Arrays :

Linear collection of similar type of data.

arr [5] =

0	1	2	3	4
1	3	7	8	10

arr [0] + arr [3] = 9

TC of accessing an element ⇒ O(1)

Q Print all elements of an array with length n.

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

TC: O(n)

Q1 Given n array elements, count number of elements having atleast 1 element greater than itself. $n \geq 1$. Values in array are int.

Note: Sorting is not allowed!

Ex1 arr[7] = {-1 2 4 6 5 6 2}

→ ans = 5

Ex2 arr[6] = {1 9 14 22 4 6}

→ ans = 5

Observation

1) Max element would not be included in the count.

Approach:
1) Find maximum
2) Find freq of maximum
3) ans ⇒ size of array - freq of max.

Pseudo Code :

1) Find maximum.

```
int max_val = arr[0] / Int.Min
for (int i=0 ; i<n ; i++) {
        if (arr[i] > max_val)
                max_val = arr[i];
}
```

→ arr[0]
→ Int.Min
→ 0

Minimum value that an int can take.

→ $O(n)$

2) Find frequency of maximum

```
int c = 0;
for (int i=0 ; i<n ; i++) {
        if (arr[i] == max_val)
                c++;
}
```

→ $O(n)$

3) return ans:

```
return (n - c)
```

TC: $O(n)$

SC: $O(1)$

Q2 Given n array elements, print true if there exists a pair of index i & j, where arr[i] + arr[j] = k.

Note
1) array is not sorted
2) i != j
3) return true / false
4) k is given.

# Sorting
# O(nlogn)
# Hashmap
O(n)

Ex1 arr[ ] = { 3 5 -1 6 4 }

0	1	2	3	4
3	5	-1	6	4

K=8

→ true

Ex2 arr[ ] = { 2 4 3 -2 -7 }

0	1	2	3	4
2	4	3	-2	-7

K=8

→ false

k = 8

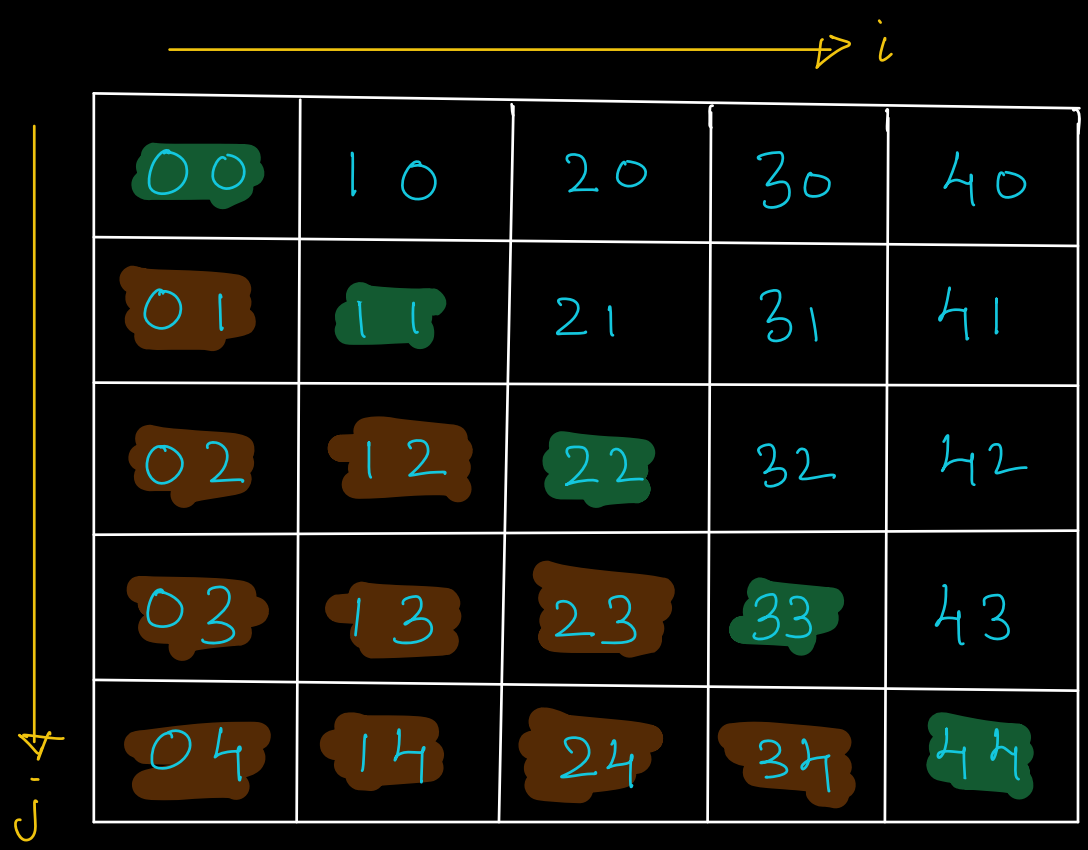
4	-1	-2	-3

Pseudo Code

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

Tc: $O(n^2)$

SC: $O(1)$.



```
for (int i=0 ; i<n ; i++) {
    for (int j= i+1 ; j<n ; j++) {
        if (arr[i] + arr[j] == k)
            return true;
    }
}
return false;
```

TC: $O(n^2)$

SC: $O(1)$

i	j	total
0	[1, n-1]	n-1
1	[2, n-1]	n-2
2	[3, n-1]	n-3
		⋮
n-1	[n-2, n-1]	0

$0 + 1 + 2 + 3 + 4 \ldots (n-1)$

Q3 Given an array. Reverse it.

SC should be O(1).

Ex1 arr[] = { 2 4 1 5 6 8 3 }

0	1	2	3	4	5	6
2	4	1	5	6	8	3

↓

0	1	2	3	4	5	6
3	8	6	5	1	4	2

Pseudo Code

```
i = 0, j = n-1
while ( i<j ) {
    // swapping.
    int temp = arr[i];
    arr [i] = arr [j];
    arr [j] = temp
    i++; j--;
}
```

TC: O(n)
SC: O(1)

i<j ✓

i<n/2 ✓

Even array

0	1	2	3	4	5
2	4	6	7	1	8

i

4n

i<3

$\sqrt{n} = n^{1/2}$

10:30pm

Q4 Reverse the array from start to end.

→ start & end are array indices and will given as input

→ start < end.

Ex1 : arr[] = { -2 4 6 7 8 1 5 }

0	1	2	3	4	5	6
-2	4	6	7	8	1	5

start = 2

end = 5

↓

{ -2 4 1 8 7 6 5 }

```
void reverse (int start, int end, int arr[]) {
    int i = start; int j = end;
    while (i<j) {
        // swap
        i++; j--;
    }
}
```

TC: O(n)

SC: O(1)

Q5 Given n array elements, rotate the array from last to first k times.

SC should be O(1).

$k < n$.

Ex1

	0	1	2	3	4	5	6
arr[] =	5	2	3	-1	6	1	2
k = 3							
k = 1	2	5	2	3	-1	6	1
k = 2	1	2	5	2	3	-1	6
k = 3	6	1	2	5	2	3	-1

0	1	2	3	4	5	6
5	2	3	-1	6	1	2

reverse full array.

2 1 6 -1 3 2 5

reverse 2nd part.

[k, n-1]

2 1 6 5 2 3 -1

reverse 1st part

[0, k-1]

6 1 2 5 2 3 -1

```
void rotate(int arr[], int k) {
    reverse(0, n-1, arr);
    reverse(k, n-1, arr);
    reverse(0, k-1, arr);
}
```

TC: O(n)
SC: O(1)

$\underline{k \geq n}$ $\qquad k = k \% n$

$$add[5] = a_1\, a_2\, a_3\, a_4\, a_5$$

$k = 0, 5, 10$	$a_1\ a_2\ a_3\ a_4\ a_5$
$k = 1, 6, 11$	$a_5\ a_1\ a_2\ a_3\ a_4$
$k = 2, 7, 12$	$a_4\ a_5\ a_1\ a_2\ a_3$
$k = 3, 8, 13$	$a_3\ a_4\ a_5\ a_1\ a_2$
$k = 4, 9, 14$	$a_2\ a_3\ a_4\ a_5\ a_1$

$$\boxed{k = k \% n}$$

3

Static Arrays : Size is fixed

```
int arr[5];
```

Dynamic Arrays ⇒ Size is not fixed.

Pseudo Code

```
list<int> l;
l.push(10);
l.push(11);
```

TC of accessing an element ⇒ O(1)

C++ ⇒ vector

Java ⇒ Array list

Python ⇒ list

JS ⇒ array.

C# ⇒ Array list

C ⇒ Change their language

Algo → Find test cases for which your algo fails.

↓

Overflow issues

$n \log n$

0	1	2	3	4	5	6
5	2	3	-1	6	1	2
		j				i

6 1 2 -1 5 2 3

6 1 2 5 2 3 -1

R = 23

[1 2 3 4 5 6 7 11 12]

list < tuple < int, int > >

0	1, 1
1	2, 2

$\boxed{200}$

6-7 ผลลัพธ์

$7 \times 4 = 28/30$