# Rotated and Sorted

# Brute force

* Idea: Rotate the array each time & then Compare it to the Sorted array. If the array after rotating equal sorted array, then the array was rotated

Eg: [ 3, 4, 5, 1, 2]

Ans: sorted array = [1 2 3 4 5]

① 4 5 1 2 3 = 1 2 3 4 5 ✗

② 5 1 2 3 4 = 1 2 3 4 5 ✗

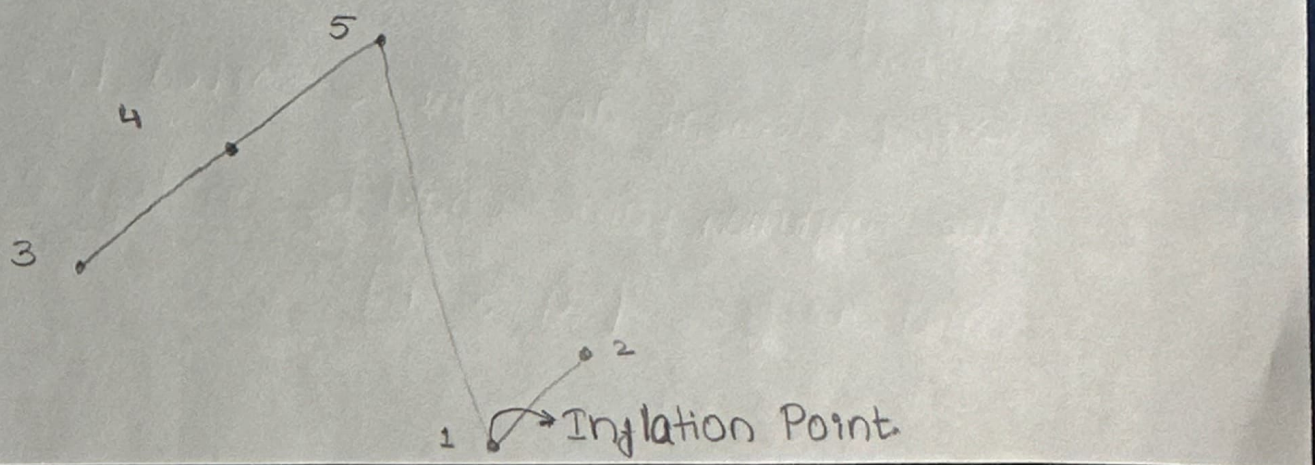
③ 1 2 3 4 5 = 1 2 3 4 5 ✓

So, we rotate by removing Element from front & attaching to the End.

# Inflation point

Inflation point is a point where the actual array got split to be rotated

Eg:



So, previously we rotated the array for every element & checked if the array was sorted.

Now we find the inflation point & the check if the array was sorted

There are 2 ways to find out the inflation point.

① Comparing with previous element

② Binary search

# Comparing with previous element:

[3 4 5 1 2]

3 > 4 ? → No

4 > 5 ? → No

5 > 1 → yes → Inflation point found

Rotate the array from here & check if the array is sorted.

[1 2][3 4 5] → 1 2 3 4 5 ✓

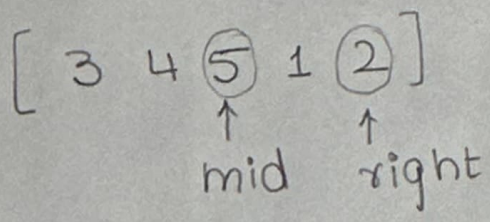
## ② Binary Search

We know that, when array is sorted, right > left.

[1 2 3 4 5] 5 > 1 → This will always be true

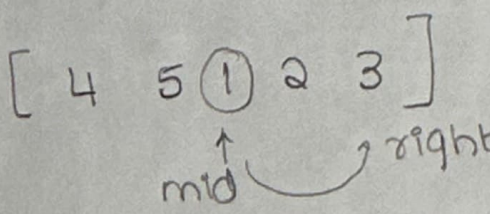
When the array gets rotated

Condition 1 : mid > right :




* If mid > right: 5 > 2
  - Every Element on left is sorted → [3 4 5]
  - So we are sure that the inflation point lies to the right of mid [1 2]

Condition 2 : mid < right




- Every Element on right is sorted [1 2 3]
- The inflation point could be on left or mid itself [4 5 1]

Eg [3 4 5 1 2]

* [3 4 5 1 2]
  ↑ ↑
  mid right

5 > 2 → So Inflation point will be to right of mid

Left = mid + 1

* [1 2]
  ↑ ↑
  mid right

1 < 2 → so Inflation point can be mid itself or to the left

right = mid

* [1] = Inflation point found
  ↑↑
  left right

→ rotate here and Check with Sorted array

[1 2 3 4 5] → 1 2 3 4 5

# Count Inflation point (optimal Solution)

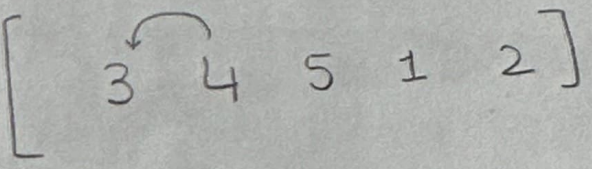
Until now we saw that we can only have 1 inflation point If there are more than One, then the array is not sorted.

→ Idea is to count the inflation point

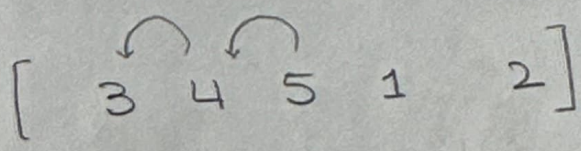
# We Count the inflation point, by comparing the Value with its previous Element

If we take an array, [ 3 4 5 1 2 ]

we know 4 comes after 3.



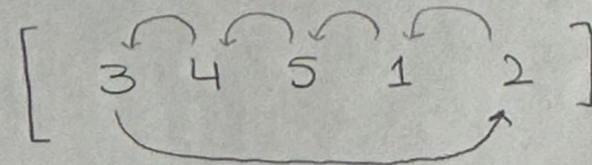

similarly 5 comes after 4




But 1 does not come after 5

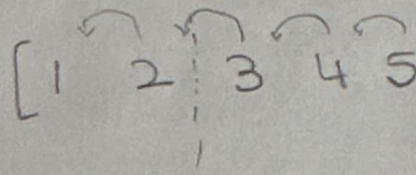
hence we found 1 inflation point

# Comparison



Why do we Compare 3 with 2

If we take a sorted array before rotation




→ The array got chopped at 2

So It became [ 3 4 5 1 2 ], so the Element that comes previous to 3 is 2.

# How do we do this mathematically

$$B[i] = A[i-1 \,\%\, n]$$

So when $i = 0$

$$B[0] = A[0-1 \,\%\, 5] = A[-1 \,\%\, 5] = A[-1]$$

$i=1$, $B[1] = A[1-1 \,\%\, 5] = A[0 \,\%\, 5] = A[0]$

$i=2$, $B[2] = A[2-1 \,\%\, 5] = A[1 \,\%\, 5] = A[1]$

$i=3$, $B[3] = A[3-1 \,\%\, 5] = A[2 \,\%\, 5] = A[2]$

$i=4 = B[4] = A[4-1 \,\%\, 5] = A[3 \,\%\, 5] = A[3]$

So