Array Rotation Solution with formula

As we know, how we translate 1-d array index into 2-d array index

row_index = (1d_index) / 2d_arr[0].length
col_index = (1d_index) % 2d_arr[0].length

And vice versa: 1d_index = row_index * 2d_arr[0].length + col_index

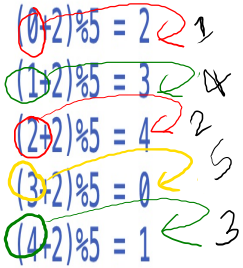
Post right rotation array will follow the col_index formula as it is similar to occupy space in next row at kth colum.

```
(i+k) % n = col_index
(i+k) % n = r

(0+k)%n = k
(1+k)%n = k+1
(2+k)%n = k+2
. .. ... ..
(x+k)%n = 0; when x+k = n
. .. ... ..
(n-1+k)%n = k-1;
```

```
formula : r =  (i+k)%n
i - represents index of array
k - represents rotation count
n - represents size of the array
r - represents index post rotation
```

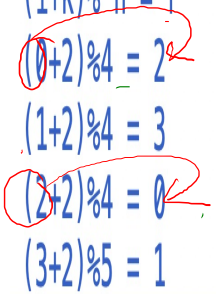
CASE_A: odd sized array
when nums length is odd, lets say array size is 5 and k=2
(i+k)% n = r



Here,
0th index --translates to --> 2th index --> 4th index--> 1th index --> 3rd index--> 0th index

So, there is full transitive iteration possible

CASE_B: even sized array
when nums length is even, lets say array size is 4 and k=2
(i+k)% n = r



Here, there is no full transitive iteration, rather broken transitive iteration are formed.

This method works by rotating an array in-place by using cyclic replacements. This approach involves repeatedly moving elements to their correct positions one step at a time until all elements are in their correct positions.

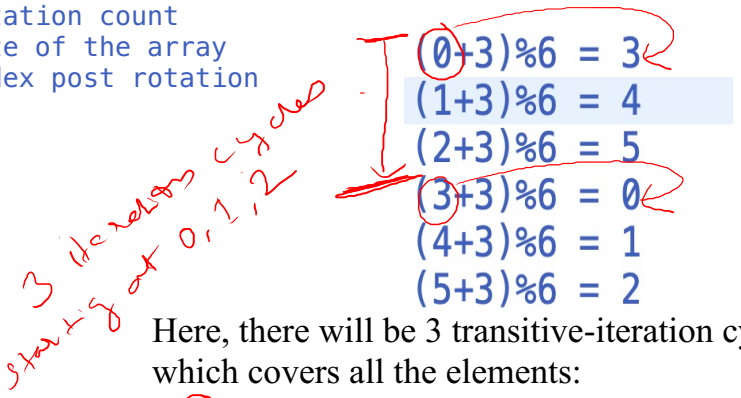
For odd-sized array transitive-iteration covers all the elements of the array. It starts from the 0th index and ends at 0th index. But for even-sized array there going to be multiple transitive-iterations . And we need to have a way for going to next transitive.

So, in this algorithm we will iterate until all the elements are moved to its correct position with two loops, outer loop will help starting next transitive iteration(to tackle the case of even-array where we have broken transitive iterations).
Inner loop will perform the actual transitive iteration.

Time complexity: O(n), because loop is getting iterated only once, that is we scan each element once combined inner+outer loop.

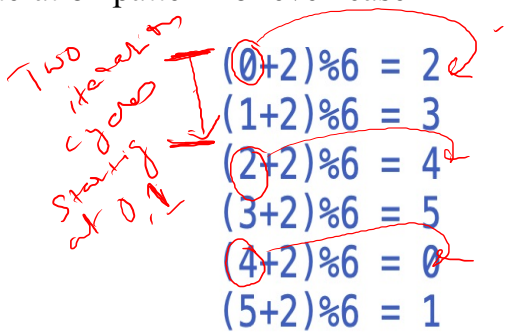
```
formula : r =  (i+k)%n
i - represents index of array
k - represents rotation count
n - represents size of the array
r - represents index post rotation
```

Let's try to figure-out iteration pattern for even case



Here, there will be 3 transitive-iteration cycles which covers all the elements:
1. 0 -> 3 : 3 -> 0
2. 1 -> 4 : 4 -> 1
3. 2 -> 5 : 5 -> 2

Note: Iteration cycle start nodes are sequential. 0,1,2



Here, there will be 2 transitive-iteration cycles which covers all the elements:
1. 0 -> 2 : 2 -> 4 : 4 -> 0
2. 1 -> 3 : 3 -> 5 : 5 -> 1

Note: Iteration cycle start nodes are sequential. 0,1

So, finally for even-sized-array outer loop needs to start the transitive-iteration clycles in sequence until we reach to the total count of elements. Since, for odd-sized-array there will be only single transitive-iteration cycle, means we need to put the total count calculation for inner loop as well.

```java
  public static void rotate(int[] nums, int k) {
      int n = nums.length;
      k %= n; // to handle cases where k is greater than the length of the array
      int count = 0;
      for (int startIndex = 0; count < n; startIndex++) {
          count = iterateTransitiveCycle(nums, n, k, startIndex, count);
      }

  }
```

```java
/**
 * A transitive cycle ends when it reaches the same index where it started.
 *
 */
private static int iterateTransitiveCycle(int[] nums, final int n, final int k, final int startIndex, int count) {

    int idx = startIndex;
    int nextIdx = Integer.MIN_VALUE;

    int prevElement = nums[idx];

    while (nextIdx != startIndex) {

        nextIdx = (idx + k) % n;

        // rotate the element
        int temp = nums[nextIdx];
        nums[nextIdx] = prevElement;
        prevElement = temp;

        idx = nextIdx;
        count++;
    }

    return count;

}
```