Today's Quote →

Good Morning !!

Today's content

arr[5] = | 5 | 7 | 4 | 3 | 11 |
0 1 2 3 4

indices : [0, N-1]

arr[2] , arr[4] , arr[i] T.C → O(1)

How to print all elements in the array ⇒

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

i : [0, N-1]
∴ N iterations

T.C → O(N)
S.C → O(1)

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

Eg → 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) }

count : +1 +1 +1 +1 +1 +1 ans = 6

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

count : +1 +1 +1 +1 +1 +1 +1 ans = 7.

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

observation: for maximum element, we don't have any element greater than itself.

find count of how many times max element is appearing. ⇒ c.

ans. = total no. of elements - c

pseudocode →

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

T.C → O(N)
S.C → O(1)

todo: try to solve this by iterating only once.

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

arr[] : { 3 5 2 1 -3 7 8 25 6 13 }
(0 1 2 3 4 5 6 7 8 9)

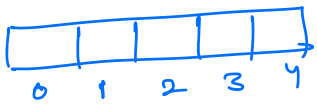
sum→10 (3,7), (2,8), (-3,13) : {true}

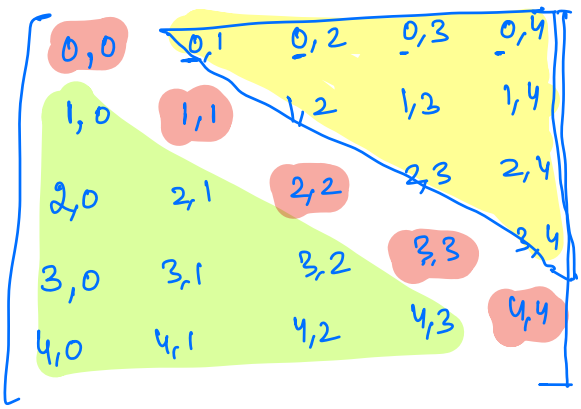
arr[] : { 2 3 7 -4 3 } : {true}
(0 1 2 3 4)

K=6

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

```
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;
}
```





all the pairs.

T.C → $O(N^2)$
S.C → $O(1)$

initially.

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

idea-2

```
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;
}
```

table

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

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

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

$\frac{N(N+1)}{2}$

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

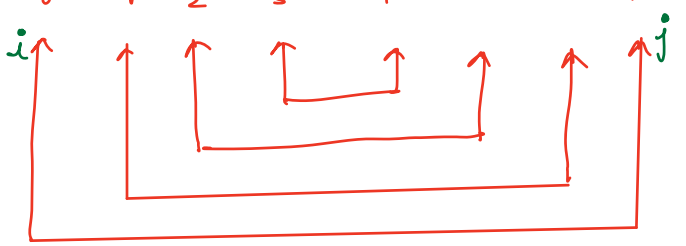
$\Rightarrow \frac{N(N-1)}{2}$.

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 }
0 1 2 3 4 5 6 7



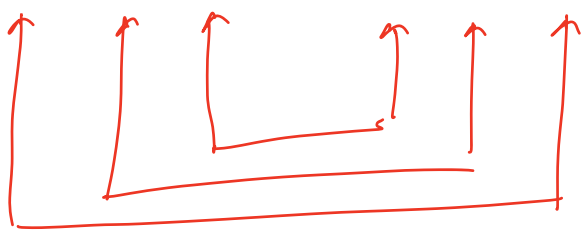
i	j
0	7
1	6
2	5
3	4
4	3

⇓
[break the loop]

arr[8] : { 10 8 17 -2 6 7 4 -1 }
0 1 2 3 4 5 6 7

---

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



i	j
0	6
1	5
2	4
3	3

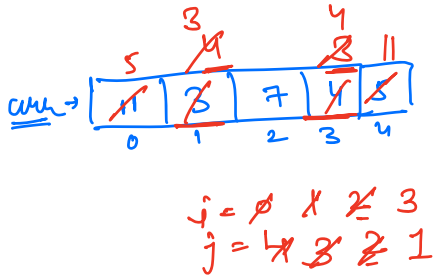
⇓
[Break the loop]

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

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
    }
}
```

→ swapping ith & jth idx value



i = 0 1 2 3
j = 4 3 2 1

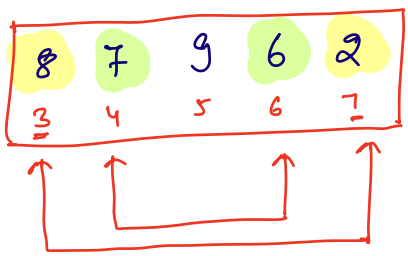
0 , 3
1 , 2
2 , 1
3 , 0

T.C → O(N)
S.C → O(1)

start-index ↑ si & ei ↑ end index

Q) Given N 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 →

```
void reversePart ( arr, N, si, ei ){
    i = si , j = ei
    while ( i < j ){
        int temp = arr[i]
        arr[i] = arr[j]
        arr[j] = temp
        i += 1
        j -= 1
    }
}
```

O(ei-si)

T.C → O(N)
S.C → O(1)

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

[S.C → O(1)]

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

reverse.

[7 → { 3 8 7 14 2 9 6 1 4 }

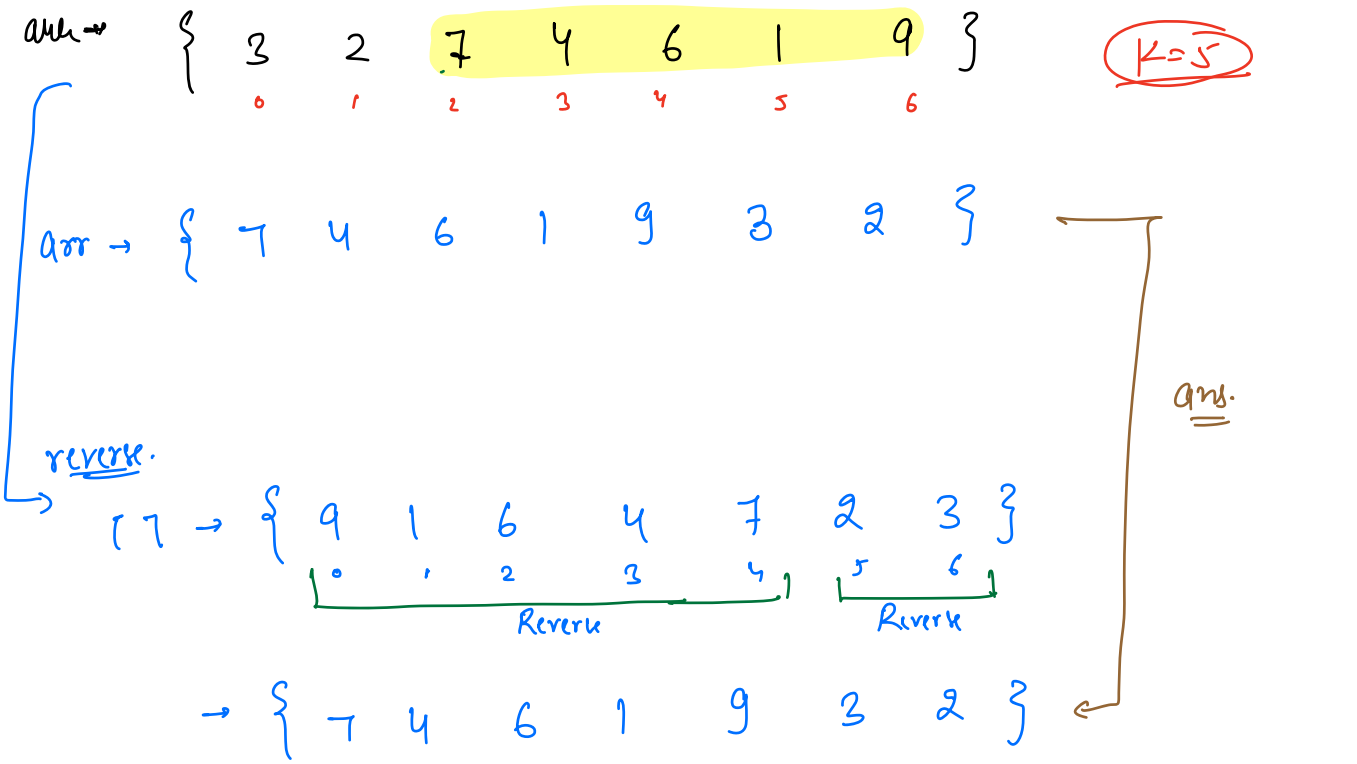
Reverse

[7 → { 14 7 8 3 2 9 6 1 4 }

Reverse

[7 → { 14 7 8 3 4 1 6 9 2 }

ans.



⓪ $K = K \% N$

① Reverse the entire array → reversePart(arr, 0, N-1) : (arr, 0, 6)

② Reverse first K - elements → reversePart(arr, 0, K-1) : (arr, 0, 7)

③ Reverse remaining elements → reversePart(arr, K, N-1)

arr → [0 1 2 3 4 5 6] , K=8 , N=7

K ≥ N

T.C → O(N)
S.C → O(1)

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

N=5.

K=N
K=N%N=0

K=1. { 4 3 2 7 5 }

K=2. { 5 4 3 2 7 }

K=3 { 7 5 4 3 2 }

K=4 { 2 7 5 4 3 }

K=5 { 3 2 7 5 4 }

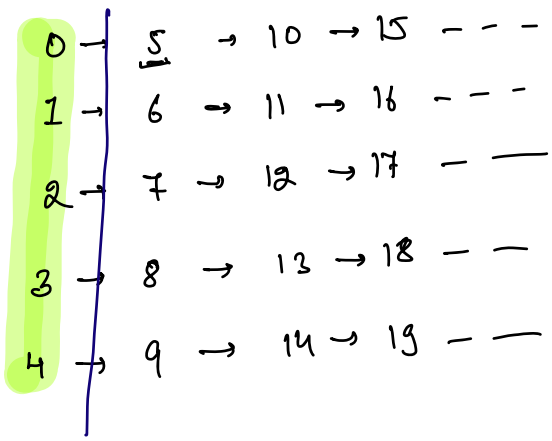
K=6 { 4 3 2 7 5 }

K=7 { 5 4 3 2 7 }

K=8 { 7 5 4 3 2 }

K=9 { 2 7 5 4 3 }

K=10 { 3 2 7 5 4 }



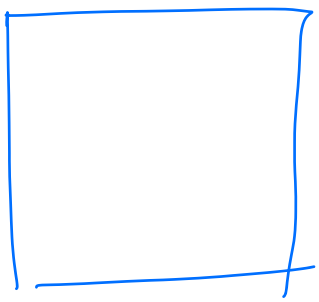
→ ans is repeating after N iterations.

→ take modulo with N.

⇒ K = K % N

Doubts →

→ $\propto$ no. of questions

→ 

→ similar to what we have discussed.

→ [small variance].

#tables

```
for( i=0 ; i < (2^n) ; i++) {
    j = i
    while ( j > 0){
        j -=1
    }
}
```

i	j	iterations.
0	[0, 0)	0
1	[1, 0)	1
2	[2, 0)	2
3	[3, 0)	3
⋮		⋮
2^n	$[2^n, 0)$	2^n

total no. of iteration = $0+1+2+3+ \cdots 2^n$.

Sum of first 2^n natural no.'s = $\frac{2^n(2^n+1)}{2}$.

$= \frac{2^{2n} + 2^n}{2} = \frac{4^n + 2^n}{2}$

$= \frac{4^n}{2} + \frac{2^n}{2}$

T.C → $O(4^n)$

```
for( i = 1; i*i <= n; i++){
    for( j = 1; j*j <= i ; j+=i){
        {
        }
    }
}
```

i	j	iterations.
1	[1,1]	1
2	[1,1]	1
3	[1,1]	1
4	[1,1]	1
5	[1,1]	1
.		
.		
.		
$\sqrt{n}$	[1,1]	1

$$1 + \sqrt{n} \quad (1+\sqrt{n}) \quad (1+\sqrt{n}) \quad <= \sqrt{n}$$

iteration = $\sqrt{n}$.

$i = N$

$j = N, N+1, N+2 \; \cdots \; 2N, 2N+1, 2N+2, \; \cdots \; 3N, \; \cdots \; 4N \; \cdots \; N^2.$

$[0, N] \Rightarrow N$

$[0, 2N] \Rightarrow 2N$

$[0, 3N] \Rightarrow 3N$

$[0, 4N] \Rightarrow 4N$

$[0, N^2] \Rightarrow N^2$

$\Rightarrow N + 2N + 3N + 4N + N^2$

$\Rightarrow N [1 + 2 + 3 + 4 + \cdots N]$

$\Rightarrow N \; \frac{N(N+1)}{2} \approx N^3.$

$i = n$

```
for(i=0; i<=n; i++){
    for(j=i; j<i*i; j++){
        if(j%i==0){
            for(k=0; k<j; k++){
                //print
            }
        }
    }
}
```

$j = n, n+1, n+2, \cdots 2n, 2n+1, \cdots 3n, \cdots n^2$

$[0, n] \Rightarrow n$ + $[0, 2n] \Rightarrow 2n$ + $[0, 3n] \Rightarrow 3n$ + $\cdots$ + n^2

Sum $= n^3$.

```
for( i = 3 ; i < n/3 ; i+=3){
    for( j = 2 ; j < n/2 ; j+=2){

    }
}
```

i	j	iterations.
3	$[2, n/2]$	$n/4$
6	$[2, n/2]$	$n/4$
9	$[2, n/2]$	$n/4$
1		⋮
1		
$n/3$	$[2, n/2]$	$n/4$.

$i \rightarrow 3, 6, 9, 12, \ldots \; n/3$ (+3)

total no. of iterations $= \underbrace{n/4 + n/4 + \ldots \; n/4}_{? \; n/9}$.

$= \frac{n}{4} \cdot \frac{n}{9} = \frac{n^2}{36}$

T.C $\rightarrow O(n^2)$

```
for( i=3 ; i < n/3 ; i+=3){
```

3

3, 6, 9, 12, 15, ——— $n/3$

1, 2, 3, 4, 5, 6, 7, 8, 9, 10, / — — $n/3$ ⇒ $n/3$.

$\frac{1}{3} \times n/3 = \boxed{n/9}$

```
for( i=0 ; i< N ; i++){
    for( j = i ; j < N ; j++){
        break.
    }
}
```

j=0	1
i=1	1 +
	|
	|
N-1	1

N iteration.

- stack
- recursion.
- queue
- d.d.

```
for( i=0 ; i<n ; i++){
    for( j=0 ; j<n ; j++){
        int sum=0
        sum +=10;
    }
}
```

i = 0
j = 1
sum = 0

12 bytes of space.

S.C → O(1)