -1 ans = 5

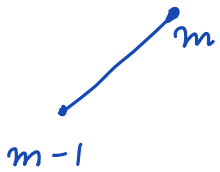
Eg2 1 3 5 10 15 12 6 ans = 15

Idea: target : search for max elem

Case 1 ar[mid] > ar[mid-1] && ar[mid] > ar[mid+1]

return ar[mid]

Case 2



ar[mid] > ar[mid-1]

goto right

Case 3

ar[mid] < ar[mid-1] &&

goto left

Code

```
int search ( int ar[], int N ){
    l = 0          h = n-1
    while (l ≤ h) {
        mid = l+h/2
        if (( mid == 0 || a[mid-1] < a[mid] ) &&
              mid == n-1 || a[mid+1] < a[mid] ) )
                return a[mid]
        if ( mid == 0 || A[mid-1] < a[mid])
                l = mid+1
        else    h = mid-1
    }
}
```

0	1	2	3	4	5	6
1	3	5	10	15	12	6

l	h	
0	6	3
4	6	5
4	4	4 peak

Q5 Find any local minima.

Distinct integers, a[i] is minima if smaller than both neighbours.

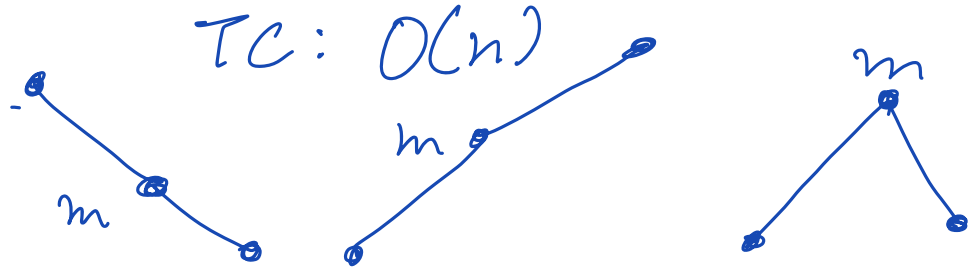
a[0] & a[n-1] have only 1 neighbour

0 1 2 3 4 5 6

Eg: a[7] = [9, 6, 3, 14, 5, 7, 4]

a[7] = [24, 21, 19, 17, 15, 9, 7]

Brute: Iterate on each elem & check. TC: O(n)



Idea: For any elem a[mid]

1) If smaller than neighbours ⇒ return a[mid]

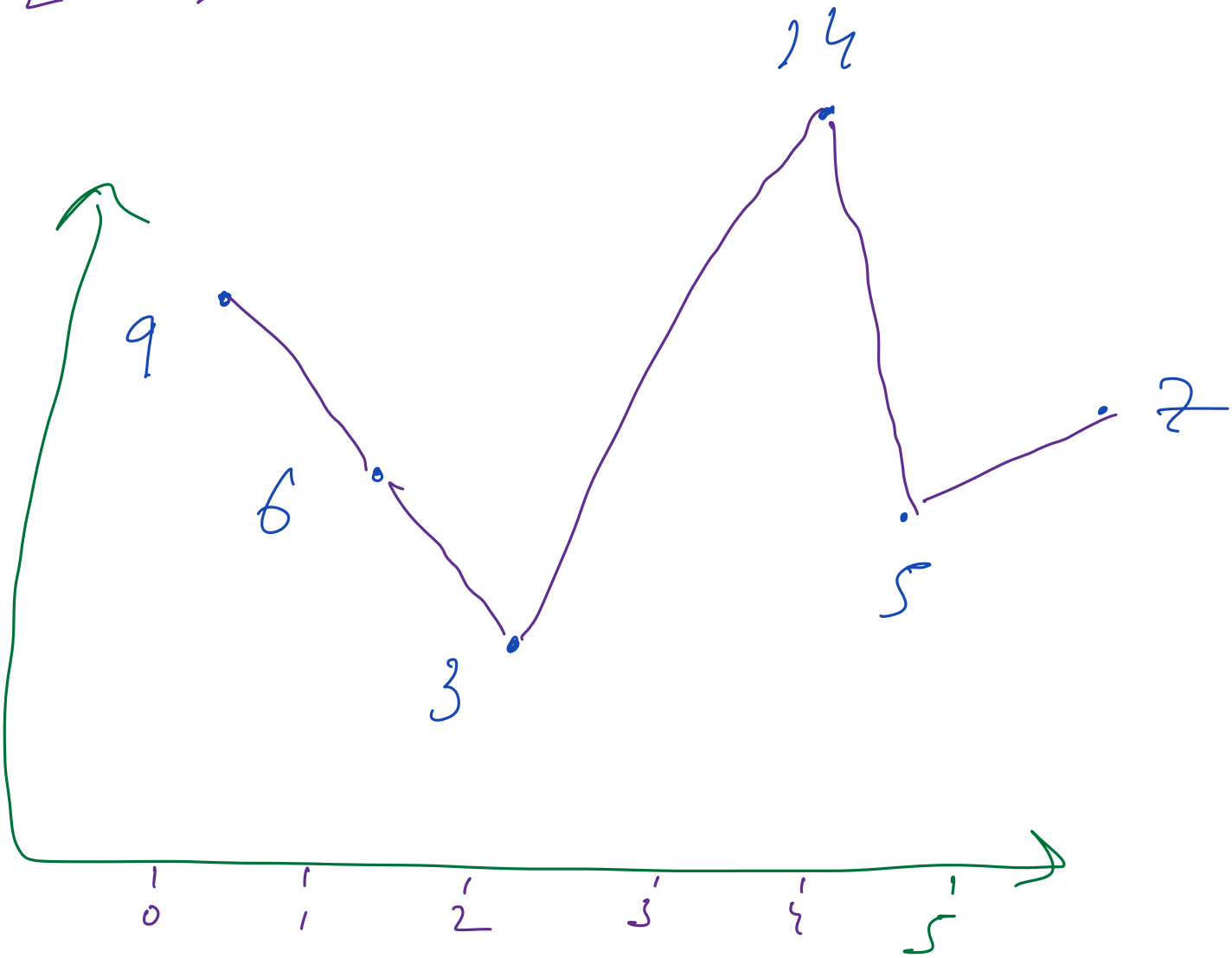
2) If left neighbour smaller ⇒ goto left

6 8 10

3) If right neighbour smaller ⇒ goto right

0	1	2	3	4	5	6
9	6	3	14	5	7	4

l	h	m
0	6	3
0	2	1
2	2	2



```
int search ( int ar[], int N )
{
    l = 0          h = n-1
    while ( l ≤ h ) {
        mid = l+h/2
        if (( mid == 0 || a[mid-1] > a[mid] ) &&
              mid == n-1 || a[mid+1] > a[mid] ) )
                return a[mid]
    if ( mid == 0 || A[mid-1] > a[mid] )
            l = mid+1
    else    h = mid-1
    }
}
```

{done}