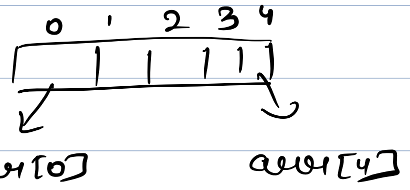
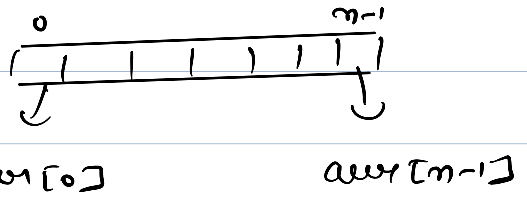
Today's Content

<- Arrays ->

int arr[5] =



int arr[n] =



```
void printarr(int arr[]) {
        int n = arr.length
        for (i=0; i<n; i++) {
        |    print (arr[i])
        }
}
```

T.C -> O(n)

S.C -> O(1)

Q1) Given N array elements, count no. of elements, having 1 element greater than itsef.

$$arr[7] = \{\underset{+1}{\overset{0}{-3}}, \underset{+1}{\overset{1}{-2}}, \underset{+1}{\overset{2}{6}}, \overset{3}{8}, \underset{+1}{\overset{4}{4}}, \overset{5}{8}, \underset{+1}{\overset{6}{5}}\} \Rightarrow \underline{5}$$

→ 7 − 2 ⇓

$$arr[8] = \{\overset{0}{2}, \overset{1}{3}, \overset{2}{10}, \overset{3}{7}, \overset{4}{3}, \overset{5}{2}, \overset{6}{10}, \overset{7}{8}\} \Rightarrow \underline{6}$$

↓ 8 − 2 ⇗

$$arr[4] = [\overset{0}{8}, \overset{1}{8}, \overset{2}{8}, \overset{3}{8}] \rightarrow \underline{0}$$

Obs1:- Largest no. of array can't have any element greater than it.

Step:- Iterate and get no. of max elements. → $\underline{c}$

final ans = (Total no. of elements) − ($\underline{c}$)

```
//Pseudo Code

int countGreater( int arr[]) {

    int n= arr.length;
    int max= arr[0]
    for (i=1; i<n; i++) {                    n-1
        if ( arr[i] > max) {                 iterations
            max= arr[i]
        }
    }

    int freq=0;
    for (i=0; i<n; i++) {
        if (arr[i]==max) {                   n
            freq++
        }
    }

    return (n-freq);

}
```

Total Iterations are $n-1+n \Rightarrow 2n-1$

$O(n)$

S.C → $O(1)$

Todo:- Try to do the above Solution using only 1 for loop.

Discuss it → Next class → doubt session.

Ques2) Given N array elements, check if there exists a pair (i, j) such that $arr[i] + arr[j] == k$ && $i != j$.

$$\begin{array}{lccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 \\ ar = \{ & 3, & -2, & 1, & 4, & 3, & 6, & 8 \} \end{array}$$

k = 10 $\quad$ (i = 3, j = 5)

$$\begin{array}{lcccc} & 0 & 1 & 2 & 3 \\ ar[\,] = \{ & 2, & 4, & -3, & 7 \} \end{array}$$

k = 5 $\quad$, return false

$$\begin{array}{lcccc} & 0 & 1 & 2 & 3 \\ arr[\,] = \{ & 2, & 4, & -3, & 7 \} \end{array}$$

k = 8

idea1:- Make all the pairs and check their sum.

```
bool   sum(int[] arr, int k) {
          int n= arr.length;
          for(i=0; i<n; i++) { →
               for (j=0; j<n; j++) {
                    if (arr[i] + arr[j]==k &&
                                   i!=j)
                         return True
                    }
               }
          }
          return false;
}
```

i	J	Total iter
0	[0 n-1]	n
1	[0 n-1]	n
⋮	⋮	⋮
n-1	[0-n-1]	n
		n^2

i ⇒ [0, n-1] , J ⇒ [0, n-1]

i=0 , J ⇒ 0 to n-1

i=1 , J → 0 n-1

i=2 J 0 to n-1

⋮

i=n-1 J → 0 to n-1

T.C → O(n²)

S.C → O(1)

n=4

(i,J)

	J			
i	0 0	0 1	0 2	0 3
	1 0	1 1	1 2	1 3
	2 0	2 1	2 2	2 3
	3 0	3 1	3 2	3 3

i → 0 , J → 1 to 3.
→ 1 , J → 2 to 3
→ 2 , J → 3 to 3 .

```
bool   Sum ( int [] arr, int k ) {
         int  n = arr.length;
         for ( i = 0 ; i < n ; i++ ) {
             for ( J = i+1 ; J < n; J++ ) {
                 if ( arr[i] + arr[J] == k )
                         return True
                 }
             }
         }
         return false;
}
```

i	J	Total it.
0	[1 n-1]	n-1
1	[2 - n-1]	n-2
2		.
⋮		.
n-2	[n-1 - n-1]	1
n-1	[n n-1]	0

for first n numbers $\frac{n(n+1)}{2}$.

for first n-1 numbers $\frac{(n-1)(n)}{2}$

T.C → $O(n^2)$

S.C → $O(1)$

10-20 pm 10: 27 pm

→ constraints → {space should be constant}

Ques) (Given an array, Reverse entire arr[].

Note:- arr[] itself should change.

0 1 2 3 4 5 6 7

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

↳ {10, 8, 7, -2, 6, 7, 4, -1}

Approach1:- Two pointer approach.

P_1 P_2

0 1 2 3 4 5 6 7

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

0 1 2 3 4 5 6

arr[7] = {-1, 4, 7, 6, -2, 7, 8}

P_1

P_2

arr is reversed → 8, 7, -2, 6, 7, 4, -1

```
void reverse(int arr[]) {
    int P1 = 0, P2 = n-1
    while (p1 < P2) {
        swap arr[P1], arr[P2]
        P1++, P2--
    }
}
```

↓

```
int temp = arr[P1]
arr[P1] = arr[P2]
arr[P2] = temp.
```

S.C → $O(1)$

T.C → $O(N)$

Ques) Given N array elements & [si & ei].

reverse array from [si, ei]

[si <= ei]

$$\begin{array}{lccccccccc} & & & & P_1 & & & & P_2 & \\ & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ arr[\,] = \{ & -3, & 4, & 2, & 8, & 7, & 9, & 6, & 2, & 10\,\} \end{array}$$

s = 3

e = 7

$$\begin{array}{lccccccccc} & 0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ \{ & -3, & 4, & 2, & 2, & 6, & 9, & 7, & 8, & 10\,\} \end{array}$$

// pseudo code

```
void reversepart( int arr[], int s, int e)
    int p1 = s, p2 = e
    while (p1 < p2) {
        swap arr[p1], arr[p2]
        p1++, p2--
    }
}
```

```
int temp = arr[p1]
arr[p1] = arr[p2]
arr[p2] = temp.
```

T.C → O(N)

S.C → O(1) → S.C → (constant)

Ques) Given n array elements, Rotate array from last to first by k times → {google / Amazon}

k=3,	0	1	2	3	4	5	6
arr[7] =	3	-2	1	4	6	9	8
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

0 1 2 3 4 5 6 7 8

arr [9] = 4 1 6 9 2 14 7 8 3

k = 4 14 7 8 3 4 1 6 9 2

k = 5

arr → a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7 a_8 a_9 a_{10} a_{11} a_{12}

↓

F.ans → a_8 a_9 a_{10} a_{11} a_{12} a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7

Step1

Reverse the whole array.

$$\underbrace{a_{12}\ a_{11}\ a_{10}\ a_9\ a_8}_{\substack{\downarrow \\ \text{reverse} \\ \text{Step2}}}\quad \underbrace{a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0}_{\substack{\downarrow \\ \text{reverse.} \\ \text{Step3}}}$$

a_8 a_9 a_{10} a_{11} a_2 a_0 a_1 a_2 a_3 a_4 a_5 a_6 a_7

idea :-

S1 :- Reverse entire array

→ reversepart (arr, 0, n-1)

S2 :- Reverse first k elements.

→ reversePart (arr, 0, k-1)

S3 :- Reverse rest elements.

→ reversePart (arr, k, n-1)

// pseudo Code

```
void RotatekTimes(int arr[], int k) {
        int n = arr.length;                                Iterations
        // Reverse entire array.
          reversePart (arr, 0, n-1);  → $\frac{n}{2}$
        // Reverse first k elements
          reversePart (arr, 0, k-1);  → $\frac{k}{2}$
        // Reverse last n-k elements
          reversePart (arr, k, n-1)  → $\left(\frac{n-k}{2}\right)$
}
```

Total => $\frac{N}{2} + \cancel{\frac{k}{2}} + \frac{n}{2} - \cancel{\frac{k}{2}} \Rightarrow \underline{n}$

Total Iterations → N,

T. Complexity → O(N)

S.C → O(1)

→ rotate

N = 6, k = 8

arr[6] = a_0 a_1 a_2 a_3 a_4 a_5 , k=0

k=1, a_5 a_0 a_1 a_2 a_3 a_4

k=2 a_4 a_5 a_0 a_1 a_2 a_3

k=3 a_3 a_4 a_5 a_0 a_1 a_2

k=4 a_2 a_3 a_4 a_5 a_0 a_1

k=5, a_1 a_2 a_3 a_4 a_5 a_0

k=6, a_0 a_1 a_2 a_3 a_4 a_5

0	6	12	18
1	7	13	
2	8	14	
3	9	15	
4	10	16	
5	11	17	

% 6

```
void RotateKTimes(int arr[], int k){
        int n= arr.length;  k = k%n
                                                  Iterations,
      // Reverse  entire  array.
         reversePart (arr, 0, n-1); → n/2
      // Reverse  first  k  elements
         reversePart (arr, 0, k-1); → k/2
      // Reverse  last  N-k  elements
         reversePart (arr, k, n-1) → ((n-k)/2)
}
```

Inbuilt functions →

$1<<N$, 2^n