Search in rotated sorted array

BRUTE FORCE :

linear search

- Time Complexity : O(N)
- Space Complexity : O(1)

Optimal Approach :

We use the sorted array to our benefit to reduce the time complexity from O(N) to O(log N) using binary search .

The idea is to find mid element and see which part of the array is sorted either of the part will be sorted. So we make a comparison to find which half is sorted. We then make a comparison to find whether our key is a part of this half if yes then we move our low and high pointer accordingly.

```
int search(int* arr, int n, int key) {
    // Write your code here.
    int low=0,high=n-1;
    while(low<=high)
    {
        int mid=low+(high-low)/2;
        if(arr[mid]==key)
            return mid;
        if(arr[mid]<arr[high])
        {
            if(key>=arr[mid]&& key<=arr[high])
            {
                low=mid+1;
            }
            else{
                high=mid-1;
            }
        }
        else
        {
            if(key>=arr[low]&&key<=arr[mid])
            {
                high=mid-1;
            }
            else
            {
                low=mid+1;
            }
        }
    }
    return -1;
}
```

- Time Complexity : O(log N)
- Space Complexity : O(1)