Todays content

1. Peak Element
2. Find Unique Element

Google:

Given a unsorted arr[] with all distinct elements return any one local maxima

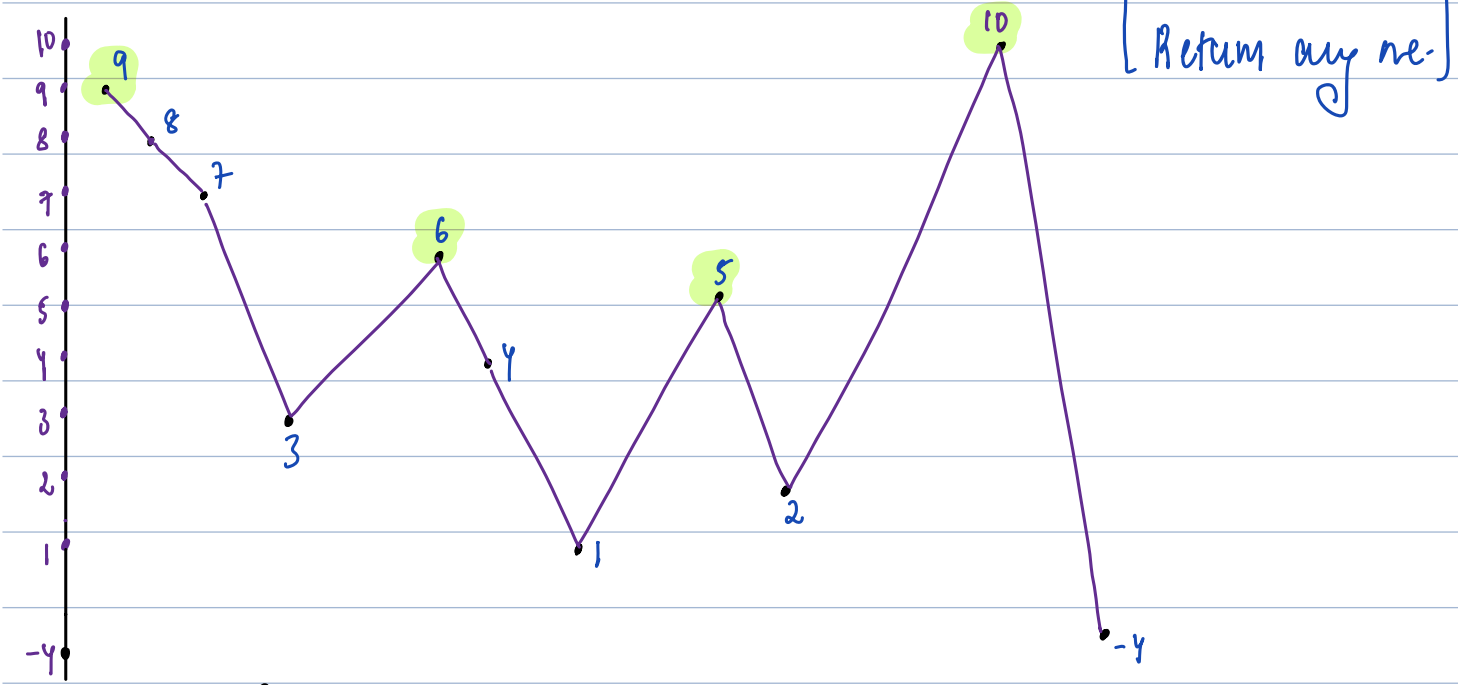
localmaxima: An element is said to be local maxima, if > than it's adjacent elements [immediate left & right]

arr[i] local maxima: arr[i] > arr[i-1] && arr[i] > arr[i+1]

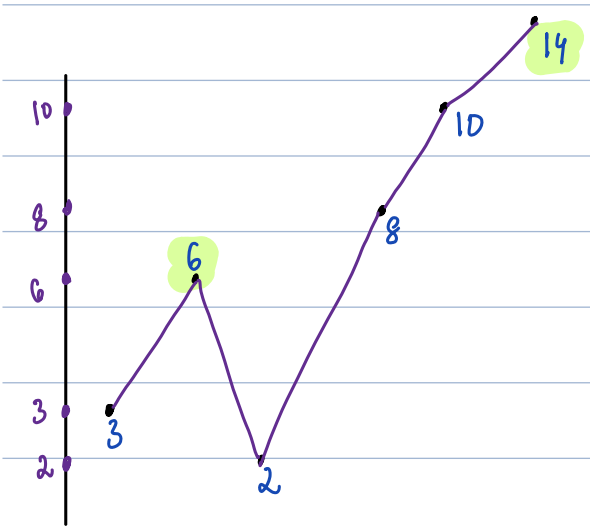
arr[0] local maxima: arr[0] > arr[1]

arr[N-1] local maxima: arr[N-1] > arr[N-2]

	0	1	2	3	4	5	6	7	8	9	10	local maxima
Ex: arr[11] = {	9	8	7	3	6	4	1	5	2	10	-4 }	[9 6 5 10 Return any one]



Ex: arr[6] = { 3 6 2 8 10 14 } local maxima = 6, 14 return anyone



#Idea1: Iterate in arr[] TC: O(N) SC: O(1)

for every arr[i]: Compare with adj if it's local maxima return it

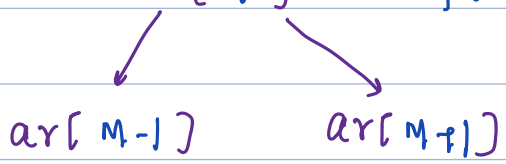
#Idea2: Iterate in arr[]: TC: O(N) SC: O(1)

Calculate & return max of arr[]

Idea:

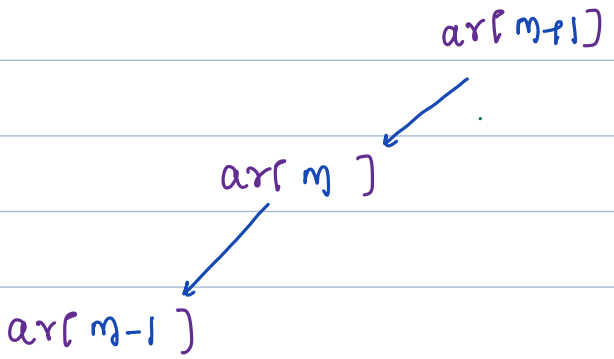
Target = Any One local maxima Search Space = In arr[]

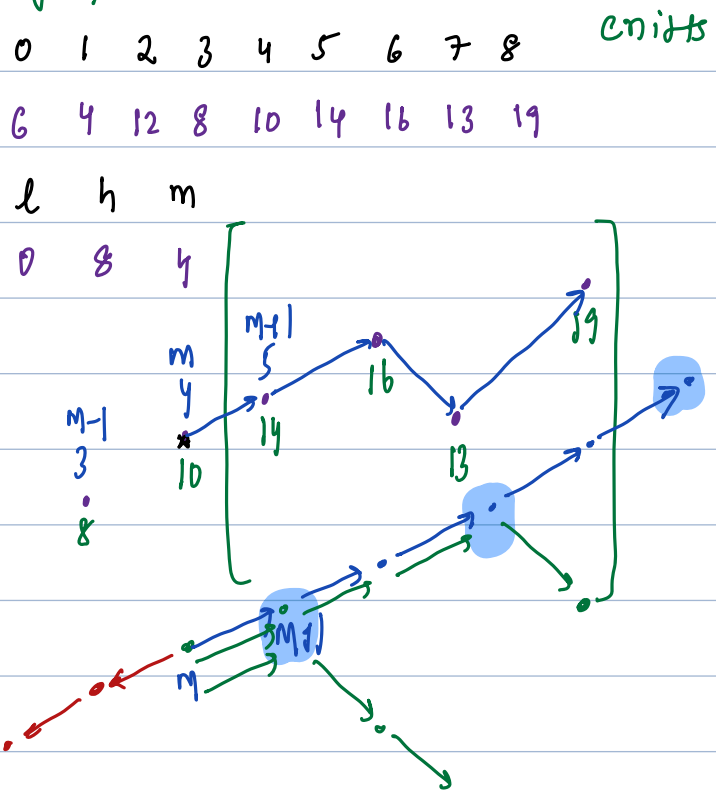
Case-1



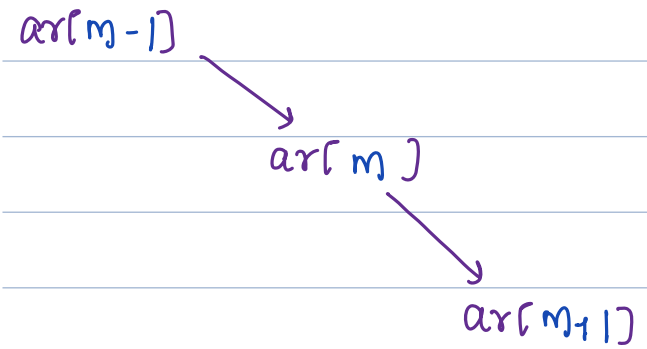
if[ar[m-1] < ar[m] && ar[m] > ar[m+1]]{
return ar[m];

Case-2 if[ar[m] < ar[m+1]] { #goto right, gtf a guranteed local maxima }

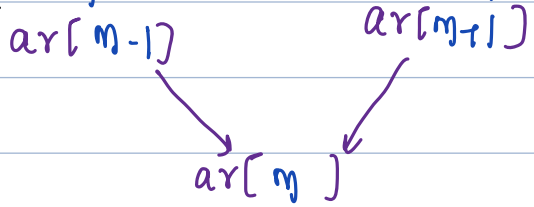




Case-3 if [ar[m-1] > ar[m]) {
#goto left



Case-IV: If Both sides are inc, go anywhere to get local maxima



#obs: Goto side where data increases.

```
int localmaxima(vector<int> &ar){ TC: O(log N/2)  SC: O(1)
    int N = ar.size();
    if(N==1){return ar[0]}
    if(ar[0]>ar[1]){return ar[0]}
    if(ar[N-1]>ar[N-2]){return ar[N-1]}
    int l=1, h=N-2;
    while(l<=h){
        int m=(l+h)/2;
        if(ar[m-1]<ar[m] && ar[m]>ar[m+1]){return ar[m]}
        else if(ar[m]<ar[m+1]){ #goto right
            l=m+1;
        }
        else{ # ar[m-1]>ar[m]  goto left
            h=m-1;
        }
    }
}
```

Note: Comparing adj elements, take care of corner elements

if m==0: ar[-1]<ar[0]: Error.

if m==N-1: ar[N-1]>ar[N]: Error.

Q: Google

Given a unsorted arr[] return anyone local maxima

localmaxima: An element is said to be local maxima, if >=than it's adjacent elements [immediate left & right]

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

If i==0: arr[0] is local maxima ⇒ arr[0] >= arr[1]

If i==N-1: arr[N-1] is local maxima ⇒ arr[N-1] >= arr[N-2]

```
int localmaximaRepeatim(vector<int>& arr){

}
```

18. Every element occurs twice, except for 1, find a unique element

Note: Duplicates are adjacent to each other

Eg1:

	0	1	2	3	4	5	6	7	8	9	10
ar[] = {	6	6	2	2	7	9	9	4	4	10	10 }

ans = 7

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
ar[] = {	3	3	1	1	8	8	10	10	19	6	6	2	2	4	4 }

ans = 19

#Idea1: 1. Calculate XOR of all elements

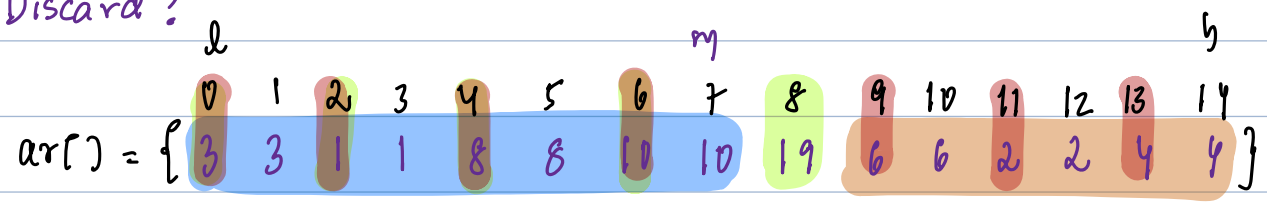
2. Iterate in arr[]: Compare adj elements & return unique ele

TC: O(N) SC: O(1)

#Idea2:

Target: Unique element Searchspace: In arr[]

Discard ?



if m is here:

goto right

if 1st occurence is Even:

On left

goto right

if m is here

goto left

if 1st occurence is Odd:

On right

goto left

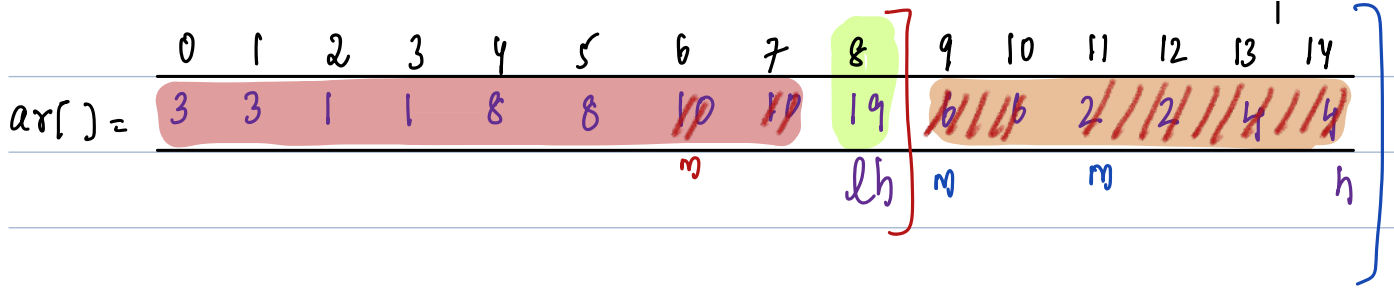
#Note: It is possible that m can find m 2nd occurence

Con: if m is m 2nd occ: Bring to 1st

```
if (ar[m-1] == ar[m]) {
    m = m-1
}
```

ar[m-1] ar[m]

1st 2nd



l	h	m	1st nr 2nd occ	Even nr Odd	
0	14	7	ar[m]==ar[m-1] : m=m-1	m=6	m: Even : goto right,	l=m+2;
8	14	11	1st occurence	m=11	m: Odd : goto left,	h=m-1;
8	10	9	1st occurence	m=9	m: Odd : goto left,	h=m-1;
8	8	8	unique ele; retun ar[m];			

```
int unique(vector<int> &ar){ TC: O(log N/2)  SC: O(1)
    int N = ar.size();
    if(N==1){ retun ar[0];}
    if(ar[0] != ar[1]){ retun ar[0]}
    if(ar[N-1] != ar[N-2]){ retun ar[N-1]}
    int l=1, h=N-2
    while(l<=h)
        int m = (l+h)/2;
        if(ar[m-1] != ar[m] && ar[m] != ar[m+1]){ retun ar[m] }

        if(ar[m-1]==ar[m]){ #m m 2nd occ: Bring to 1st occ
            m = m-1;
        }

        if(m%2==0){ #on left
            l=m+2; #goto right
        }else{ #m%2==1, m right
            h=m-1;
        }
    }
}
```

if m==0: ar[-1] != ar[0]
if m==N-1: ar[N-1] != ar[N]