Search in rotated sorted array

ascending order ; distinct values

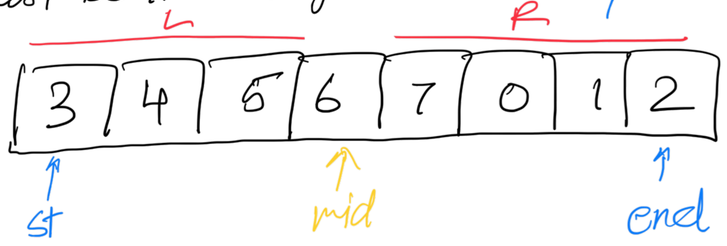
[3 4 5 6 7 0 1 2] tar = 0

given array (rotated)

[0 1 2 3 4 5 6 7]

original array after being rotated

must be in O(logn) TC ⇒ binary search



if you see one side of this array, one side WILL always be SORTED

abhi you know ki left side is sorted then use binary search apply karke we can find

RS Array

LS sorted

to know if its LS then

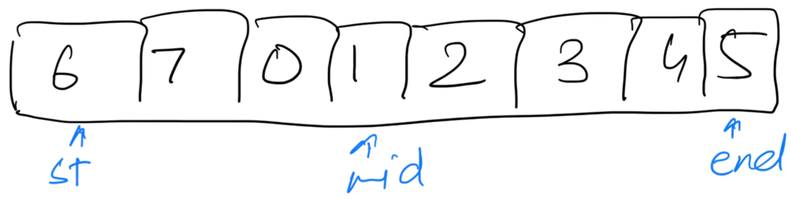
A[st] <= A[mid]

if yes then LS

L->BS (left side mein search karna)

RS sorted

else RS



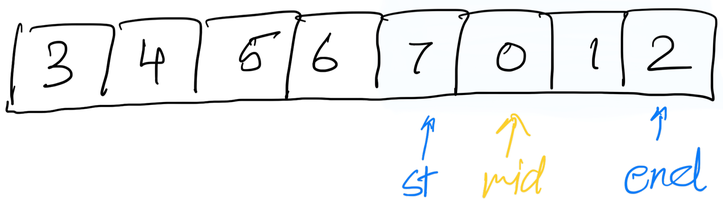
R-BS (right side mein search karna hai)

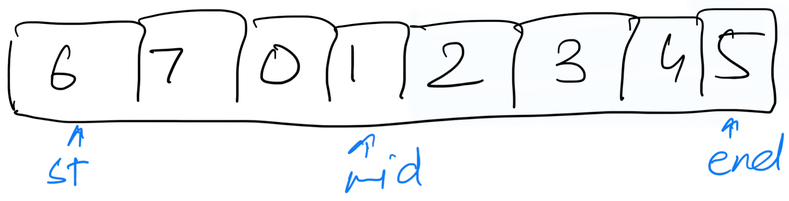
A[mid] <= tar <= A[end] ∴ (mid+1, end]

else ∴ [st, mid-1)

A[st] <= tar <= A[mid] left ∴ [st, mid-1)

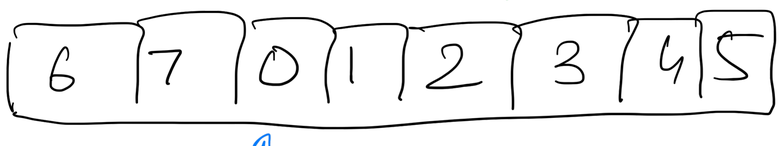
else right ∴ (mid+1, end]





Searching in right side but target doesn't exist so we go left side then we change the st & end pts

again we will find which side is sorted in this case left side sorted, we'll check the conditions again and move accordingly



Code

```
st=0, end=n-1;
while (st <= end) {
    mid = st + (end-st)/2;
    if (A[mid] == tar)
        return mid;
    if (A[st] <= A[mid]) { // left sorted
        if (A[st] <= tar <= A[mid]) ⇒ left ⇒ end = mid-1
        else ⇒ right ⇒ st = mid+1
    } else { // right sorted
        if (A[mid] <= tar <= A[end]) ⇒ right ⇒ st = mid+1
        else ⇒ left ⇒ end = mid-1
    }
}
```