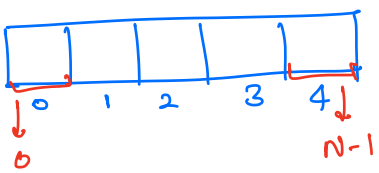
Today's Quote →

[Arrays]

Today's.

int arr[5] =



arr = { 3, 1, 2, 5, 7, 4, 9 }
0 1 2 3 4 5 7

arr[3], arr[i] → T.C → O(1)

How to print all the elements of array.

```
void printArr(arr) {
    for(i = 0 to N-1) {
        print(arr[i]);
    }
}
```

T.C → O(N)
S.C → O(1)

i : [0, N-1]
: N iterations.

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

arr[7] : { -3 (0) ✓, 2 (1) ✓, 6 (2) ✓, 8 (3), 4 (4) ✓, 8 (5) ✓, 5 (6) ✓ }

count → +1 +1 +1 +1 +1 = 5.

arr[8] : { 2 (0), 3 (1), 10 (2), 7 (3), 3 (4), 2 (5), 10 (6), 8 (7) }

ans = 6

arr[10] : { 2 (0), 5 (1), 1 (2), 4 (3), 8 (4), 0 (5), 8 (6), 1 (7), 3 (8), 8 (9) }

ans = 7

arr[5] : { 8 (0), 8 (1), 8 (2), 8 (3), 8 (4) }

ans → 0

observation: For maximum element, we can't have any no. greater than itself.

obs2. : Get count of no. of max elements → C

final ans. → total no. of elements − C.

C = no. of times max^m no. is appearing.

pseudocode →

```
int countGreater(arr, N) {
    max = arr[0];
    for(i → 1 to n-1) {
        if (arr[i] > max) {
            max = arr[i]
        }
    }                                   → N
    c = 0;
    for(i → 0 to n-1) {
        if (arr[i] == max) {
            c = c+1;
        }
    }                                   → N
    return N-c;
}
```

T.C → O(N)
S.C → O(1)

TODO: try to solve it by iterating on the array only once.

Python

```python
max = arr[0]
n = len(arr)
for i in range(1, n):
    if(arr[i] > max):
        max = arr[i]

count = 0
for i in range(0, n):
    if(arr[i] == max):
        count += 1

print(n - count)
```

Java.

```java
int n = arr.length;
int max = arr[0];
for(int i = 1; i < n; i++){
    if(arr[i] > max){
        max = arr[i];
    }
}
int count = 0;
for(int i = 0 ; i < n; i++){
    if(arr[i] == max){
        count++;
    }
}
System.out.println(n - count);
```

Q.) Given N array elements, check if there exists a pair(i,j) such that arr[i] + arr[j] == k & i != j

Note:→ i and j are index value, K is given sum.

arr[] : { 3 -2 1 4 3 6 8 } ∴ true
0 1 2 3 4 5 6

K=10

arr[] : { 2 4 -3 7 } ∴ false
0 1 2 3

K=5

arr[] : { 2 4 -3 7 } ∴ false.
0 1 2 3

K=8

i → 1 → 4
j → 1 → 4

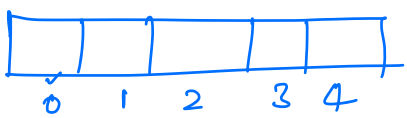
arr[] : { 3 5 2 7 3 } ∴ true
0 1 2 3 4

K=6

i j

arr[i] + arr[j]

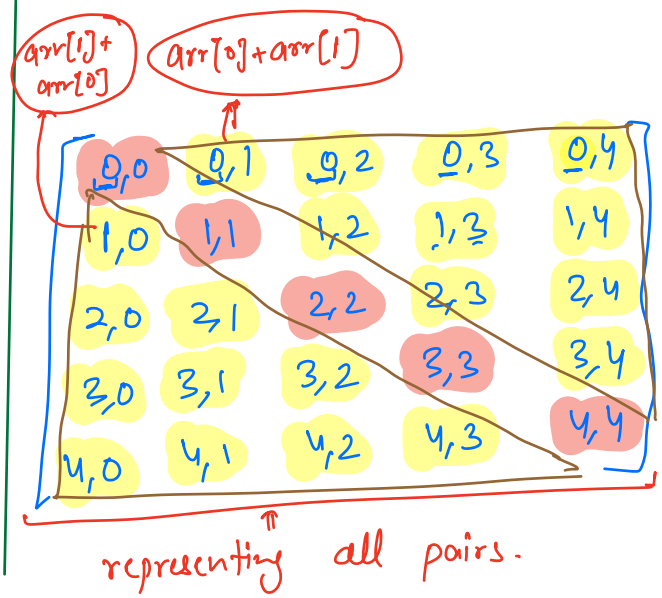
⇒ 3 + 3 = 6



idea-1 - Check all the pairs. If any pair (i,j) is having sum = K ⟶ return true.

```
boolean check( arr, N) {
    for( i = 0 to N-1) {
        for( j = 0 to N-1) {
            if ( i != j){
                if (arr[i]+arr[j] == K)
                {  return true;
                }
        }
    }
    return false;
}
```

All pairs



representing all pairs.

T.C → $O(N^2)$
S.C → $O(1)$

upper half.

i = 0	j → 1 to 4
i = 1	j → 2 to 4
i = 2	j → 3 to 4
i = 3	j → 4 to 4
i = 4	No iteration.

idea-2

```
boolean checkPair(arr, N, K){
    for(i = 0 to N-1){
        for(j -> i+1 to N-1){
            if(arr[i] + arr[j] == K){
                return true
            }
        }
    }
    return false;
}
```

table-

i	j → [i+1, N-1]	iterations
0	[1, N-1]	N-1
1	[2, N-1]	N-2
2	[3, N-1]	N-3
3	[4, N-1]	N-4
⋮	⋮	⋮
N-1	[N, N-1]	0

total no. of iterations = $(N-1) + (N-2) + (N-3) + \cdots + 3 + 2 + 1 + 0$

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

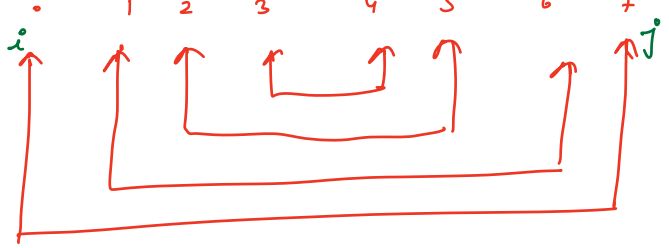
T.C → $O(N^2)$

S.C → $O(1)$

Q) Given an array : Reverse entire array. [s.c → O(1)]

Note → array itself should change.

arr[8] : { -1 4 7 6 -2 17 8 10 }

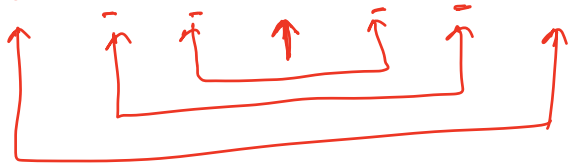


i	j
0	7
1	6
2	5
3	4
4	3

we have to break.

arr[] → { 10, 8, 17, -2, 6, 7, 4, -1 }

arr[7] : { -1 6 3 2 8 9 10 }



i	j
0	6
1	5
2	4
3	3

we have to break

arr[] : 10 9 8 2 3 6 -1

pseudo-code:

```
void reverse ( arr , n ) {
    i = 0 , j = n-1
    while ( i < j ) {
        temp = arr[i]
        arr[i] = arr[j]
        arr[j] = temp
        i = i+1
        j = j-1
    }
}
```

⇒ swap arr[i] with arr[j]

T.C → O(N)
S.C → O(1)

#Todo: try to solve this question using for loop.

starting index ending index

Q) Given N1 array elements and si & ei.

Reverse array from si to ei. Note si <= ei

arr[] = { -3 4 2 8 7 9 6 2 10 }, si = 3, ei = 7

indices: 0 1 2 3 4 5 6 7 8

arr[] : { -3 4 2 2 6 9 7 8 10 }

indices: 0 1 2 3 4 5 6 7 8

pseudo-code →

```
reverseArrayPart ( arr, N, si, ei ){
    i = si , j = ei
    while ( i < j ){
        temp = arr(i)
        arr(i) = arr[j]      } swap arr[i] with arr[j]
        arr[j] = temp
        i = i+1
        j = j-1
    }
}
```

no. of iterations = ei-si/2

T.C → O(ei-si) , S.C → O(1)
O(N)

Q) Given N array elements, Rotate array from last to first by K times. { Google, Amazon }

arr[7] : { 3 -2 1 4 6 9 8 } K=3

0 1 2 3 4 5 6

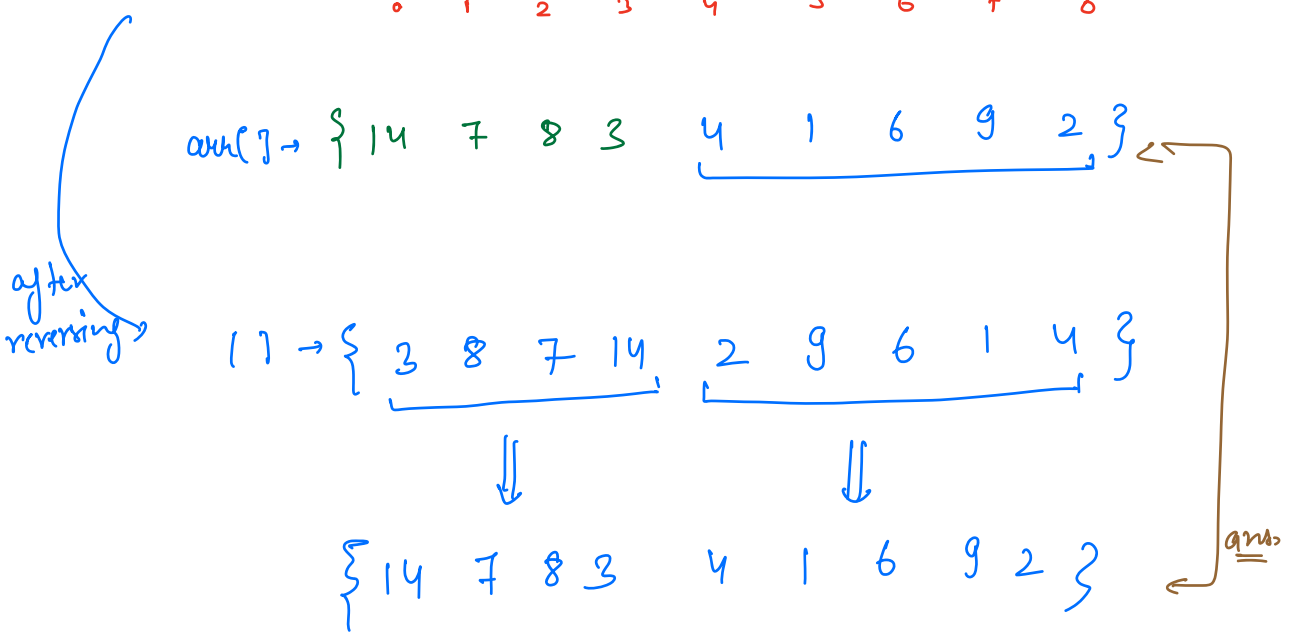
K=1. { 8 3 -2 1 4 6 9 }

K=2. { 9 8 3 -2 1 4 6 }

K=3 { 6 9 8 3 -2 1 4 }

arr[9] : { 4 1 6 9 2 14 7 8 3 } K=4

0 1 2 3 4 5 6 7 8



arr → { 3 2 7 4 6 1 9 }
0 1 2 3 4 5 6

K = 5

arr → { 7 4 6 1 9 3 2 }

rev

[] → { 9 1 6 4 7 2 3 }

reverse first K-elements

{ 7 4 6 1 9 2 3 }

reverse them

ans.

{ 7 4 6 1 9 3 2 }

Eg: arr → [0 1 2 3 4 5 6], K = 8

① K = K % N

② reverse whole array.

reverseArrayPart (arr, 0, n-1)

: (arr, 0, 6)

③ reverse first K elements

reverseArrayPart (arr, 0, K-1)

: (arr, 0, 7)

: Error.

④ reverse remaining elements

reverseArrayPart (arr, K, n-1)

T.C → O(N)

S.C → O(1)

arr : { 3 2 7 5 4 }
0 1 2 3 4

k=1 : { 4 3 2 7 5 }

k=2 : { 5 4 3 2 7 }

k=3 : { 1 5 4 3 2 }

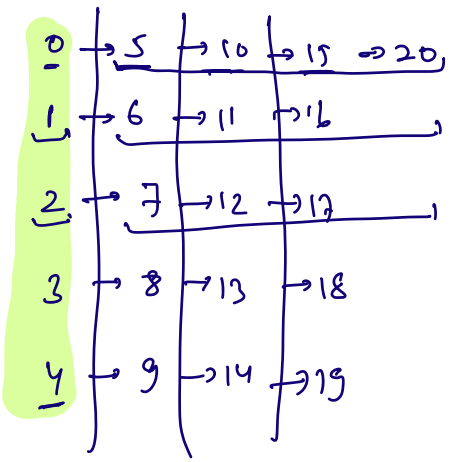
k=4 : { 2 1 5 4 3 }

k=5 : { 3 2 7 5 4 }

k=6 : { 4 3 2 7 5 }

k=7

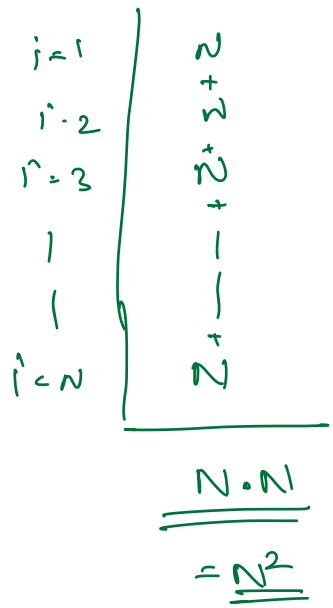
k=8



34 % 5 = 4.

//Doubts

```
for( i = 1 ; i <= N ; i++) {
    for( j = 1 ; j <= N ; j++){
        ==== //task
    }
}
```



$N \cdot N$

$= N^2$