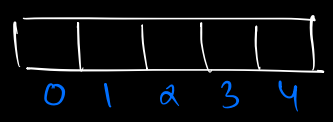
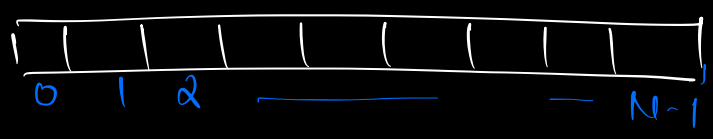
int arr[5] → 

↓

integer array with 5 elements

int arr[N] → 

↓

int. array with N elements

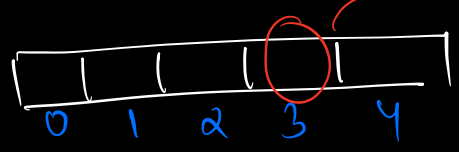
// print an entire array

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

⇒ TC : O(N)

→ accessing element from ith index ⇒ arr[i] → O(1)

int arr[5] ⇒



4th element
3rd index

get the 4th element ⇒ arr[3]

Q.1. Given N array element, count no. of element having atleast 1 element greater than itself [no lib. func.]

largest | count

arr[7] ⇒ { -3 -2 6 8 4 8 5 } ⇒ 8 2

✓ ✓ ✓ ✗ ✓ ✗ ✓

ans = 5.

arr[8] ⇒ { 2 3 10 7 3 2 10 8 } ⇒ 10 | 2

✓ ✓ ✗ ✓ ✓ ✓ ✗ ✓

ans = 6.

arr[10] ⇒ { 2 5 1 4 8 0 8 13 8 } 13 | 1

✓ ✓ ✓ ✓ ✓ ✓ ✓ ✗ ✓

ans = 9.

Bruteforce

```
for( i=0; i<N; i++) {
        for( j=0; j<N; j++) {
            if( any bigger element found)
                cnt++; break;
        }
    }
return cnt.
```

TC ⇒ $O(N^2)$.

Observation

I) every element except the largest element will have any element bigger than itself

II) ans = all element except largest elements

↓

ans = N − (count of largest of element)

pseudo

I) iterate to get maxm

II) iterate the get count of maxm

III) ans = N − count of maxm

```
maxV = 0
for( i=0; i<N; i++) {
        if( maxV < arr[i] ) {
                maxV = arr[i]
        }
}
c=0
for( i=0; i<N; i++) {
        if ( maxV == arr[i] )
                c++
}
```

not correct, if all element -ve, it wont work

valid values

maxV = INT_MIN

maxV = arr[0]

return arr.length − c;

ex ⇒ arr[6] ⇒ [2 2 2 2 2 2]

ans = 0.

maxm = 2

count = 6

ans = l − count

= 6 − 6 = 0

TC ⇒ O(2N) ≃ O(N)

SC ⇒ O(1).

How ⇒ Change this code, and solve in a single loop.

Q2 Given N array elements, check if there exists a pair (i, j) such that arr[i] + arr[j] == k, & i != j. Note :- i & j are indexes, k is given sum.

arr ⇒

3	−2	1	4	3	6	8
0	1	2	3	4	5	6

k = 10.

ans = true ⇒ arr[3] + arr[5] = 10

arr ⇒

2	4	-3	7
0	1	2	3

k = 5

op ⇒ false.

arr ⇒

2	4	-3	7
0	1	2	3

k = 8

op ⇒ false

(arr[1] + arr[1] == 8)
XXX not-allowed.

Soln

check for all pairs.

N = 5 ⇒ { _ _ _ _ _ }
0 1 2 3 4

pairs ⇒

(0,0)	(1,0)	(2,0)	(3,0)	(4,0)
(0,1)	(1,1)	(2,1)	(3,1)	(4,1)
(0,2)	(1,2)	(2,2)	(3,2)	(4,2)
(0,3)	(1,3)	(2,3)	(3,3)	(4,3)
(0,4)	(1,4)	(2,4)	(3,4)	(4,4)

Brute force

```
for( i=0; i<N; i++){
        for( j=0; j<N; j++){
        if(i==j) continue;
         if(arr[i] + arr[j] == k)
                return true
        }
}
return false
```

iterations => N * N => N^2

TC => $O(N^2)$

SC => $O(1)$

Optimised

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

$N-1 \times 4 \vee$

$N-1$

$N-1-2+1$

$= N-2$

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

$N-1 + N-2 + N-3 + \cdots + 3 + 2 + 1 + 0$.

$\Rightarrow 1 + 2 + 3 + \cdots + N-1$

$\Rightarrow \dfrac{(N-1) \times (N-1+1)}{2} \Rightarrow \dfrac{N(N-1)}{2}$ iteration

B.F $\Rightarrow N^2$ iterations $\Rightarrow TC \Rightarrow O(N^2)$

Optimised $\Rightarrow \dfrac{N(N-1)}{2}$ iterations $\Rightarrow TC \Rightarrow O(N^2)$

baeldung

Java revisited

How to do it in Java

scaler topics

Q3. Given an array, reverse entire array. | SC :- O(1).

arr ⇒	-1	4	7	6	-2	7	8	10
	0	1	2	3	4	5	6	7

rev. arr ⇒	10	8	7	-2	6	7	4	-1
	0	1	2	3	4	5	6	7

SC ⇒ O(1) → no extra array
given array itself should change (get reversed)

	0	1	2	3	4	5	6	7
arr ⇒	-1	4	7	6	-2	7	8	10

result ⇒	10	8	7	-2	6	7	4	-1

	0	1	2	3	4	5	6
arr ⇒	-1	6	3	2	8	9	10

i	j	
0	6	→ swap.
1	5	→ swap.
2	4	→ swap
3	3	→ swap.

pseudo

```
reverse (arr, N) {
    i = 0
    j = N-1
    while ( i <= j ) {
        swap(arr[i], arr[j])
        i++
        j--
    }
    return arr
}
```

$$\boxed{\begin{array}{l} TC \Rightarrow O(N) \\ SC \Rightarrow O(1) \end{array}}$$

Q4. Given N array elements, & $[S_i, E_i]$, reverse array from $[S_i - E_i]$, note $\Rightarrow S_i \leq E_i$.

	0	1	2	3	4	5	6	7	8
arr ⇒	-3	4	2	8	7	9	6	2	10

$S_i = 3$

$E_i = 7$

rev. arr ⇒ -3 4 2 2 6 9 7 8 10.

pseudo

```
reversePart(arr, Si, Ei){
    i = Si , j = Ei
    while (i <= j) {
        swap(arr[i], arr[j])
        i++
        j--;
    }
}
```

$$\boxed{\begin{array}{l} TC \Rightarrow O(N) \\ SC \Rightarrow O(1) \end{array}}$$

Q5. Given N array elements, rotate array from last to first by k times.

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

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

k = 4

rot arr = 14 7 8 3 4 1 6 9 2

arr[10] = -2 3 1 4 6 2 8 7 9 3

k = 3

rot. arr = 7 9 3 -2 3 1 4 6 2 8

last 3 element of original ⇒ first 3 elements of rotated array

first 7 elements of original ⇒ last 7 element of rotated array

arr[13] ⇒ 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}

k = 5

rot arr ⇒ 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

reverse ⇒ $\underbrace{a_{12}\ a_{11}\ a_{10}\ a_9\ a_8}_{\text{reverse first k element}}$ $\underbrace{a_7\ a_6\ a_5\ a_4\ a_3\ a_2\ a_1\ a_0}_{\text{reverse last (N-k) element}}$

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

$\Rightarrow$ $k = k\%N$

pseudo

i) reverse the entire array $\rightarrow$ reversePart(arr, 0, N-1)

ii) reverse the first k element $\rightarrow$ reversePart(arr, 0, k-1)

iii) reverse the last N-k element $\rightarrow$ reversePart(arr, k, N-1)

$$\boxed{\begin{aligned} TC &\Rightarrow O(N + k + N-k) \\ &= O(2N) \simeq O(N) \\ SC &\Rightarrow O(1) \end{aligned}}$$

$k > N \Rightarrow$ reversePart(arr, 0, (k-1))

$\Rightarrow N-1$

arr[5] k=0. $\Rightarrow$ a_0 a_1 a_2 a_3 a_4

k=1 $\Rightarrow$ a_4 a_0 a_1 a_2 a_3

k=2 $\Rightarrow$ a_3 a_4 a_0 a_1 a_2

k=3 $\Rightarrow$ a_2 a_3 a_4 a_0 a_1

k=4 $\Rightarrow$ a_1 a_2 a_3 a_4 a_0

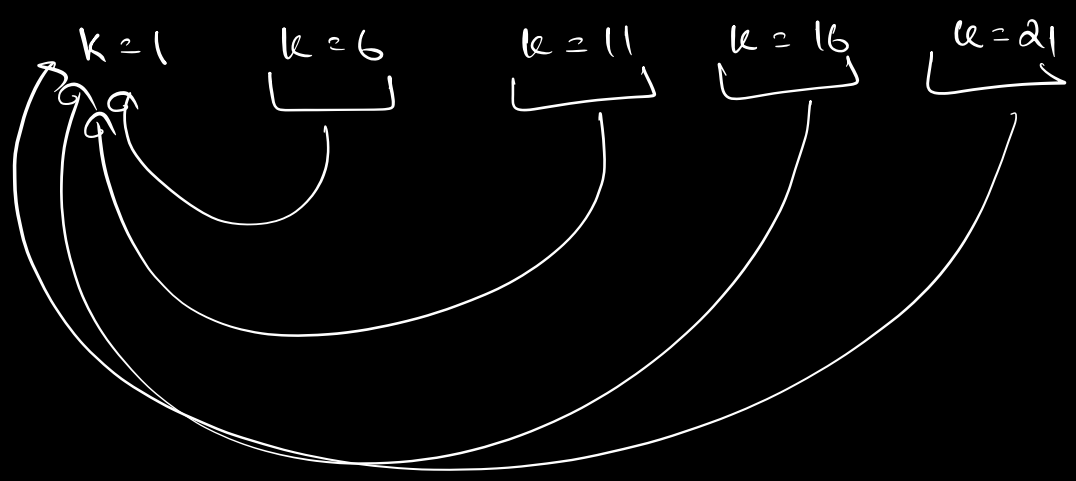
k=5 $\Rightarrow$ a_0 a_1 a_2 a_3 a_4

k=6 $\Rightarrow$ a_4 a_0 a_1 a_2 a_3

k=7 $\Rightarrow$ a_3 a_4 a_0 a_1 a_2

k=8 $\Rightarrow$ a_2 a_3 a_4 a_0 a_1

$N = 5$



$K = k \% N$

↳ mate for K

Optional

How ⇒ Rotate k time (L → R)

Static length array → arr[5] → fixed arrays.

dynamic arrays → arrays with dynamic length

C++ → vector

Java → arraylist

Python → list

C# → arraylist | list

Js → array

C → change to C++ / Java.

list<int> l

datatype — name

l.size() → 0

l.insert(1)

l.insert(2)

l.insert(3)

l.size() → 3

0	1	2
1	2	3

print(l.get(1)) → 2

TC ⇒ l.get()
l.insert()
l.size()
] O(1)

// print all elements of a list

```
for i=0; i<l.size(); i++) {
    print(l.get(i))
}
```