Single element in sorted array

arr = [1, 1, 2, 3, 3, 4, 4, 8, 8]

output = 2

return element amongst the duplicates in sorted order

O(log n) TC & O(1) SC

BINARY SEARCH

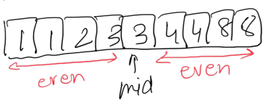
single ⇒ A[i-1] ≠ A[i] ≠ A[i+1]

MOST IMP PART

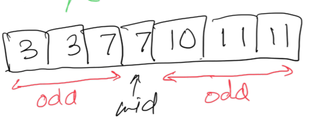
choosing search space

Question mein always odd size ka array hoga kyuki if we have n unique values so every n besides the single element will be dup so 2*n + 1 ⇒ odd

so to choose the search space we will choose the side jidar A[mid] ka bajo is same but it depends if jo left and right hai of mid is even or odd let's take an example



in this case we can say ki single element will be on the side jidar either A[mid-1] or A[mid+1] is equal to A[mid] kyuki 4 elements se 1 is equal to mid, 2 are dupes so baki 1 is the single element



in this case we can say ki single element will NOT be on the side jidar either A[mid-1] or A[mid+1] is equal to A[mid] kyuki 3 elements se 1 is equal to mid, 2 are dupes so single element can't lie on the equal side it'll be on other side wala

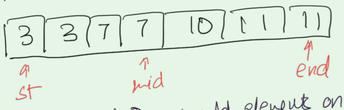
to solve this issue we will take mid % 2 == 0 and do for even number case and else for odd number case

Pseudocode

```
st = 0, end = n-1
while(st <= end) {
    mid = st + (end-st)/2;
    if(mid == 0 && A[0] != A[1]) return mid
    if (mid == n-1 && A[n-1] != A[n-2]) return mid
    if (A[m-1] != A[m] != A[m+1])
        return mid
    if mid % 2 == 0 {
        if A[mid-1] == A[mid] ⇒ search space is on the left
            end = mid-1
        else ⇒ search space is on the right
            st = mid+1
    } else {
        if A[mid-1] == A[mid]
            st = mid+1
        else
            end = mid-1
    }
}
```

} Handling edge cases if single element is on the first ya last position and mid gradually comes to that point

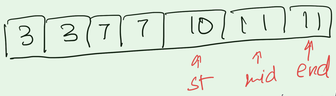
Dry run



mid % 2 ≠ 0 ∴ odd element on either side

A[mid-1] = A[mid]

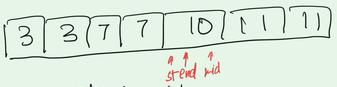
∴ st = mid + 1



mid % 2 ≠ 0 ∴ again odd element

A[mid-1] ≠ A[mid]

∴ end = mid - 1



return mid