Introduction to Arrays

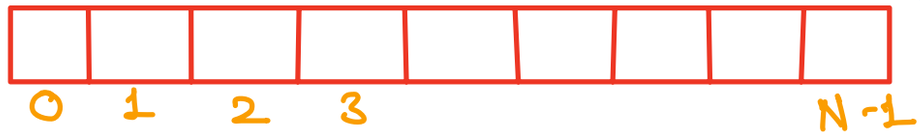
Vinay Neekhra

Senior Instructor & Mentor

Reachable in Scaler Lounge

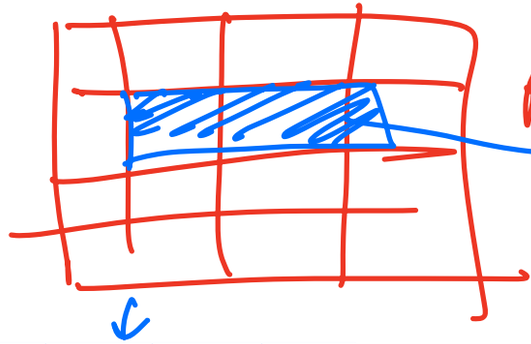
"Computer science is not really about computers and it's not about computers in the same sense that astronomy is not about telescope"

§ Array:



⇒ Give me a memory chunk to store 10 elements

Compiler says to OS



RAM

10 elements size

, N = 5 [0, N-1]

A[2] = 9 ⇒ [T.C. = O(1)]

Q How to print all elements in the array:

```
Void printArray (vector<int> A) {
    int N = A.size();
    for (i=0; i<N ; i++) {
```

// C++ declaration for arrays.

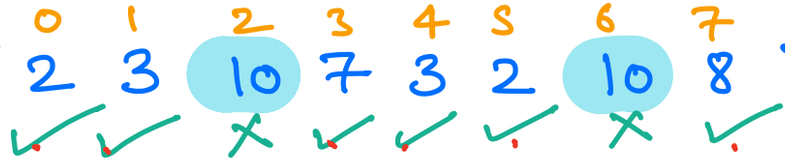
```
        print(A[i]);
    }
}
```

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.

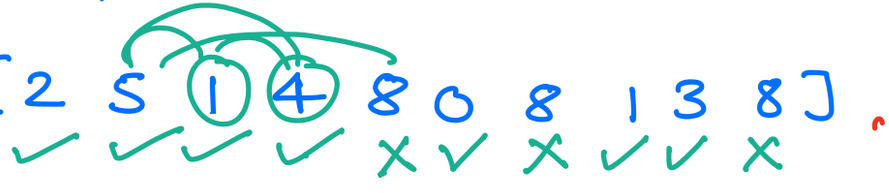
ex.1 A = [-3 2 6 8 4 8 5], Ans = 5

ex.2 A = [2 3 10 7 3 2 10 8], Ans = 6 [8-6]



N = 0, A = [ ], ans = 0
N = 1, A = [2], ans = 0
N = 1, A = [5], ans = 0

A = [2 5 1 4 8 0 8 1 3 8], ans = 7



(*) A = [7 7 7 7 7], ans = 0

Obs^n → (1) For maximum element, we don't have any element greater than itself

(2) ans = total elements − #[max element] (number)

```
1. int countGreater(A){
2.     int N = A.size();
3.     int max = A[0]
4.     for(i=1; i<N; i++){
5.         if(A[i] > max){
6.             max = A[i];
7.         }
8.     }
9.     int count = 0
10.    for(i=0; i<N; i++){
11.        if(A[i] == max){
12.            count += 1
13.        }
```

A = [2 9 11 7 5 5]
0 1 2 3 4 5

N = 6

max 2 → 9 → 11

i 1 2 3 4 5 6

count 0 → 1

i = 0 1 2 3 4 5 6

ans = 6 − 1

5 ← ans

```
14.     }
15.     int ans = N - count
16.     return ans;
    }
```

TC = O(n)
SC = O(1)

H.W. = try to solve this by iterating only once

Q. Given N array elements, check if there exists a pair (i, j) such that A[i] + A[j] = K and i != j.

Note: i and j are indices. K is given sum.

ex. A = [3 -2 1 4 3 6 8], K = 10, ans = true

ex. A = [2 4 -3 7], K = 5, ans = false

A = [2 4 -3 7], K = 8, ans = false

Quiz A = [3 5 2 1 -3 7 8 15 6 13]
0 1 2 3 4 5 6 7 8 9

Sum = 10 Ans = true

= (3,7), (2,8), (-3,13) (8,2) = ?

A = [5 5], Sum = 10 , Ans = true

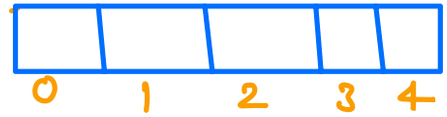
⇒ Brute force soln: check all the pairs ⇒ N*(N-1)

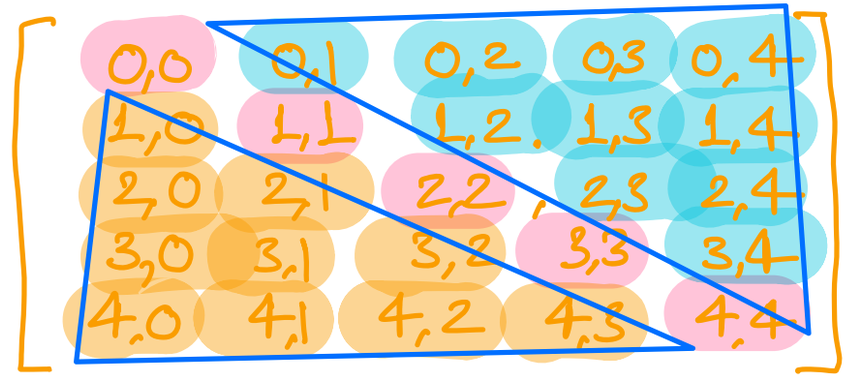
if sum == K, return true

```
boolean checkPairs ( A, K ) {
    int N = A.size();
    for (i=0; i<N; i++){
        for ( J=0; J<N; J++ ){
            if (i != J) {
                if (A[i]+A[J] == K){
                    return true;
                }
            } }
    }
    return false
}
```

TC = $O(N^2)$

SC = $O(1)$





A = [5 3], K = 10

initially

i = 0 → J (0→4)

i = 1 → J (0→4)

optimising the code

i = 0, → J [1-4)
i = 1, → J [2-4)
i = 2, → J [3-4)

```
boolean checkPairs(A, K){
    int N = A.size()
    for (i=0; i<N; i++){
        for (J= i+1; J<N; J++){
            if (A[i] + A[J] == K){
                return true;
            }
        }
    }
    return false;
}
```

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 +

TC = $O(N^2)$
SC = $O(1)$

$$\#\text{iterations} = (N-1) + N-2 + N-3 + \dots + (N-N)$$

$$= \sum_{j=0}^{N-1} j = \frac{(n-1)(n-1+1)}{2}$$

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

Q. Given an array, Reverse entire array. S.C ⇒ O(1)

Note: array itself should change

ex: [2 4 3 1] | [1 3 4 2]

Soln approach → swap the elements

ex: [9 -11 3 4 7 8 6 5 101]

0 1 2 3 4 5 6 7 8

[101 5 6 8 7 4 3 -11 9]

i J

0 < 8

1 < 7

2 < 6

3 < 5

4 = 4

5 > 3

swap (i, J)

i → (0 → N-1) ✗

i → (0 → (N-1)/2)

Until i<J then swap

6 > 2
7 > 1
8 > 0

```
void reverseArray (A) {
int N = A.sizeof();
i = 0, J = N-1;
while (i < J) {
    int temp = A[i]
    A[i] = A[J]
    A[J] = temp;
    i++;
    J--;
}
}
```

TC = O(N)
SC = O(1)

Q. Given N elements and Si (start index) and ei (end index), reverse the array from $S_i \to e_i$ ($S_i <= e_i$)

ex: A = [3 4 2 -1 9 5 2 7], Si = 4, ei = 6
(indices: 0 1 2 3 4 5 6 7)

A* = [3, 4 2 -1 2 5 9 7) (am)

// Reverse sub-array.

```
void rererse PartialArray(A, Si, ei)
    int N = A.sizeO;
    i = Si, J = ei

    while (i<J){
        int temp = A[i]

        A[i] = A[J]
        A[J] = temp;

        i++;
        J--;
    }
```

TC = O(N) → O(ei - Si)
SC = O(1)

}

Q. Given N size array, rotate the array from last to first by K times. (S.C ⇒ O(1))

Google, Amazon

10:52

ex

$A = [3\ -2\ 1\ 4\ 6\ 9\ 8]$, $K = 3$

(indices: 0 1 2 3 4 5 6)

$= [8\ 3\ -2\ 1\ 4\ 6\ 9]$, $K = 1$

$= [9\ 8\ 3\ -2\ 1\ 4\ 6]$, $K = 2$

$= [6\ 9\ 8\ 3\ -2\ 1\ 4]$ ans

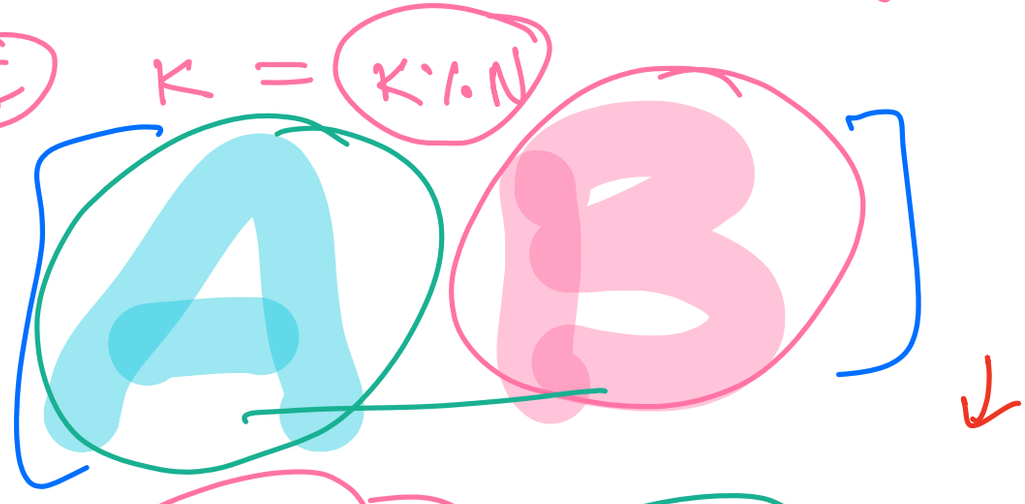
N	A	K	Ans.
0	[ ]	0	[ ]
0	[ ]	1-100	[ ]
1	[7]	~~0~~ 1, 2	[7]
1	[a]	y	[a]
2	[a, b]	1	[b, a]
2	[a, b]	2	[a, b]

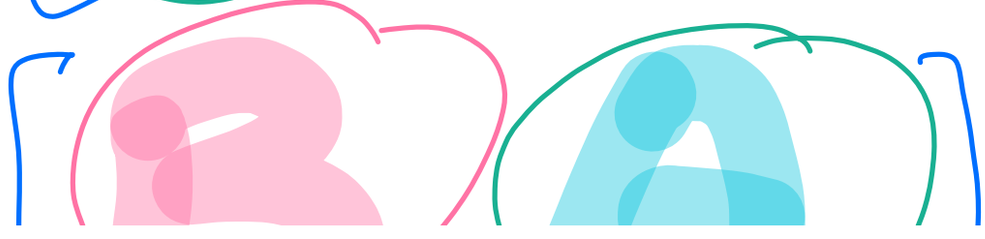
2	[a, b]	5	[b, a]
5	[a b c d e]	2	[d e a b c]
5	[a b c d e]	4	[b c d e a]
5	[a b c d e]	6	[e a b c d]

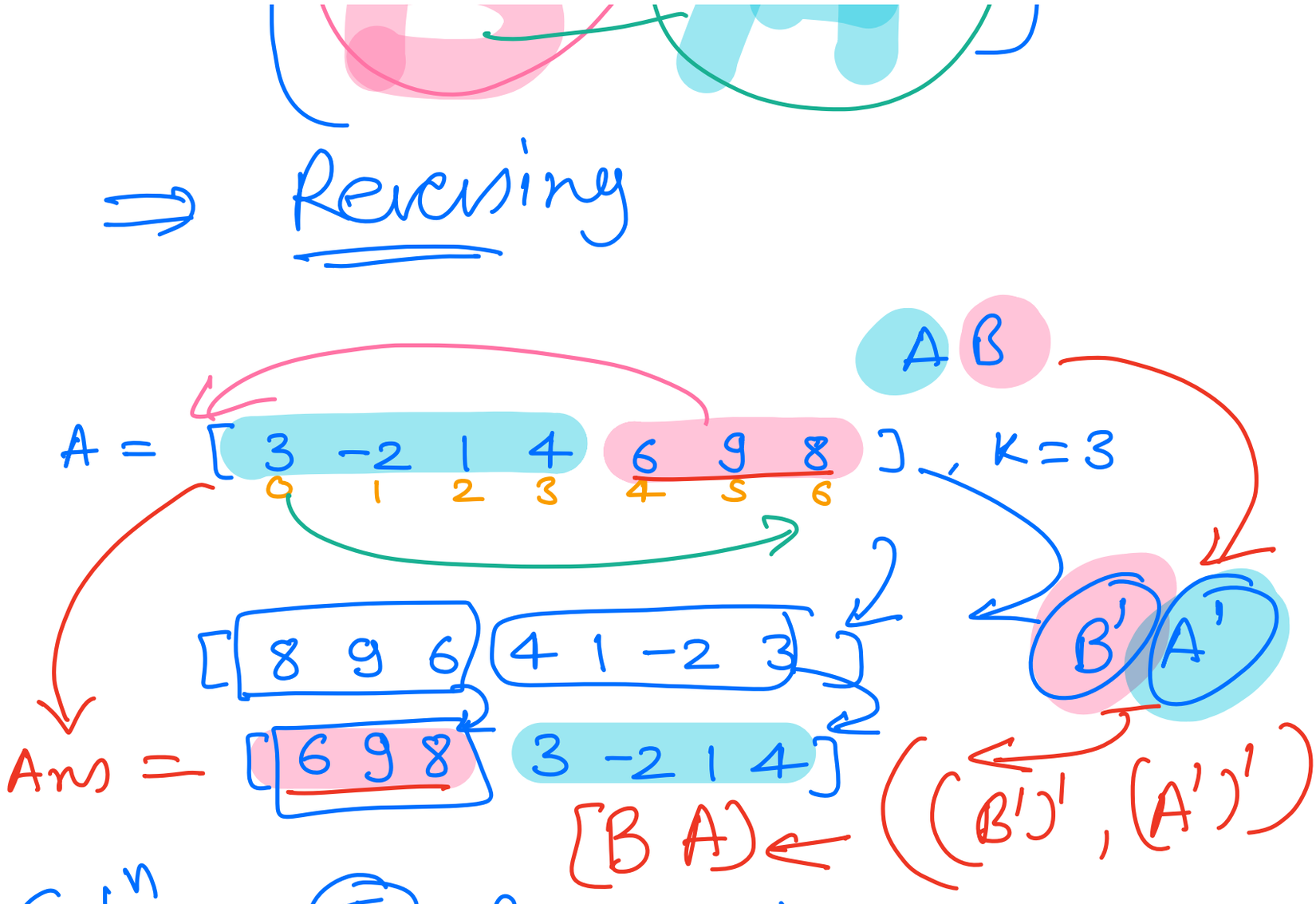
Obsⁿ (I) If K=N don't change, K=0 ✓

(II) K = K%N

thinking






⇒ Reversing
A B
A = [3 -2 1 4 6 9 8] , K = 3
0 1 2 3 4 5 6
[8 9 6 4 1 -2 3]
B' A'
Ans = [6 9 8 3 -2 1 4]
[B A]
((B')', (A')')
Soln
I Reverse the array
II reverse first k elements

(III) reverse second chunk of the array [K, N-1].

```
1. void rotateArray(A, K) {
2.     int N = A.size();
3.     K = K % N;
4.     reverseArray(A)
5.     reversePartialArray(A, 0, K-1)
6.     reversePartialArray(A, K, N-1)
   }
```

A = [3 2 7 5 4]
0 1 2 3 4

K = 14, 2, 99 → 4, 2, 4

Ans = [4 3 2 7 5] K = 1

= [5 4 3 2 7] K = 2

Dry Run N = 5, K = 2 → 2 % 5 = 2 → 2

rA = [4 5 7 2 3] → (#5 in the code)

= [5 4 7 2 3] → (#6 in the code)

Am = [5 4 3 2 7]

K = 14

PSP ⇒ Problem solving percentage

⇒ ( ) ← open problem.

Q1 try to unlock homework.

PSP → Just for you → indicates your learning

PSP >80%

Q.2 Big-O

theta Θ

omega Ω

intuition

$O(n) \leq O(n^2)$

Q

```
for (i = N/2; i<=N; i++) {
    for (J = 2; J<=N; J=J*2) {
        K = K + N/2
    }
}
```

	i	J	# iterations
	$N/2$	$[2, 4, 8, \dots 2^k]$	$\log_2 N$
Sum	$(N/2+1)$	$[2^1 \dots 2^k]$	$\log N$
	⋮		$\log N$

(N) $[2^1 \text{ —— } 2^k)$ $\log N$

$(\frac{N}{2} \to N) \Rightarrow \frac{N}{2} * \log N$

$\Rightarrow O(N \log N)$

Q

```
for (i=0; i < pow(2,N); i++) {
    int j = i
    while (j > 0){
        j -= 1
    }
}
```

i	j	#iteration
0	[]	0
1	[1]	1
2	[1,2]	2
3	[1,2,3]	3
⋮		
$2^N - 1$	$[1 \ldots 2^N - 1]$	$(2^N - 1)$

$(1 + 2 + 3 + \ldots + 2^N - 1)$

$\to N$

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

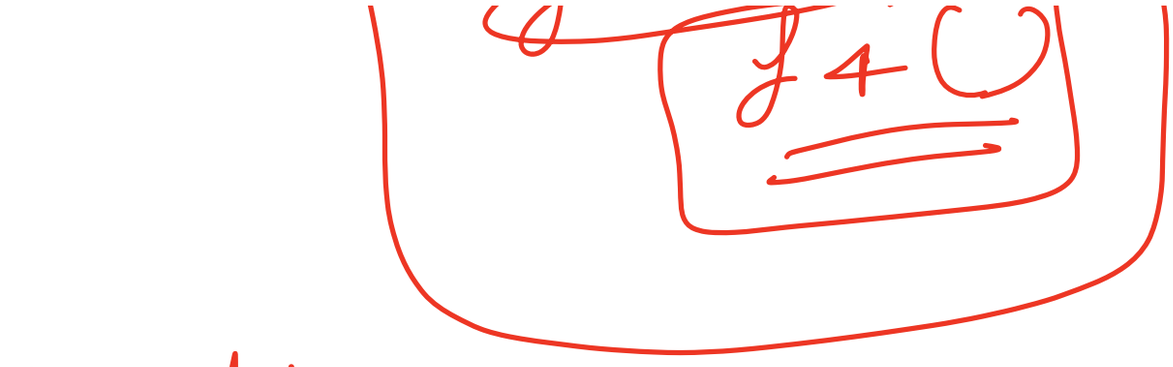
$\Rightarrow \frac{(2^N - 1) * (2^N + 1)}{2}$

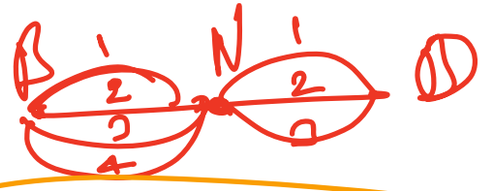
$\Rightarrow \frac{2^N * 2^N - 2^N}{2}$

$\Rightarrow \frac{2^{2N} - 2^N}{2}$ $\quad 2^{2N} = 4^N$

$\Rightarrow \frac{(2^2)^{N^2}}{2} \Rightarrow \frac{4^N}{2} = 4^N$

```
f1 ( ) {
    f2 ( ) {
        f3 ( )
        ter( ⟶ 2 ~~k~~
```





$B \to D \Rightarrow 4 \times 3 = \underline{12}$

```
while (N>0){
    N= N/3
}
```

$N \to \frac{N}{3^0} \to \frac{N}{3^1} \to \frac{N}{3^2} \to \frac{N}{27}$

$\frac{N}{9}$

$\approx K$ ↓ iteration

$$\frac{N}{3^K} \geq 1$$

$$\Rightarrow \quad N \geq 3^K$$

$$\Rightarrow \quad 3^k \leq N$$

$$\Rightarrow \quad \log_3 3^k \leq \log_3 N$$

$$\Rightarrow \quad k \leq \log N$$