Single Element in Sorted Array

arr = [1,1,2,2,3,(4),5,5,6,7,8,7,8,3]

(Single Element which is unique as well)

→ Brute Force Approach

Linear Search O(n)

Single $A[i-1] \neq A[i] \neq A[i+1]$

→ Optimal Approach.

Binary Search ⟶

① $mid = (s + (e-s/2))$

② $A[mid-1] \neq A[mid] \neq A[mid+1]$

return mid

Array having unique element would always have the odd number size.

Perfect Duplicates ⟶ Even

Duplicate + Single ⟶ Odd

[1,1,2,3,(3),4,4,8,8]

N

Save

Odd Number ⟶ So unique element must exist here

[1,1,2,3,3,4,4,8,8] → mid%2==0

L & R → Even Number Size

A[m-1] = A[m]

↓

left duplicate + Single Odd

A[m] = A[m+1]

↓

Right Duplicate + Single Odd

[3,3,7,7,10,11,11] Unique

mid%2!= 0

L & R Odd Number

A[m-1] = A[m] → Right

A[m] = A[m+1] → Left

Pseudcode

```
while(st<=end){
mid=(st+(e-s/2))
if(A[m]!=A[m]!=A[m+1])
    return A[mid]
if(mid%2==0) //Even
  if(A[m-1]=A[m])- end=mid-1
  else st=mid+1
else -odd
  if(A[m-1]==A[m])
      st=mid+1
  else
      end=mid-1
```

T.C O(logn)

S.C O(1)

Corner cases

① if(mid==0 && A[0] != A[1]){
return A[mid]}

② if(mid=n-1 && A[n-1] != A[n-2]
return A[mid]

③ if(n==1) return A[0].