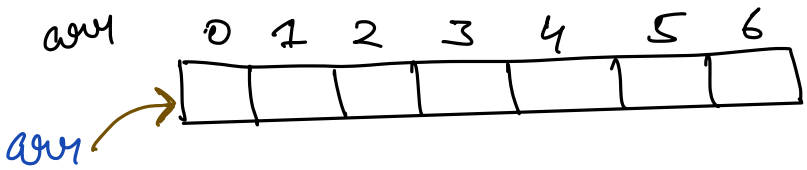
Welcome ☺ Arrays

Agenda : 4-5 problems



size = 7

arr[0] arr[3]

Func^n to print all elements of an array

HW

Time complexity to access ith element in an array

O(1)

arr[1]

Q1 Given N array elements, count no. of elements having atleast one element greater than itself.

eg: arr[7] : { -3 2 6 8 4 8 5 }

✓ ✓ ✓ ✗ ✓ ✗ ✓

Ans: 5

arr[10] : { 2 5 1 4 8 0 8 1 3 8 }

✓ ✓ ✓ ✓ ✗ ✓ ✗ ✓ ✓ ✗

Ans = 7

arr[5] : { 5, 5, 5, 5, 5 }

Obs: for maximum element, we don't have any element which can be greater.

Find count of how many times max. element is appearing.

ans = total element − count$_{max}$

Pseudocode

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

HW Do it in one loop

T.C → O(N)

S.C → O(1)

Q Given an array of size N. Check if there exists a pair (i, j) such that `arr[i] + arr[j] == K` and `i != j`

arr : { 3 -2 1 4 3 6 8 }

K = 10 True

arr : { 2 4 -3 7 }

K = 5 False

arr : { 2 3 7 -4 3 }

K = 6 True.

App 1 Brute force

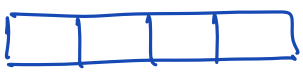
→ Check all pairs. If any pair (i, j) is having sum == K, return true.

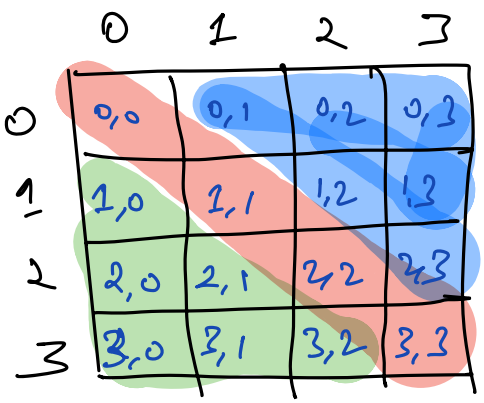
Pseudo code

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

T.C = $O(N^2)$

S.C = $O(1)$





Optimised code

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

T.C. $O(N^2)$

S.C $O(1)$

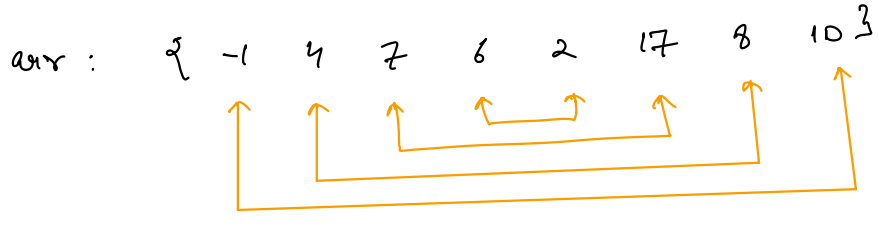
iterations : $\frac{N*(N-1)}{2}$

Q Given an array of size N. Reverse entire array. [S.C → O[1])

Note: array itself should be changed.

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

Ans : { 10 8 17 2 6 7 4 -1 }



i	j
✓ 0	7
✓ 1	6
✓ 2	5
✓ 3	4
✗ 4	3

break loop

Pseudo code:

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

T.C = O(N)

S.C = O(1)

Q Given an array of size N and s_i , e_i (start index, end index)

Reverse the array from s_i to e_i . $s_i < e_i$

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

$s_i = 3$

$e_i = 7$

arr ⇒ { -3 4 2 2 6 9 7 8 10 }

H.W

Google Amazon

Q Given an array of size N, rotate array from last to first by K times. K>0 S.C → O(1)

arr : { 3 -2 1 4 6 9 8 }

K=3

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

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

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

Eg:2 arr : { 2 4 1 6 9 2 14 7 8 3 }

K=4

arr { 14 7 8 3 4 1 6 9 2 }

reverse

{ 3 8 7 14 2 9 6 1 4 }

↓ reverse

{ 14 7 8 3 2 9 6 1 4 }

↓ reverse

{ 14 7 8 3 4 1 6 9 2 }

App.

⓪ $K = K \% N$

① Reverse entire array — reverse(arr, 0, N-1)

② Revers first K elements — revers(arr, 0, K-1)

③ Revers remaining element — reverse(arr, K, N-1)

T.C → O(N)

S.C → O(1)

eg: arr { 3 2 7 5 }

K = 6

K=1 → 5 3 2 7

K=2 → 7 5 3 2

K=3 → 2 7 5 3

K=4 → 3 2 7 5

K=5 → 5 3 2 7

K=6 → 7 5 3 2

→ 0 → 4 → 8 → 12

→ 1 → 5 → 9 → 13

→ 2 → 6 → 10 → 14

obs: arr is repeating after N iterations

∴ take modulo of