Lecture :- Binary search - 1

Agenda

- Introduction
- first occurence in sorted array
- Unique element
- Max element in inc / decr array
- Local minima in an array

Introduction to searching

Bro | sis ⟶ Police ⟶ target = bro | sis
search space = fair | Mela.

Word ⟶ Dict ✓
Newspaper
Book

Target
Search space.
Condition

Binary Search

Qu. Given A[n] of distinct elements. Return the index of an element k. If not found, return -1.

(A[n] — sorted)

0	1	2	3	4
3	8	11	19	28

k = 19. ans = 3

0	1	2	3	4
3	8	11	19	28

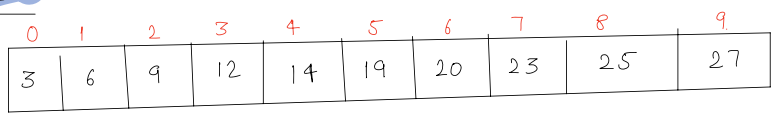
k = 27 ans = -1

Brute force: O(n) — traverse the array.

3	1	2	8	18

k = 8 —— O(n).

Approach 2



K = 12

target = 12

mid value = 14 $\left[\frac{0+9}{2}\right]$

target < 14

Search space = (0 - 3) idx

↑ mid - 1

K = 25

target = 25

mid value = 14 $\left[\frac{0+9}{2}\right]$

target > 14

Search space = (5 - 9) idx.

↑ mid+1

Dry run:

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

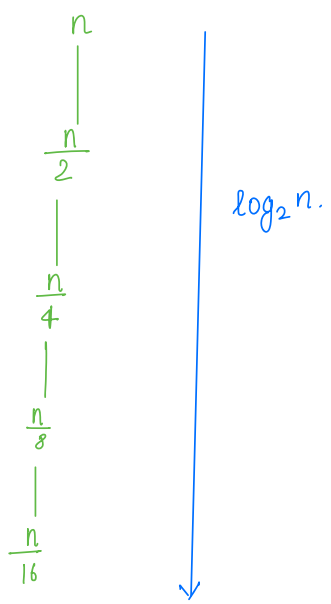
target = 12.

s	e	Action
0	9	mid = 14 target < mid s = 0 , e = 3.
0	3	midIdx = 1 mid = 6 12 > 6 s = 2 e = 3
2	3	midIdx = $\frac{2+3}{2}$ = 2 mid = 9 12 > 9 s = 3, e = 3
3	3	midIdx = 3 mid = 12 12 = 12 (return midIdx).

Binary search pseudocode

```
int binarySearch(int[] A, int target) {
    s = 0;
    e = A.length-1;
    while( s <= e ) {
        midIdx = (s+e)/2;
        mid = A[midIdx];
        if( target == mid) {
            return midIdx;
        } else if( target < mid) {
            e = mid-1;
        } else {
            s = mid+1;
        }
    }
    return -1;
}
```

TC: $\log_2 n$.
SC: O(1)



Qu Given A[n] sorted, find first occurence of target element

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
-5	-5	-3	0	0	1	1	5	5	5	5	5	5	5	10	10	15	15

target = 5 ans = 7

target = 15 ans = 16

target = -5 ans = 0

Brute force: O(n)

Approach 2

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
-5	-5	-3	0	0	1	1	5	5	5	5	5	5	5	10	10	15	15

target = 5

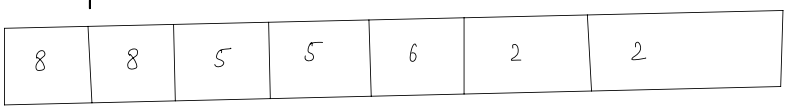
s	e	midIdx = (s+e)/2	Action
0	17	8	mid = 5 5 = 5 ans = 8 s = 0 e = 7
0	7	3	mid = 0 5 > 0 s = 4 e = 7
4	7	5	mid = 1 5 > 1 s = 6 e = 7
6	7	6	mid = 1 5 > 1 s = 7 e = 7
7	7	7	mid = 5 5 = 5 ans = 7 s = 7 e = 6
7	6		stop

Pseudocode

```
int firstOccurence(int[] A, int target) {
    s = 0;
    e = A.length-1;
    ans = -1;
    while( s <= e ) {
        midIdx = (s+e)/2;
        mid = A[midIdx];
        if( target == mid) {
            ans = midIdx;
            e = mid-1;
        } else if( target < mid) {
            e = mid-1;
        } else {
            s = mid+1;
        }
    }
    return ans;
}
```

TC: $O(\log_2 n)$
SC: $O(1)$

Qu Given A[n]. Every element occurs twice except for one element.
find that unique element.
Duplicate el are adjacent to each other but array is unsorted.



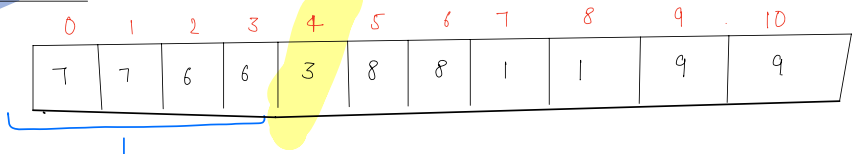
ans =

Brute force

xor — O(n)

Hashmap — O(n), O(n)

Approach 2



first occ of 7 = 0
" " " 6 = 2

fir occ of any no is even idx.

first occ of 8 = 5
" " " 1 = 7
" " " 9 = 9

fir occ of any no is odd idx.

A[mid] != A[mid-1] && A[mid] != A[mid+1] } return A[mid]

Dry run:

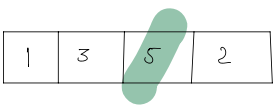
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
3	3	1	1	8	8	10	10	19	6	6	2	2	4	4

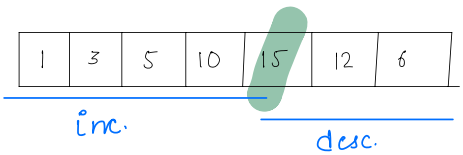
s	e	midIdx	mid	Actions
0	14	7	10	10 == A[mid-1] Not unique el. fi occ of 10 = 6 arr[mid] == arr[mid-1] → mid-1 f.o f.o = 6 s = 8 e = 14
8	14	11	2	Not unique. fi occ of 2 = 11 arr[11] == arr[10] → f.o = 10 != f.o = 11 s = 8 e = 10
8	10			
⋮				

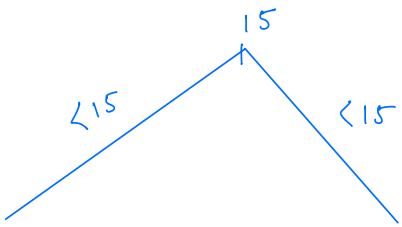
Pseudocode

```
if ( A[mid] != A[mid-1]   &&
     A[mid] != A[mid+1] )
        return A[mid]

f.o =  if ( A[mid] == A[mid-1] )
           mid-1 is f.o
       else  mid is f.o.

if ( f.o is odd idx) {
        left side

} else {
        right side.
}
```

Qu Given increasing — decreasing array with distinct elements. find maximum element.



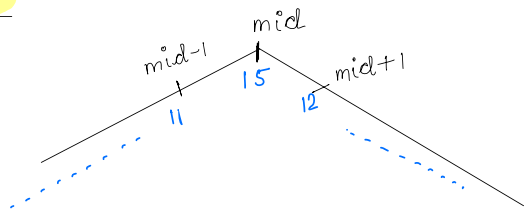




Brute force: O(n)

Approach

Case1

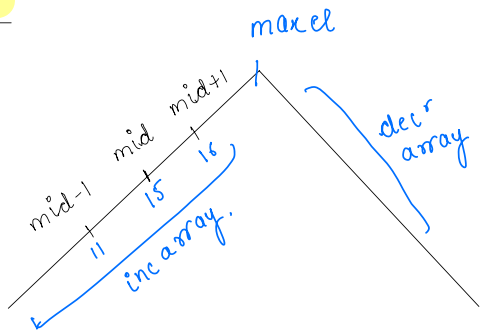


arr[mid] > arr[mid-1] &&

arr[mid] > arr[mid+1]

ans ⇒ arr[mid]

Case2

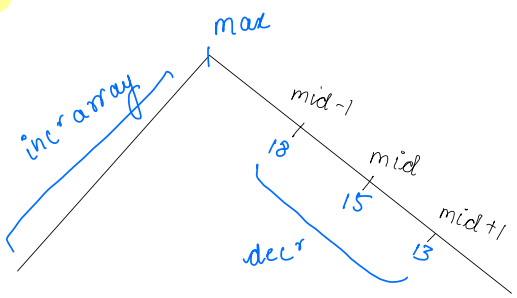


arr[mid] > arr[mid-1] &&

arr[mid] < arr[mid+1]

ans ⇒ go to right

Case3



arr[mid] < arr[mid-1] &&

arr[mid] > arr[mid+1]

ans = go left

arr[mid] > arr[mid-1] && ⟶ return arr[mid].

arr[mid] > arr[mid+1]

arr[mid] > arr[mid-1] && ⟶ right s = mid+1

arr[mid] < arr[mid+1]

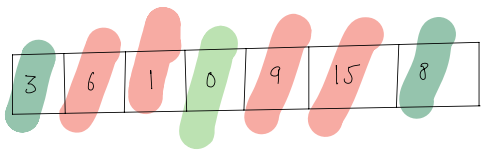
arr[mid] < arr[mid-1] && ⟶ left e = mid-1.

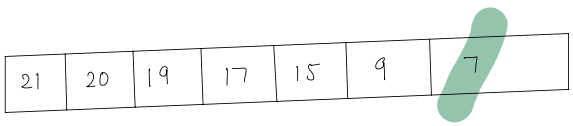
arr[mid] > arr[mid+1]

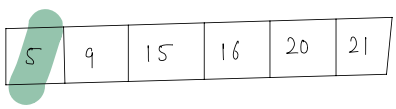
Break: 8:26:8:36

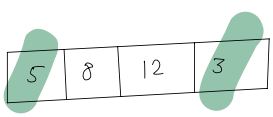
Qu Given A[n] of distinct elements, find any local minima in array.

Local minima: A no. smaller than its adjacent neighbours









Brute force: $O(n)$

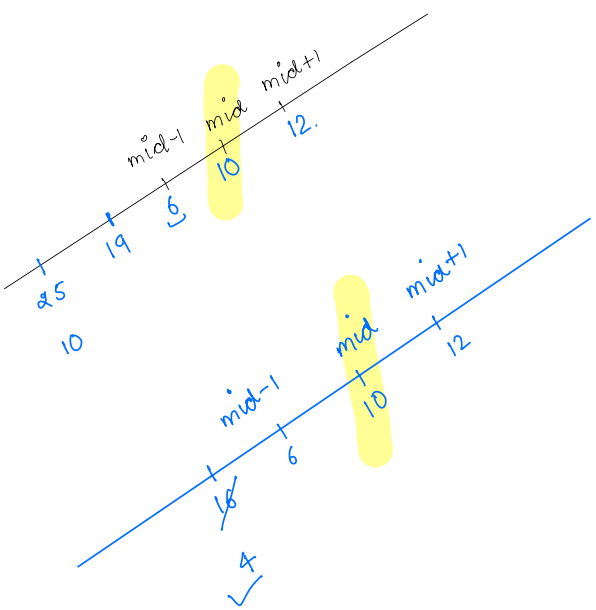
Approach

Case1.

arr[mid] < arr[mid-1] &&

arr[mid] < arr[mid+1] ans = 10

Case2:

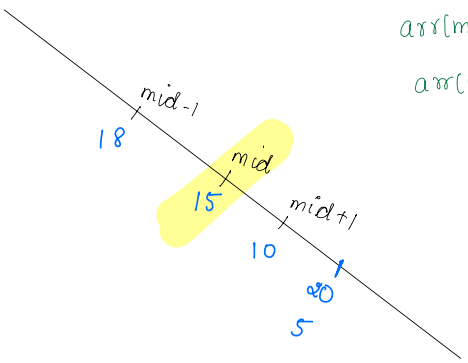


arr[mid] > arr[mid-1]

&& arr[mid] < arr[mid+1]

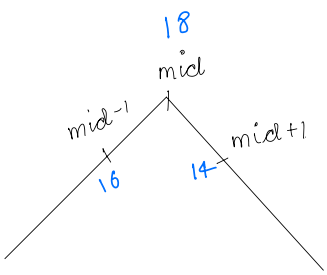
Left

Case3



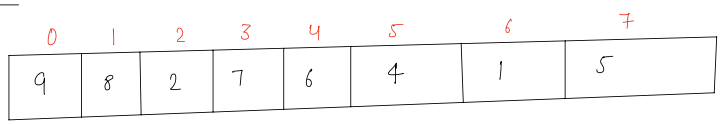
arr[mid] < arr[mid-1] &&
arr[mid] > arr[mid+1]
right

Case4



if (arr[mid] > arr[mid-1] &&
arr[mid] > arr[mid+1])
↓
any sides — left / right

Dry run:-



s	e	midIdx	Action
0	7	3	mid = 7 7>2 && 7>6 left s=0, e=2
0	2	1	mid=8 8<9 && 8>2 right s=2 e=2
2	2	2	mid = 2 2<8 && 2<7 return ans. (2).

TC: $O(\log_2 n)$

SC: O(1)

Pseudocode

```
int localMinima(int[] A) {
```

h/w.

Thankyou