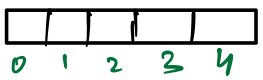
Introduction to Arrays

Array ?

↳ contigous list of homogenous items
↳ same type

↳ size is fixed → static size

A = 

→ index to access a value in a array

For an array of size N : 0 to N-1 indices

0, 1, 2, 3, . . . , N-2, N-1

a[n] a[i] = ? TC = O(1)

To access all the elements it will take O(N) time.

A[10] → a[0], a[1], ... , a[9], a[10], a[N], . .

Traverse in Array

```
int a[n]
for (i=0; i<n; ++i) {
    a[i] . . . . .
}
```

iterations = N

TC = O(N)

Question 1

Given an array of size N. find count of elements which has atleast one greater element than itself.

eg 1 4 5 3 -1 5 4 ans = 5

eg 2 5 1 4 8 0 8 1 3 8 ans = 7

1. find the max element
2. count the occurences of max element
3. ans = N - count

```
mx = a[0]
for (i=1; i<n; ++i) {
    if (a[i] > mx)
        mx = a[i]
}
```

N-1 iteration

```
count = 0
for (i=0; i<n; ++i) {
    if (a[i] == mx)
        ++count
}
```

N iteration

```
ans = N - count
```

iterations = $2N-1$ ($N-1+N$)

TC = O(N)

SC = 4+4+4 B = O(1)

HW: Solve in one ~~for~~ loop.

eg -3 -1 -2 mx=0

In case of all -ive values, mx=0 will not work.

Question 2

Given an array of N elements. find count of pair (i,j) where i,j are indices such that

a[i] + a[j] = K (given) i<j

eg a[6] = { 1 5 6 2 4 3 } K=7

0 1 2 3 4 5

(0,2) (1,3) (4,5) ans=3

a[] = { 3 5 2 1 -3 7 8 15 6 13 }

0 1 2 3 4 5 6 7 8 9

K=10

(0,5) (2,6) (4,9) ans=3

a[4] → {0,1,2,3} → N^2 pairs

i j

(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)

```
count=0
for(i=0; i<n; ++i) {
    for(j=i+1; j<n; ++j) {
        //(i,j) is valid
        if(a[i]+a[j]==K)
            ++count
    }
}
print(count)
```

i	j = [i+1, n-1]	iterations
0	[1, n-1]	n-1
1	[2, n-1]	n-2
⋮	⋮	⋮
n-2	[n-1, n-1]	1
n-1	[]	0

$= 1+2+\dots+n-1$

iterations $= \frac{n(n-1)}{2}$

$= \frac{n^2}{2} - \frac{n}{2}$

TC : $O(N^2)$

```
count=0
for(i=0; i<n; ++i) {
    for(j=0; j<n; ++j) {
        if(i<j && a[i]+a[j]==K)
            ++count
    }
}
print(count)
```

iterations $= n^2$

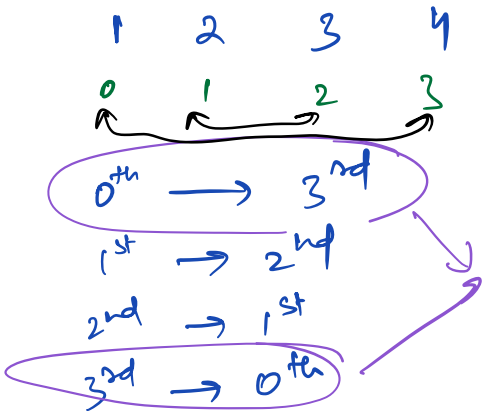
TC : $O(N^2)$

SC = O(1)

Question 3

Given an array of size N. Reverse the array without using extra space.

eg {1 2 3 4} reverse → {4 3 2 1}



a[0] ⟷ a[n-1]

a[1] ⟷ a[n-2]

⋮

a[i] ⟷ a[j]

j = n-i-1

0 + n-1 = n-1

1 + n-2 = n-1

i + j = n-1 ⇒ j = n-i-1

```
for (i=0; i<n; ++i) {
    swap(a[i], a[n-i-1]);
}
```

HW ← swap

✗ NOT WORK

a[4] = 1 2 3 4

0 1 2 3

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

1 2 3 4 5

⇓

5 4 3 2 1

```
i=0, j=n-1
while (i<j) {
    swap(a[i], a[j]);
    i++, j--;
}
```

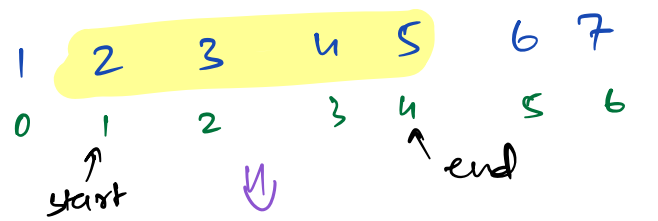
WORKS

iterations = n/2

TC: O(N)

SC: O(1)

What if we have to reverse only some part of array. : [start, end]



1 5 4 3 2 6 7

```
i = start, j = end;
while (i<j) {
    swap(a[i], a[j]);
    i++, j--;
}
```

BREAK : 10:17 - 10:27

Question 4

Given an array of size N. Rotate your array K times clockwise. K<N

eg

	1	2	3	4	5
	0	1	2	3	4
K=1	5	1	2	3	4
K=2	4	5	1	2	3
K=3	3	4	5	1	2
	0	1	2	3	4

K=3

a[0] = 1 $\xrightarrow{+3}$ a[3]=1

a[1] = 2 $\xrightarrow{+3}$ a[4] = 2

a[2] = 3 $\xrightarrow{-2}$ a[0] = 3

a[3] = 4 $\xrightarrow{-2}$ a[1] = 4

a[4] = 5 $\xrightarrow{-2}$ a[2] = 5

(0+3) % 5 = 3

(1+3) % 5 = 4

(2+3) % 5 = 0

(3+3) % 5 = 1

(4+3) % 5 = 2

```
int temp[n]
for(i=0; i<n; ++i) {
    j = (i+K)%n
    temp[j] = a[i]
}
for(i=0; i<n; ++i) {
    a[i] = temp[i]
}
```

[a[j] = a[i]] → location of value

TC: O(N)

SC: O(N)

4	5	3	1	2
0	1	2	3	4

K=3

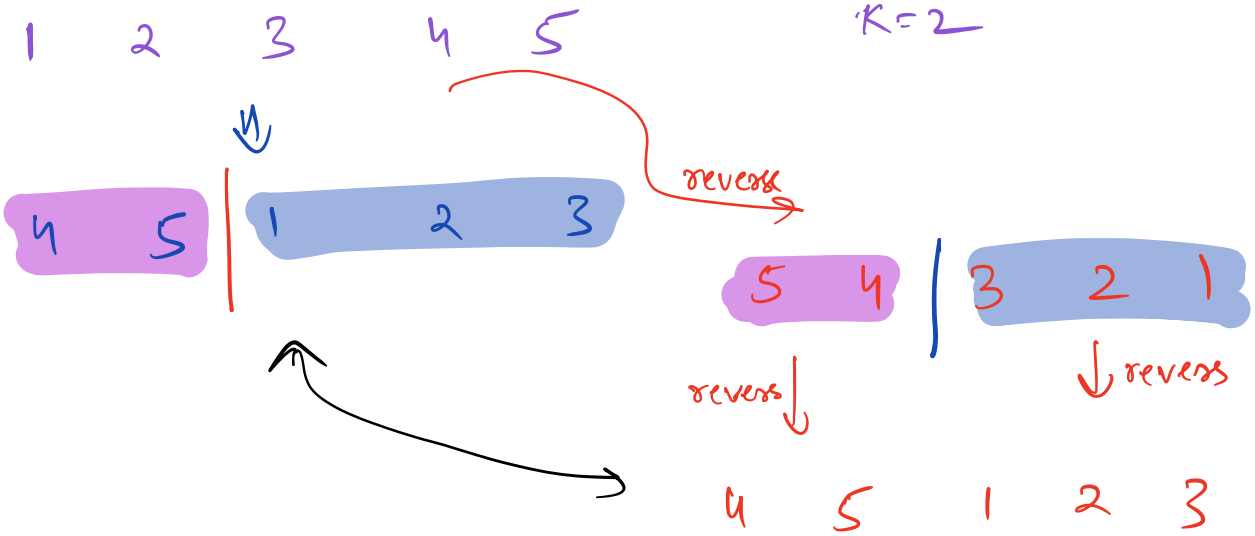
i=0, j=3
i=1, j=4
i=2, j=0

If we don't want extra space, we can shift one by one in K iterations.

```
for (i=0; i<K; ++i) {
    temp = a[n-1];
    for (j=1; j<n; ++j) {
        a[j] = a[j-1]
    }
    a[0] = temp
}
```

TC : O(K*N)

SC : O(1)



To shift element K times

clockwise ⇒

iterations

reverse (0, N-1) ⇒ N/2

reverse (0, K-1) ⇒ K/2

reverse (K, N-1) ⇒ N-K/2

total iterations = $N/2 + K/2 + N-K/2 = \frac{N+K+N-K}{2}$

$= N$

TC: O(N)

SC: O(1)

reverse(a, 0, n-1)

reverse(a, 0, k-1)

reverse(a, k, n-1)

⇐

```
void reverse(a[], start, end){
    i=start, j=end
    while(i<j){
        swap(a[i], a[j])
        ++i, --j
    }
}
```

Dynamic Array: Array with variable size

C++	Java	Python	JS/Ruby
vector	ArrayList	List	array

append() / add() / insert() / push_back()

HW: Understand dynamic array in your language.