int arr[5];

int → Type

5 → Size of the array.

0	1	2	3	4
4	3	2	5	6

Quiz-1

int arr[N]

0	1	2	3	- - - - - - . . .			N-2	N-1
			10					

Quiz-2

Index of first element ⇒ 0

Index of last element ⇒ N-1

Print the array elements :-

```
for(i=0; i<N; i++){
        print(arr[i]);
}
```

O(N)

Quiz-3 :

Access TC of arr[i] ⇒ O(1)

⇒ Arrays are most frequently used data structure because it provides O(1) access & update TC.

Q: Given N array elements, count the no. of elements having at least one element greater than itself.

	0	1	2	3	4	5	6	
A[7]:	{ -3,	-2,	6,	8,	4,	8,	5 }	⇒ 5
	✓	✓	✓	×	✓	×	✓	

Quiz

N=10

A:	{ 2,	5,	1,	4,	8,	0,	8,	1,	3,	8 }
	✓	✓	✓	✓	×	✓	×	✓	✓	×

⇒ 7

count(8) = 3

ans = 10 - 3 = 7

Observations

1) for max element, there won't be any element greater than that.

2) Get the count / frequency of max element
   = count

3) return N - count;

1) Find max of the array.

```
max = INT_MIN
for ( i= 0; i< N; i++) {
        if(a[i] > max)
            max = a[i]
}
```

N

2) Get the freq. of max

```
count = 0
for ( i= 0; i< N; i++) {
        if(a[i] == max)
            count ++
}
```

N

3) return N- count;

# of iterations = 2N

TC : O(N)

SC : O(1) {Constant Extra Space}

Todo :- Try to do in Single loop. => 30 mins

Amazon
VmWare
⋮

Q.2 Given N array elements, check if there exists a pair i, j s.t arr[i] + arr[j] = k & i != j. i & j are indexes & k is the target sum. (TWO SUM)

	0	1	2	3	4	5	6
A[]:	{ 3,	-2,	1,	4,	3,	6,	8 }

k = 10

Sum = 10

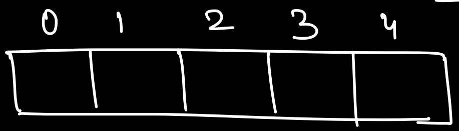
→ True

A[]: { 2 4 -3 7 } k = 8

→ false

Idea 1 :- Check all possible pairs.

N=5



Upper triangular

i, i+1

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)

lower triangular matrix

```
for ( i = 0; i < N; i++ ){
    for( j = 0; j < N; j++ ) {
        if( a[i] + a[j] == k && i != j ){
            return true;
        }
    }
}
return false;
```

# of iterations = N^2

TC : $O(N^2)$

SC : $O(1)$

a[i] + a[j] = a[j] + a[i]

```
                     N-1
for ( i = 0; i < N; i++ ){
    for( j = i+1; j < N; j++ ) {
        if( a[i] + a[j] == k && i != j ){
            return true;
        }
    }
}
return false;
```

i	j	# of iteration
0	[1, N-1]	N-1
1	[2, N-1]	N-2
2	[3, N-1]	N-3
⋮	⋮	⋮
N-2	[N-1, N-1]	1
N-1	[N, N-1] ×	0

$$\underbrace{1 + 2 + 3 + \cdots + (N-1)}_{\text{Sum of (N-1) natural no's}}$$

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

TC: $\underline{\underline{O(N^2)}}$

SC: $\underline{\underline{O(1)}}$

Q. Given an Array, Reverse the entire array.

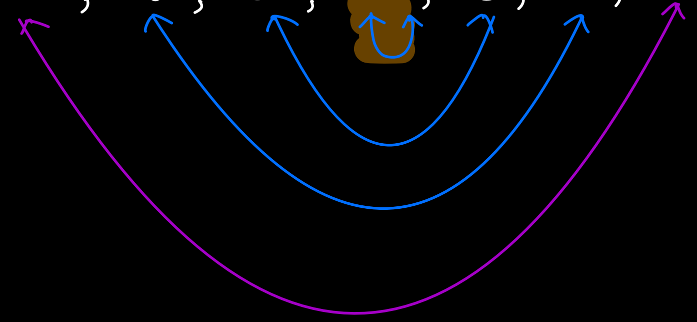
Amazon
MS

{Expected SC : O(1)}

$$\begin{array}{lcccccccc} & & & & e & s & & & \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A[8]: & \{-1, & 4, & 7, & 6, & -2, & 7, & 8, & 10\} \end{array}$$

$\Rightarrow \{10, 8, 7 \quad -2 \quad 6 \quad 7 \quad 4 \quad -1\}$

$$\begin{array}{lccccccc} & & & & se & & & \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A[7] = & -1, & 6, & 3, & 2, & 8, & 9, & 10 \end{array}$$



s		e
0	⟷	6
1	⟷	5
2	⟷	4
3	⟷	3

$$\begin{array}{lcccccccc} & s & & & & & & & e \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ A[8]: & \{-1, & 4, & 7, & 6, & -2, & 7, & 8, & 10\} \end{array}$$

s = 0, e = 7 ⇒ Swap
s = 1, e = 6 ⇒ Swap
s = 2, e = 5 ⇒ Swap
s = 3, e = 4 ⇒ Swap
s = 4, e = 3 ⇒ Break the While loop.

$$\begin{array}{lccccccc} & & & & s,e & & & \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ A[7] = & -1, & 6, & 3, & 2, & 8, & 9, & 10 \end{array}$$

s = 0, e = 6 ⇒ Swap
s = 1, e = 5 ⇒ Swap
s = 2, e = 4 ⇒ Swap
s = 3, e = 3 × × × Break

```
void  reverse ( A[], N){
      Start = 0
      end = N-1
       while( Start < end ){
             Swap (a[start], a[end]);
             Start++;
             end--;
       }
}
```

Todo Implement swap without using temp variable.

$\#$ of iterations $= \frac{N}{2}$

TC: O(N)
SC: O(1)

Q: Given an Array & [S_i & S_j], Reverse array from S_i to S_j [S_i <= S_j]

	0	1	2	3	4	5	6	7	8
A:	-3	4	2	8	7	9	6	2	10

$S_i = 3$
$S_j = 7$

A: -3	4	2	2	6	9	7	8	10

```
void  reversePart(A[] , N ,Si , Sj ){
        Start  = Si
        end = Sj
        while( Start < end ){
                Swap (a[start] , a[end]);
                Start++;
                end--;
        }
}
```

$[S_i, S_j] \Rightarrow S_j - S_i + 1$

$\# \text{ of iterations } = \dfrac{S_j - S_i + 1}{2}$

Worst Case : $S_i = 0, \; S_j = N-1$

$\# \text{ of iter } = \dfrac{N}{2}$

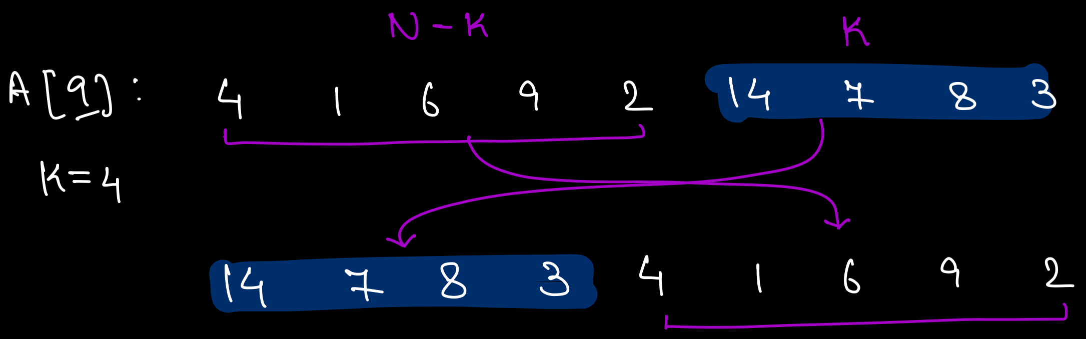
TC: $O(N)$

SC : $O(1)$

Q: Given N array elements, Rotate the array from last to first by K times.

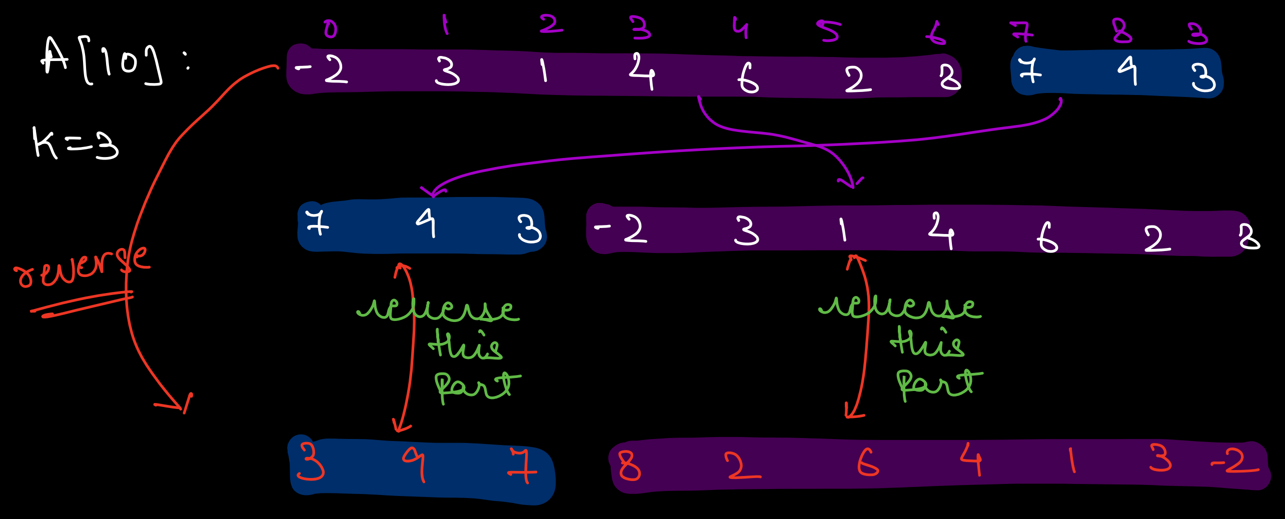
{SC : O(1)}

	0	1	2	3	4	5	6
A[7] :	3	-2	1	4	6	9	8
K=1	8	3	-2	1	4	6	9
K=2	9	8	3	-2	1	4	6
K=3	6	9	8	3	-2	1	4
K=4	4	6	9	8	3	-2	1

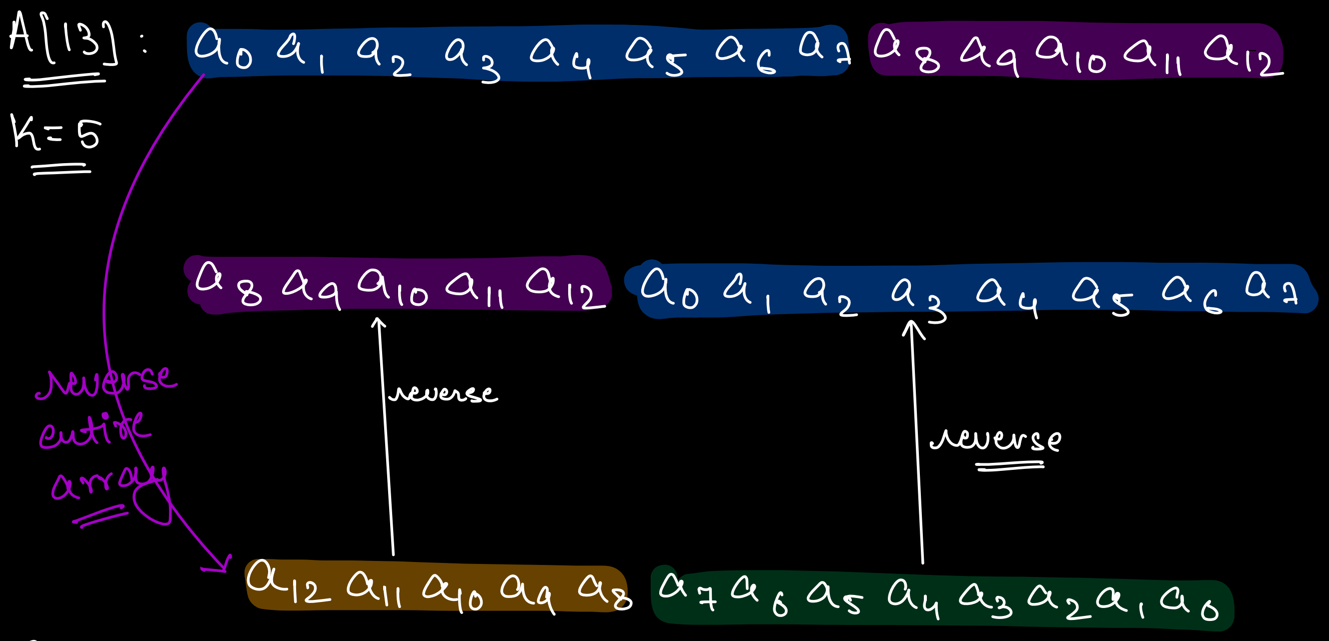


⇒ last 4 elements ⇒ first 4 elements after rotation.

⇒ first 5 elements ⇒ last 5 elements after rotation.



General


A[13] :
K= 5
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}
a_8 a_9 a_{10} a_{11} a_{12} a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7
reverse entire array
reverse
reverse
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

Steps

1) Reverse the entire array
2) Reverse first K elements
3) Reverse last N-K elements.

Pseudo code

1) reversePart(arr, N, $\overset{Si}{0}$, $\overset{Sj}{N-1}$) ⇒ $N/2$

2) reversePart(arr, N, 0, K-1) ⇒ $K/2$

3) reversePart(arr, N, K, N-1) ⇒ $\frac{N-K}{2}$

Total no. of iterations $= \frac{N}{2} + \frac{K}{2} + \frac{N-K}{2}$

$= N$

$$\left\{ \begin{array}{l} TC : O(N) \\ SC : O(1) \end{array} \right\}$$

K=N

1) reversePart(arr, N, 0, N-1)

2) reversePart(arr, N, 0, N-1)

3) reversePart(arr, N, N, N-1)

K>N

↳ Array Index Out of bound Exception

A[6]:	a_0	a_1	a_2	a_3	a_4	a_5	N=6
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, 21
k=4	a_2	a_3	a_4	a_5	a_0	a_1	10, 16, 22
k=5	a_1	a_2	a_3	a_4	a_5	a_0	11, 17, 23
k=6	a_0	a_1	a_2	a_3	a_4	a_5	
k=7	a_5	a_0	a_1	a_2	a_3	a_4	

```
if (k >= N)
    k = k % N
```

Quiz

K=9

A : $\{a_0\ a_1\ a_2\ a_3\ a_4\ a_5\}$ N=6

$K > N \Rightarrow K \% N$

$= 9 \% 6 = \boxed{3}$

$\{a_3\ a_4\ a_5\ a_0\ a_1\ a_2\}$

# int arr[5];
↳ 5 size
int arr[N]; ⇒ N elements

Static Array

Dynamic Array

C++	:	vector
Java	:	Array List
Python	:	list
C#	:	Array List
JS	:	array
C	:	Upgrade to C++

⇒

vector<int> v;	list<int> l;
v.push_back(10)	l.insert(10)
v.push_back(20)	l.insert(20)
v.push_back(30)	l.insert(30)

v.size() ⇒ $\bigcirc\!\!\!\!\!3$ l.size() ⇒ 3

TC of insert = O(1)