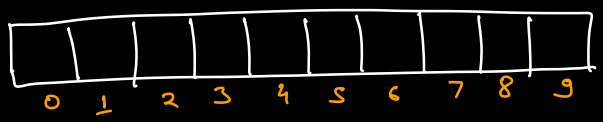
- Array usage
- Basic problems
- Subarrays

C / C++ / Java / C# | Python / JS / Ruby

int arr[10];



int arr[N] ⇒ [0, N-1]

* Why array index start with 0?

Given an array
Print it.

```
for (i=0; i < n ; i++){
    [Print( arr[i]).
]
```

TC : O(N)

O(1) } Random Access

Arrays / Linked list / BT / Graphs / Heaps

Q. Given an array count no. of elements that have atleast 1 element greater than themselves.

arr : { 3, -2, 6, 8, 4, 8, 5 } => 5

✓ ✓ ✓ ✗ ✓ ✗ ✓

App 1

- Sort the array
- Return everything except last element. ✗

-2, 3, 4, 5, 6, 8, 8

TC : O(N log N)

App 2

Obs 1 • The max no. is the only no. not satisfying this condition.

- All elements except the max should be counted.

max → max of array.

```
ans = 0;
for(i=0; i<N; i++){
    if(arr[i] != max){
        ans++;
    }
}
```

FindMax → O(N)
FindAns → O(N)

TC : O(N) + O(N) = O(N)

HW => Implement in a single loop.

Q Given an array. Count the no. of pairs (a[i], a[j])

such that $a[i] + a[j] = K$ $\quad i \neq j$

[No built in fn / library
No extra space.

ex : 3, -2, 1, 4, 3, 6, 8

K = 10

→ 1

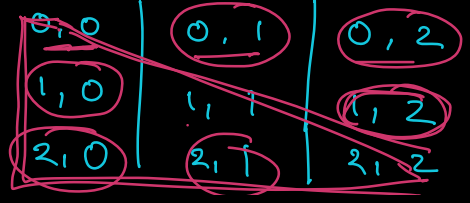
Quiz 3, 5, 2, 1, -3, 7, 8, 15, 6, 13

3, 7
2, 8
-3, 13] 3

Quiz 2, 7, 3, 14, 6, 1, 0, 10, 14

14, 6
6, 14] 2

```
cnt = 0;
for (i = 0; i < N; i++) {
    for (j = i+1; j < N; j++) {
        if (a[i] + a[j] == K && i != j)
            cnt++;
    }
}
```



i	j	No. of iterations
0	[1, N-1]	N-1
1	[2, N-1]	N-2
2	[3, N-1]	N-3
N-1	—	0

N-1 : 2 : 1 : 0 → AP

$S = 0 + 1 + 2 + 3 \text{ ---- } N-1$

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

TC : $O(N^2)$

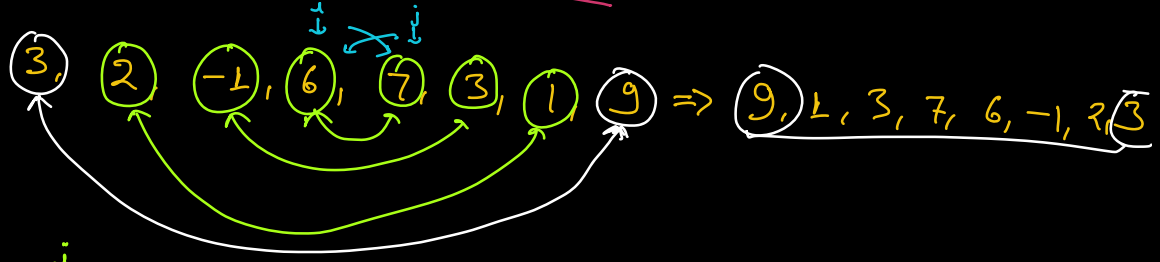
Expected TC : $O(N)$ ($O(N)$ extra space) $\Rightarrow$ HashMap class

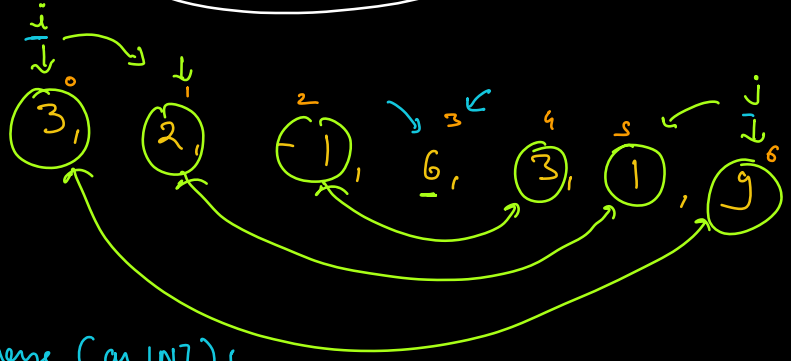
or

$O(N \log N)$ ($O(1)$ extra space) $\Rightarrow$ Two Pointer class

Q Given an array. Reverse the array.

Without using any extra space. $\Rightarrow$ SC = $O(1)$






```
Reverse (arr[N]){
  i = 0; j = n-1;
  while (i<j) {
    Swap(arr[i], arr[j]);
    i++; j--;
  }
}
```

```
int temp = a;
a = b;
b = temp;
```

```
a, b = b, a  // Python
```

```
a = a + b;
b = a - b;   // a+b-b = a
a = a - b    // a+b-a = b
```

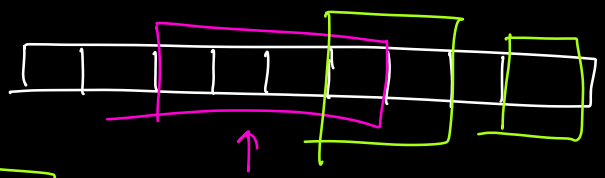
→ OVERFLOW

```
a = a ^ b
b = a ^ b    // a^b^b = a
a = a ^ b    // a^b^a = b
```

O(N/2) ✗ → O(N)

TC-1 ✗
TC-2 ✗

Subarray



⇒ 4, 5, 1, 9, 0, 2, 3, 5

5 ✓
4, 5, 1, 0 ✗
9, 0, 2, 3 ✓
4, 5, 1 ✓

- An array is a sub-array of itself.
- An empty array is a sub-array of all the arrays.
- A single element is also a subarray.

Subarray ⟹ Non-empty subarrays (Most of the time)

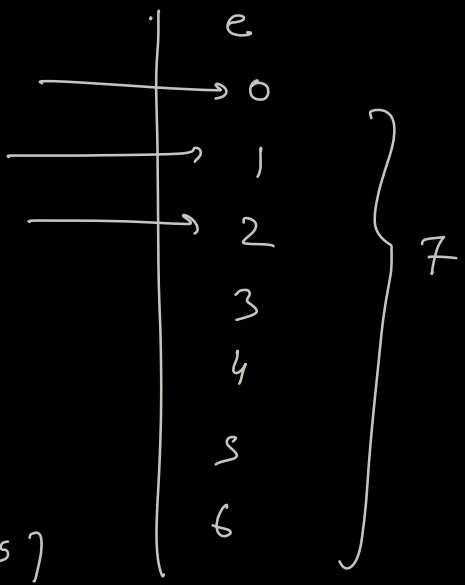
Q

A : [4, 2, 10, 3, 12, -2, 15]
0 1 2 3 4 5 6

How many sub-arrays start from index 0.

s = 0

[4]
[4, 2]
[4, 2, 10]
[4, 2, 10, 3]
[4, 2, 10, 3, 12]
[4, 2, 10, 3, 12, -2]
[4, 2, 10, 3, 12, -2, 15]



s = 1 e

[2] ⟶ 1
[2, 10] ⟶ 2
[2, 10, 3] ⟶ 3
[2, 10, 3, 12] ⟶ 4
[2, 10, 3, 12, -2] → 5
[2, 10, 3, 12, -2, 15] → 6

⇒ 6

Total Non-Empty subarrays;

Count of SA starting from index 0 ⇒ N

+

Count of SA starting from index 1 ⇒ N-1

+

Count of SA starting from index 2 ⇒ N-2

⋮

Count of SA starting from index N-1 ⇒ 1

$$1 + 2 + 3 \cdots (N-2) + (N-1) + N$$

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

Break till 10:35p

⇒ Interview Bit → Practice

⇒ Leetcode

Q Given an array. Print the sums of all the subarrays. → (Non-empty)

A : 3, 1, 2

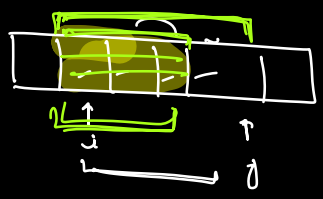
→ [3] → [3] ✓ 1 → [1] ✓ 2 → [2] ✓

3, 1 → [4] ✓ 1, 2 → [3] ✓

3, 1, 2 → [6] ✓

⇒ 3, 4, 6, 1, 3, 2

```
for (i=0; i<N; i++) {
    for (j=i; j<N; j++) {
        // i → start of subarray
        // j → end of subarray
        sum = 0;
        for (k=i; k<=j; k++) {
            sum = sum + a[k];
        }
        Print (sum);
    }
}
```



Brute Force ⇓ How to iterate on all the subarrays.

TC : $O(N^3)$

Print (sum); ⟹ $\frac{N(N+1)}{2}$ times

[Can we do better than $O(N^2)$? ✗

Can we do this in $O(N^2)$? ✓

HW : 30 min

Q Given s & e. Reverse the subarray from s to e.

3, 2, -1, 4, 6, 8, 9, 2

0 1 2 3 4 5 6 7

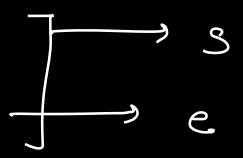
s ⇒ 2

e = 5

3, 2, 8, 6, 4, -1, 9, 2

i = 0

j = n-1



Q How many sub-arrays of length 3

4, 2, 10, 3, 12, -2, 15

⇒ 5

K	No of sub arrays
1	N
2	N-1
3	N-2
4	N-3
K	N-(K-1) ⇒ N-K+1

C++ → vector

Java → ArrayList

C# → List

Python → list

JS → Array

Ruby → Array

Dynamic Array

insert()
Size()
get(i)

int arr[10] → 10 Elements.

```
ArrayList<Integer> list = new ArrayList<Int>();

list.add(10);

for (i=0; i < l.size(); i++) {

}
```

Time Complexity of random index access → O(1)

Insert → O(1) [Amortized]

Future classes

→ Write code on →
→ Copy paste to Scaler Ide
→ Analyze mistakes
→ Make sure of not repeating them.

- Google doc ✓
- Note Pad ✓
- WordPad ✓
- Editor without any feature ✓

Production Ready Code

Logic Building

1. Understand the problem
   Check out tests
2. 30 - 40 mins [Hint] → Brute force
   → Try to optimize
3. Understand the sol.
   → Analyze the obs / fact / points that you had missed.