Arrays

Java

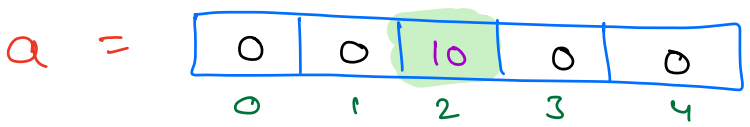
Syntax

```
int [] a = new int[5]
```

Python

Syntax

```
a = [0] * 5
```

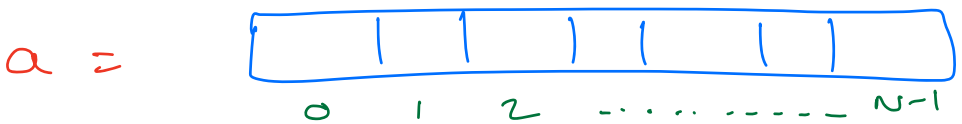


a[2] = 10

How to declare an array of size N ?

```
int [] a = new int[N];
```

```
a = [0] * N
```



Passing array to functions

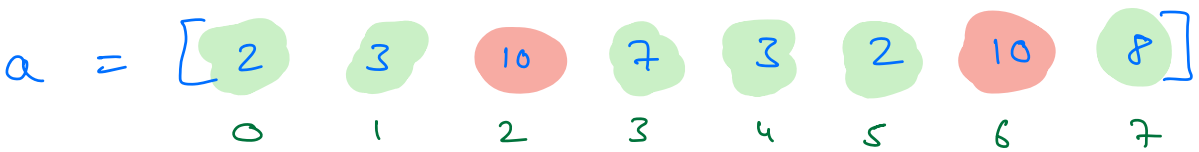
```
void printArr(int[] a){
    int n = a.length;
    for(int i=0; i<n; i++){
        sop(a[i]);
    }
}
```

```
def printArr(a):
    n = len(a)
    for i in range(n):
        print(a[i])
```

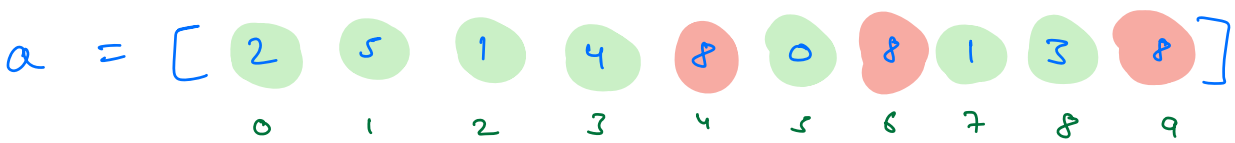
Q1.

Given N array elements , count no. of elements having atleast 1 element greater than itself.

↳ 5



↳ 6



↳ 7

Observations

Obs 1 :

For every max element, there won't be any element greater than it.

Obs 2 :

For every other element, there will atleast one element greater than it.

Pseudo code

1) Iterate & get the max of array.

2) Iterate & get count of elements != max.

```
int solve(int arr[]){
    // Find the max
    maxVal = -∞
    for (i=0; i<n; i+=1)
        if ( maxVal < arr[i])
            maxVal = arr[i]

    count = 0
    for (i=0; i<n; i+=1)
        if (arr[i] != maxVal)
            count += 1

    return count
}
```

Java

```java
int solve(int []a) {
    int n = a.length;
    int maxValue = Integer.MIN_VALUE;
    for (int i = 0; i < n; i++) {
        if (maxValue < a[i])
            maxValue = a[i];
    }

    int c = 0;
    for (int i = 0; i < n; i++) {
        if (maxValue != a[i]) {
            c += 1;
        }
    }
    return c;
}
```

Python

```python
def solve(a):
    n = len(a)
    maxValue = -float('inf')
    for i in range(n):
        if maxValue < a[i]:
            maxValue = a[i]

    c = 0
    for i in range(n):
        if maxValue != a[i]:
            c += 1

    return c
```

N

N

Total iterations = N + N = 2N

Time - O(N)

Space - O(1)

Extra Space

Q2 Given N array elements, check (bool) if there exists a pair (i, j) such that

$$arr[i] + arr[j] == k \quad \&\& \quad i != j$$

Note : i & j are index values , k is given sum

a = [3 -2 1 4 3 6 8]
0 1 2 3 4 5 6

k = 10

i = 3, j = 5 → 4 + 6 = 10

⇒ True

a = [2 4 -3 7]
0 1 2 3

k = 5

No pair ⇒ False

$a = [\ 2 \quad 4 \quad -3 \quad 7\ ]$

$0 \quad 1 \quad 2 \quad 3$

$k = 8$

~~$4 + 4$~~

No pairs

$\Rightarrow$ False

Idea 1

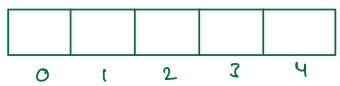
→ Run a loop with i over all indices
→ Run a nested loop with j over all indices

Time - O(N^2)
Space - O(1)

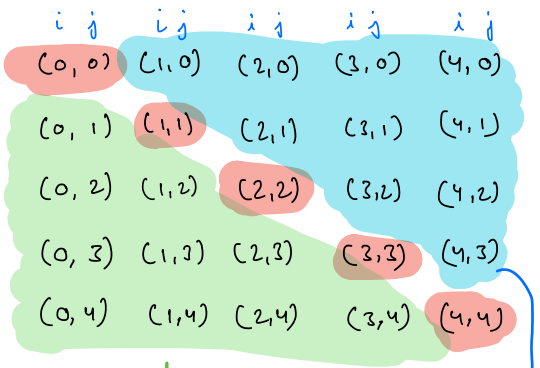
```
boolean checkSum(int arr[], int k) {
    n = arr.length
    for (i=0; i<n; i++) {
        for(j=0; j<n; j++) {
            if(arr[i] + arr[j] == k  and  i != j)
                return True
        }
    }
    return False
}
```

arr[i] + arr[j] = arr[j] + arr[i]

a =



All pairs:



Idea 2

```
boolean checkSum(int arr[], int k) {
    n = arr.length
    for (i=0; i<n; i++) {
        for (j=0; j<i; j++) {
            if( arr[i] + arr[j] == k)
                return True
        }
    }
    return false
}
```

i	j	Iterations
0	0 → 0	0
1	0 → 1	1
2	0 → 2	2
3	0 → 3	3
⋮		
N-1	0 → N-1	N-1

Time - O(N^2)
Space - O(1)

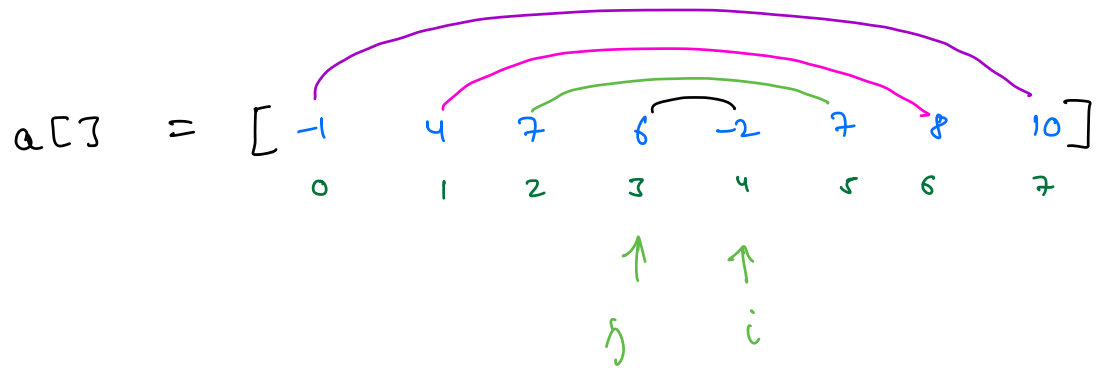
Total iterations = 0 + 1 + 2 + ... + N-1

= $\frac{N(N-1)}{2}$

Q3 Given an array , reverse the entire array

Note : Array itself should change

Expected SC: O(1)



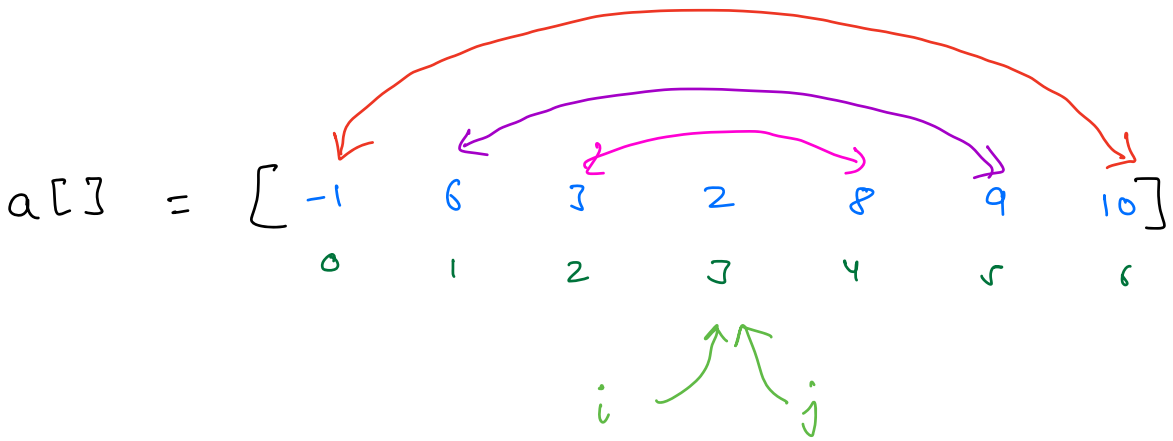
Final res [10 8 7 -2 6 7 4 -1]

0 1 2 3 4 5 6 7

x = 5

y = 2

x, y = y, x



Swap a[0] and a[6]

Swap a[1] and a[5]

Swap a[2] and a[4]

Swap(a[i], a[j])

i += 1

j -= 1

Keep on swapping till

$i < j$

Pseudo code

```
reverseArray(int arr[]) {
        n = arr.length
        i=0 , j = n-1
        while ( i < j ) {
            swap( a[i], a[j])
            i +=1
            j -=1
        }
}
```

Java

```java
void reverse(int []a) {
    int i = 0;
    int j = a.length - 1;
    while (i < j) {
        int temp = a[i];
        a[i] = a[j];
        a[j] = temp;
        i++;
        j--;
    }
}
```

Python

```python
def reverse(a):
    i = 0
    j = len(a) - 1

    while i < j:
        a[i], a[j] = a[j], a[i]
        i += 1
        j -= 1
```

Time - O(N)

Space - O(1)

i → 0 to $\frac{N}{2}$

j → N-1 to $\frac{N}{2}$

Total no of iterations = $\frac{N}{2}$

Q4

Given an array , and [S & E],
reverse the array from [S E], where S and E are indices.
Note : S <= E

a[] = [-3 4 2 8 7 9 6 2 10]
0 1 2 3 4 5 6 7 8

S E
[3 7]

[-3 4 2 2 6 9 7 8 10]

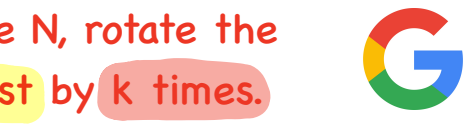
Worst case , s = 0 , e = n-1
→ Previous Problem

```
reversePart(int arr[], int s, int e) {
    i = s, j = e
    while (i < j) {
        swap (a[i], a[j])
        i += 1
        j -= 1
    }
}
```

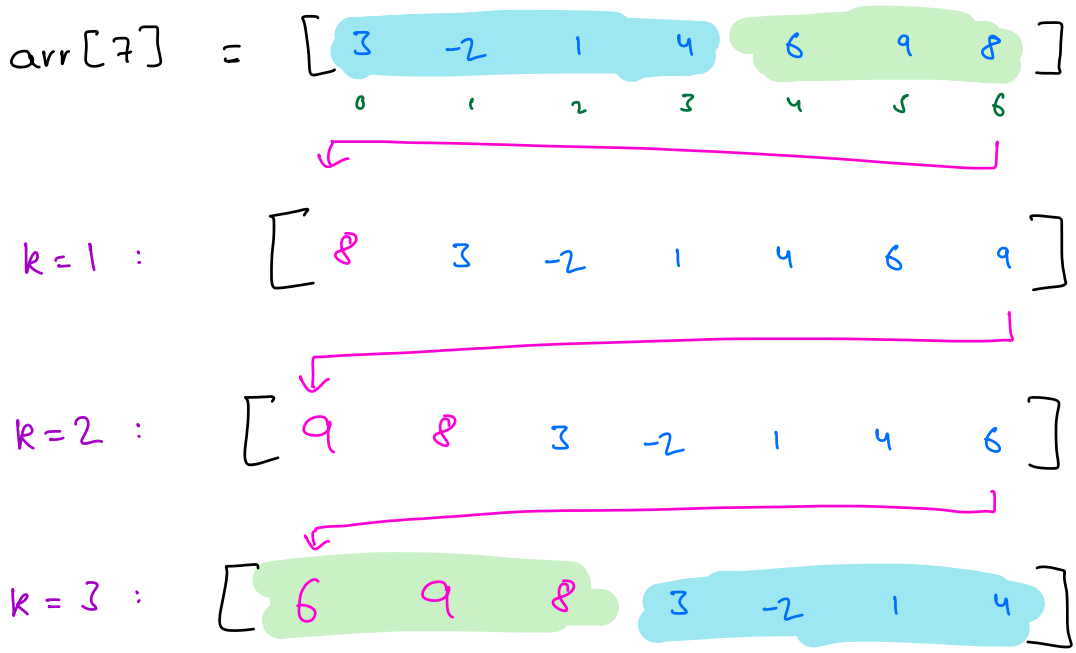
Time - O(N)
Space - O(1)

Break till 10:30 AM

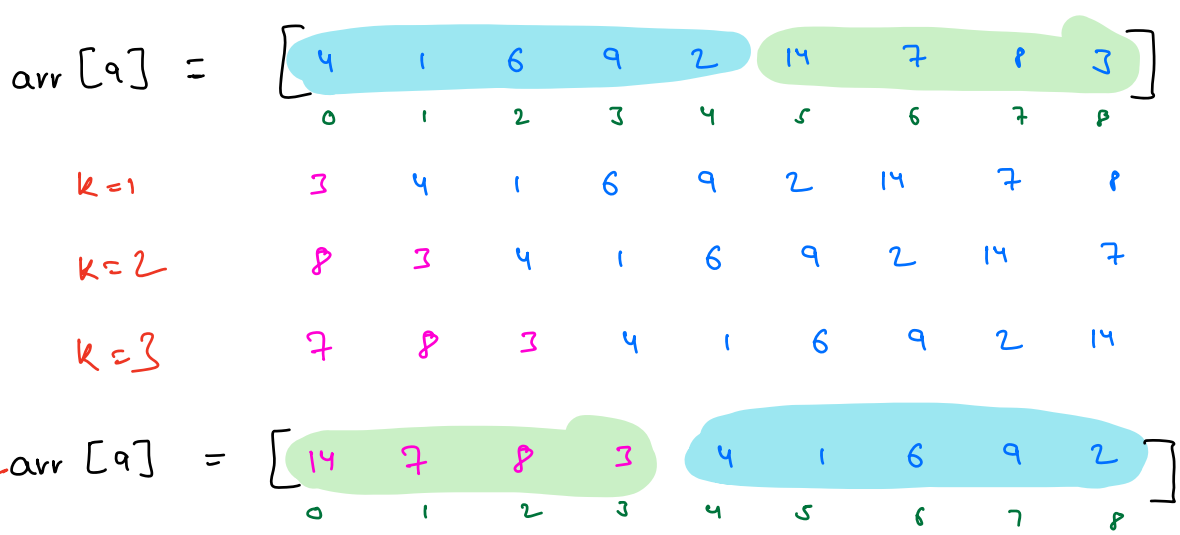
Q5. Given an array of size N, rotate the array from last to first by k times.



Expected SC: O(1)



Example



Ans arr[9] = [14 7 8 3 4 1 6 9 2]

0 1 2 3 4 5 6 7 8

arr [13] = [a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12}]

Rotate by 5 times :

$k = 5$

[a_8 a_9 a_{10} a_{11} a_1 a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7]

5 elements 8 elements

k elements (n-k) elements

---

arr [13] = [a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12}]

Reverse the entire array

a_{12} a_{11} a_{10} a_9 a_8 a_7 a_6 a_5 a_4 a_3 a_2 a_1 a_0

Reverse first k elements

Reverse last (n-k) elements

[a_8 a_9 a_{10} a_{11} a_1 a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7]

1) Reverse entire array ✓ $s=0, e=n-1$

2) Reverse first k elements $s=0, e=k-1$

3) Reverse last (n-k) elements $s=k$, $e=n-1$

```
rotateTimes(int arr[], int k) {
        n = arr.length
        reversePart (arr, 0, n-1)
        reversePart (arr, 0, k-1)
        reversePart (arr, k, n-1)
}
```

Java

```java
class RotateKTimes {
  static void reverse(int[] arr, int start, int end) {
    while (start < end) {
      int temp = arr[start];
      arr[start] = arr[end];
      arr[end] = temp;
      start++;
      end--;
    }
  }

  static void rotate_k_times(int []A, int k) {
    int n = A.length;

    // Reverse entire array
    reverse(A, 0, n-1);

    // Reverse first k elements
    reverse(A, 0, k-1);

    // Reverse last n-k elements
    reverse(A, k, n-1);
  }

  public static void main(String[] args) {
    int []a = {2, 5, 7, 1, 8, 9, 2, 0, 2, 6, 7, 10};
    int k = 4;

    rotate_k_times(a, k);
    for (int x: a) {
      System.out.print(x + " ");
    }
  }
}
```

Python

```python
def reverse(A, start, end):
    while start < end:
        A[start], A[end] = A[end], A[start]
        start += 1
        end -= 1


def rotate_k_times(A, k):
    n = len(A)

    # Reverse entire array
    reverse(A, 0, n - 1)

    # Reverse first k elements
    reverse(A, 0, k - 1)

    # Reverse last n-k elements
    reverse(A, k, n - 1)


def main():
    a = [2, 5, 7, 1, 8, 9, 2, 0, 2, 6, 7, 10]
    k = 4
    rotate_k_times(a, k)
    print(a)


main()
```

N ← N

k ← k

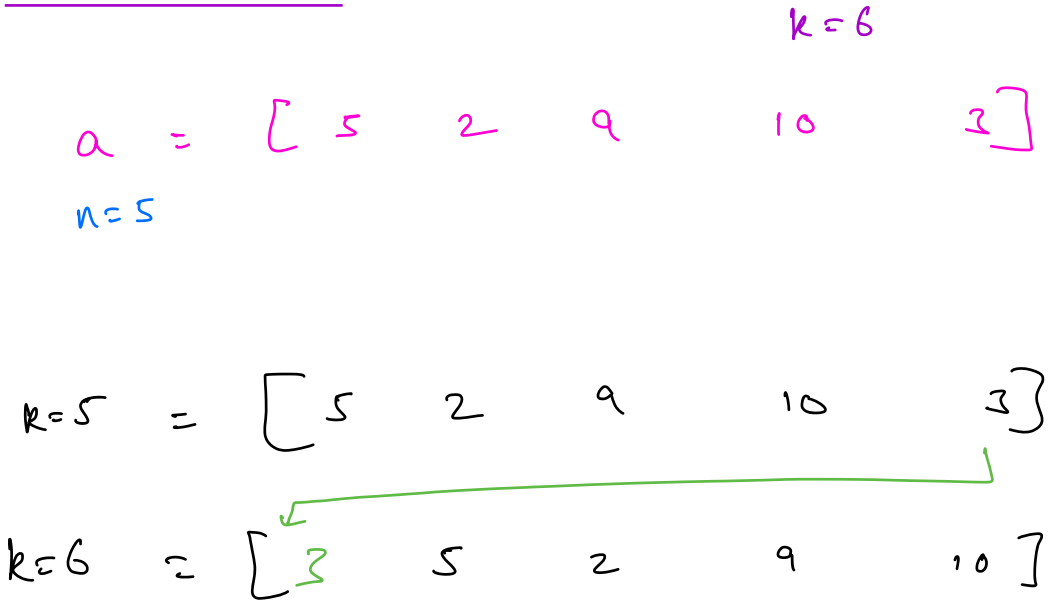
n-k ← n-k

Total = N + k + (N - k) = 2N

Time - O(N)

Space - O(1)

What if k > N ?



ar[6] = a_0 a_1 a_2 a_3 a_4 a_5

n = 6

		Rotate k times		
k=0	a_0 a_1 a_2 a_3 a_4 a_5	6	12	18
k=1	a_5 a_0 a_1 a_2 a_3 a_4	7	13	19
k=2	a_4 a_5 a_0 a_1 a_2 a_3	8	14	20
k=3	a_3 a_4 a_5 a_0 a_1 a_2	9	15	⋮
k=4	a_2 a_3 a_4 a_5 a_0 a_1	10	16	
k=5	a_1 a_2 a_3 a_4 a_5 a_0	11	17	

$k = k \% n$

```
rotateTimes(int arr[], int k) {
    n = arr.length
    k = k % n
    reversePart (arr, 0, n-1)
    reversePart (arr, 0, k-1)
    reversePart (arr, k, n-1)
}
```

Java

```java
class RotateKTimes {

  static void reverse(int[] arr, int start, int end) {
    while (start < end) {
      int temp = arr[start];
      arr[start] = arr[end];
      arr[end] = temp;
      start++;
      end--;
    }
  }

  static void rotate_k_times(int []A, int k) {
    int n = A.length;
    k = k % n;

    // Reverse entire array
    reverse(A, 0, n-1);

    // Reverse first k elements
    reverse(A, 0, k-1);

    // Reverse last n-k elements
    reverse(A, k, n-1);
  }
  public static void main(String[] args) {
    int []a = {2, 5, 7, 1, 8, 9, 2, 0, 2, 6, 7, 10};
    int k = 4;

    rotate_k_times(a, k);
    for (int x: a) {
      System.out.print(x + " ");
    }
  }
}
```

Python

```python
def reverse(A, start, end):
    while start < end:
        A[start], A[end] = A[end], A[start]
        start += 1
        end -= 1


def rotate_k_times(A, k):
    n = len(A)
    k = k % n

    # Reverse entire array
    reverse(A, 0, n - 1)

    # Reverse first k elements
    reverse(A, 0, k - 1)

    # Reverse last n-k elements
    reverse(A, k, n - 1)


def main():
    a = [2, 5, 7, 1, 8, 9, 2, 0, 2, 6, 7, 10]
    k = 4
    rotate_k_times(a, k)
    print(a)


main()
```

Doubts

Thank You

$$2^N \times 2^N$$

$$= 2^{2N} = 4^N$$

$$a^b \times a^c = a^{b+c}$$

$$a^{b*c} = (a^b)^c$$

$$2^{2N} = (2^2)^N = 4^N$$

Good Night

Thank You

Friday