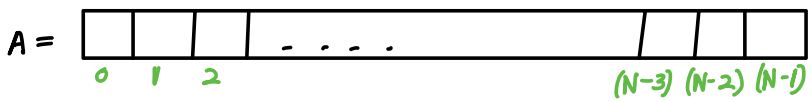
`int x;` → 4 Bytes `int x1, x2, x3, x4, x5;` → 5*4 Bytes

`int A[5];` → 5*4 Bytes

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



A[-100]
A[N+10]
A[N]
} Array index out of bound error.

Q→ Print all array elements.

i/p
void printArray(int A[]) {

TC = O(N) SC = O(1)

for i → 0 to (A.length - 1)
print(A[i])
}

int A[5]; //c++
int A[] = new int[5]; // Java

```
for(i = N; i > 0; i /= 2)
    for(j = 0; j < i; j++)
```

i	j	# iterations
N	[0 N-1]	N
N/2	[0 N/2 - 1]	N/2
N/4	[0 N/4 - 1]	N/4
⋮		
1	[0 0]	1

✓ ⇒ $(N + \frac{N}{2} + \frac{N}{4} + \cdots 1)$ ✓

Q→ Given an integer array of size N, count the number of elements having atleast 1 element greater than itself.

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

Ans = 5 (7-2)

A = [10 3 10 8 2 10 10 8]
0 1 2 3 4 5 6 7

Ans = 4 (8-4)

Observation → Only for largest element there will not be any element greater than itself.

Ans = N − (count of max elements)

A = [8 8 8 8 8] → Ans = 0

```
int count(int A[]) {
    N = A.length
    am = A[0]
    for i → 1 to (N-1)
        am = max(am, A[i]);   → if (A[i] > am)
                                    am = A[i]
    cnt = 0          // 4 Bytes integer
    for i → 0 to (N-1)
        if (A[i] == am)
            cnt += 1
    return N - cnt
}
```

TC = O(N+N) = O(N) ✓

SC = O(1)

H.W → Try the code in 1 for loop iteration. ✓

Q → Given an integer array of size N & an integer K, check if there exists a pair of index (i, j) s.t

boolean

A[i] + A[j] = K & i != j

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

K = 10

∵ A[3] + A[5] = 10

Ans = true

A = [2 4 -3 7] K = 5 Ans = false
0 1 2 3

A = [2 4 -3 7] K = 8 Ans = false
0 1 2 3

A = [2 3 8 5 3] K = 6 A[1] + A[4] = 6 Ans = true
0 1 2 3 4

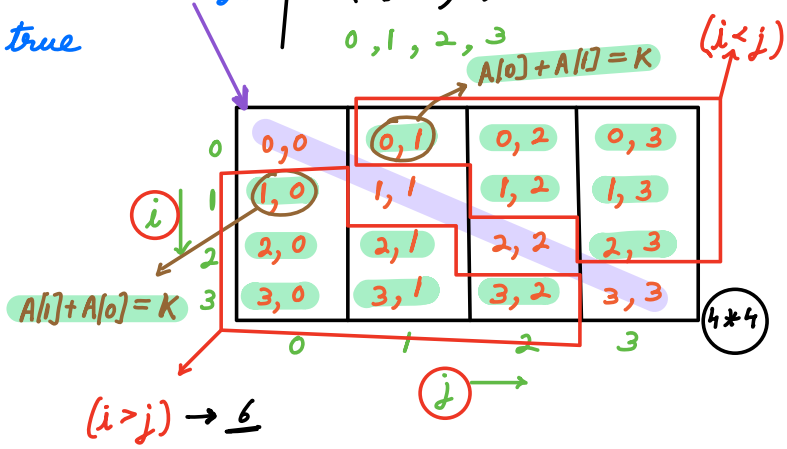
Bruteforce → (Try all possibilities) ∀ i,j pair check if A[i] + A[j] = K & i != j

```
boolean sumPair (int A[], K) {
    N = A.length
    for i → 0 to (N-1)
        for j → 0 to (N-1)
            if (A[i] + A[j] == K && i != j)
                return true
    return false
}
```

TC = O(N²)
SC = O(1)

(x + y) = (y + x)

∀ integers x, y



∴ only check (i > j) or (j > i) cases.

```
boolean sumPair (int A[], K) {
    N = A.length
    for i → 1 to (N-1)
        for j → 0 to (i-1)
            if (A[i] + A[j] == K)
                return true
    return false
}
```

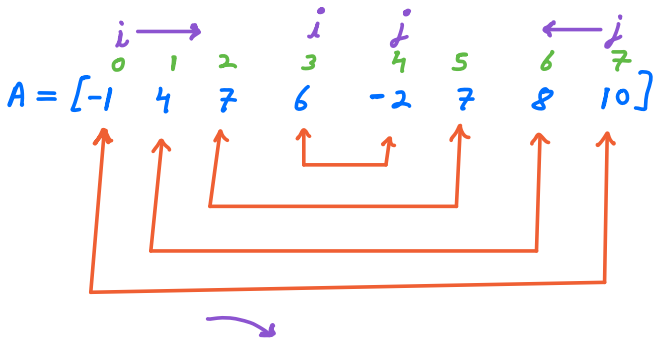
(j < i)

for i → x to y
for(i = x; i <= y; i++)

TC = O(N²)
SC = O(1)

10:37 PM

Q → Reverse the given integer array in SC = O(1).
(modify i/p array)



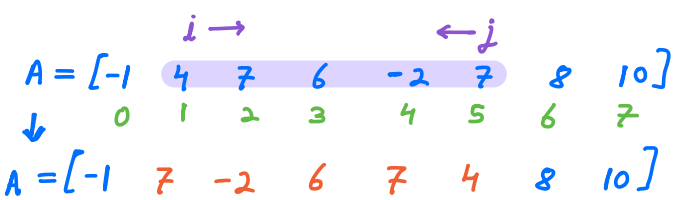
swap (A[0], A[7])
swap (A[1], A[6])
swap (A[2], A[5])
swap (A[3], A[4]) ✓
stop!

i < j

```
void reverse (int A[]) {
    i = 0
    j = A.length - 1
    while (i < j) {
        // swap A[i], A[j]
        temp = A[i]
        A[i] = A[j]
        A[j] = temp
        i++
        j--
    }
}
```

TC = O(N) SC = O(1)

Reverse from index L to R. L = 1 R = 5



```
void reverse (int A[], int L, int R) {
    i = L
    j = R
    while (i < j) {
        // swap A[i], A[j]
        temp = A[i]
        A[i] = A[j]
        A[j] = temp
        i++
        j--
    }
}
```

TC = O(N) (worst case: L=0, R=N-1) SC = O(1)

Q → Given an integer array of size N & an integer K, rotate the array from last to first (forward) by K times in SC = O(1).

(Modify the i/p array)

K	A = [	0	1	2	3	4	5	6	7	]
	A = [	-1	4	7	6	-2	7	8	10	]
1 →	A = [	10	-1	4	7	6	-2	7	8	]
2 →	A = [	8	10	-1	4	7	6	-2	7	]
③ →	A = [	7	8	10	-1	4	7	6	-2	]

K=0 A = [-1 4 7 6]

1 [6 -1 4 7]

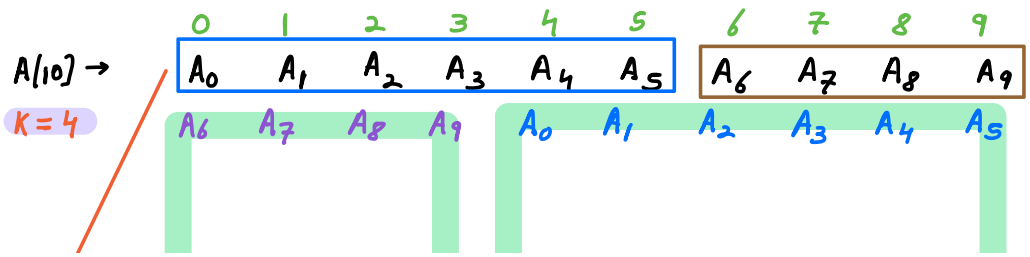
② [7 6 -1 4]

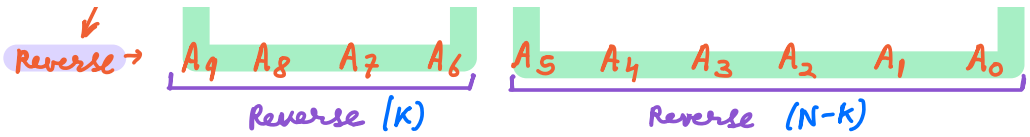
3 [4 7 6 -1]

4 [-1 4 7 6]

K=5, K=6 ← K=10

if (K >= N) K = K % N





reverse(A, 0, N-1)
reverse(A, 0, K-1)
reverse(A, K, N-1)

TC = O(N) SC = O(1)

H.W → Read about dynamic array in your programming language.

C++ → vector
Java → Arraylist ...
