First Index And Last Index

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
arr =	1	5	10	15	22	33	33	33	33	33	40	42	55	66	77

d = 33

$\boxed{33 == 33}$

Fi = ~~-1~~ 5

Count = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

Li = -1

Li = Fi + Count -1

```
if(arr[i] == d){
    if(Fi == -1){
        Fi = i;
    }
    Count++;
}
```

num = 33

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
arr:	1	5	10	15	22	33	33	33	33	33	40	42	55	66	77

fi = ~~-1~~ 5

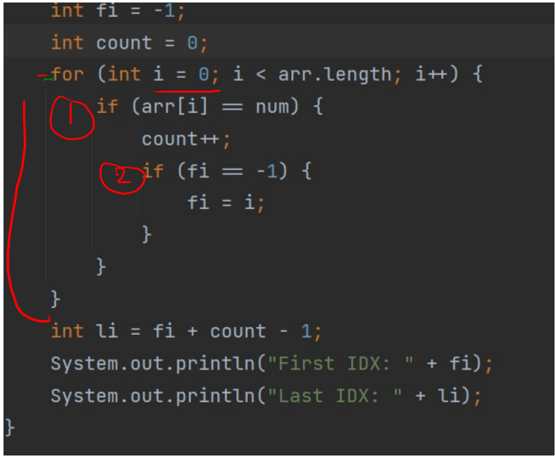
Count = ~~0~~ ~~1~~ ~~2~~ ~~3~~ ~~4~~ 5

i = 5
↳ 33 == 33 ✓

i = 6
33 == 33 ✓

i = 7 33 == 33 ✓

Li = fi + Count - 1
5 + 5 - 1 = 9



```
    int fi = -1;
    int count = 0;
    for (int i = 0; i < arr.length; i++) {
        if (arr[i] == num) {
            count++;
            if (fi == -1) {
                fi = i;
            }
        }
    }
    int li = fi + count - 1;
    System.out.println("First IDX: " + fi);
    System.out.println("Last IDX: " + li);
}
```

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
arr =	1	5	10	15	22	33	33	33	33	33	40	42	55	66	77

num = 33

i = 5 -> 33 == 33 ✓

i = 6 = 33 == 33 ✓

i = 7 -> 33 == 33 ✓

fi = ~~-1~~ 5

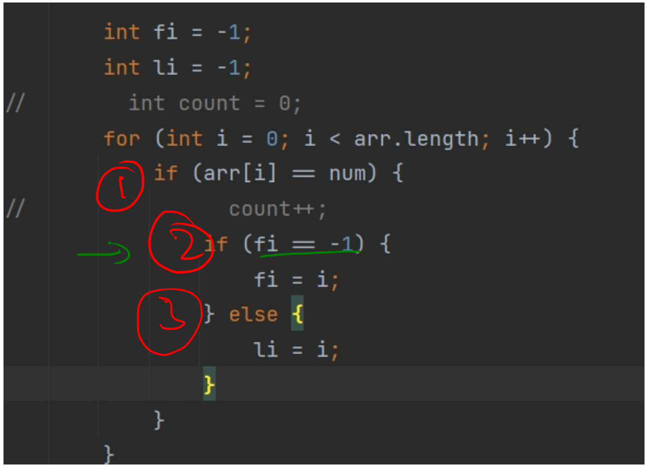
Li = ~~-1~~ ~~6~~
~~7~~
8
9

num = 15

fi = ~~-1~~ 3

li = ~~-1~~ 3

edge case



```
        int fi = -1;
        int li = -1;
//        int count = 0;
        for (int i = 0; i < arr.length; i++) {
      1   if (arr[i] == num) {
//            count++;
      ->  2   if (fi == -1) {
                fi = i;
          3   } else {
                li = i;
            }
        }
    }
```

1). Reverse Array

n → i/p → size of arr

1, 2, 3 — — n numbers

arr =

0	1	2	3	4	5
1	2	3	4	5	6

space
O(n)

→ m1 ⇒ o/p array
space

o/p ⇒ ↓

arr =

6	5	4	3	2	1

→ m2 = modify i/p
arr
O(1)
space

i = n-1 ⟶ ansidx => n-i

i = n-2 → n-(n-1) = 1

i = n-3 → n-(n-2) = 2

→ n-(n-3) → 3

i = 0

I/P

arr =>

0	1	2	3	4	5
1	2	3	4	5	6

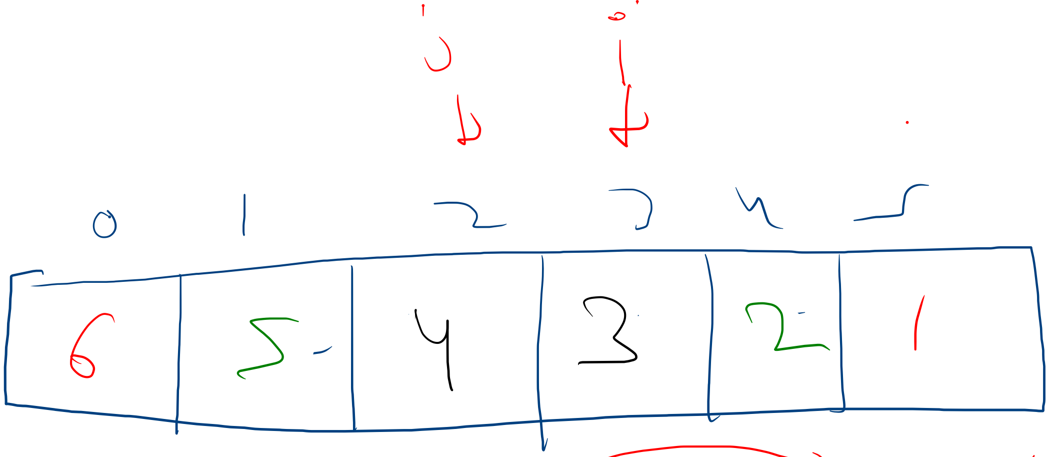
(i = arr.l-1;
i >= 0; i--)

new

O/P

arr =>

0	1	2	3	4	5
6	5	4	3	2	1



i = 0

j = arr.len - 1

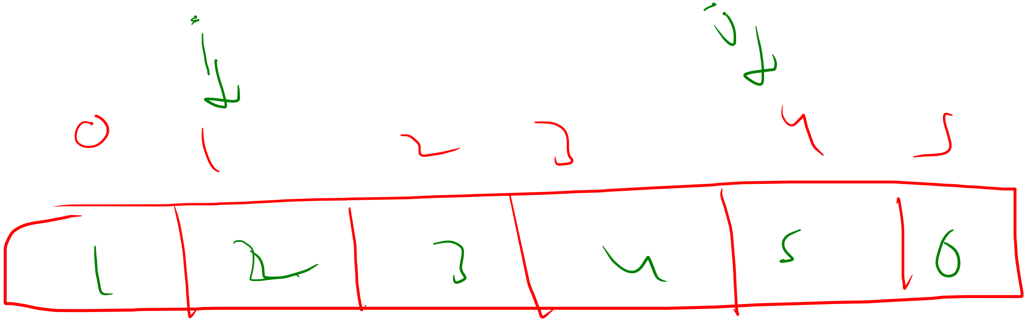
→ swap(i, j) → not built in

i ++

j --

i > j → break

Swap



arr[i] = ~~2~~ 5

arr[j] = ~~5~~ 2

temp = arr[i] → 2

arr[i] = arr[j]

arr[j] = temp

Space → using heap memory
comp

↓

new

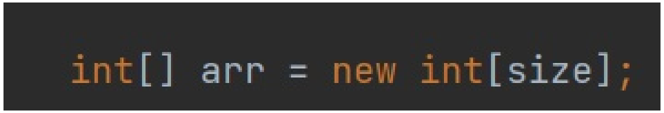
+

String

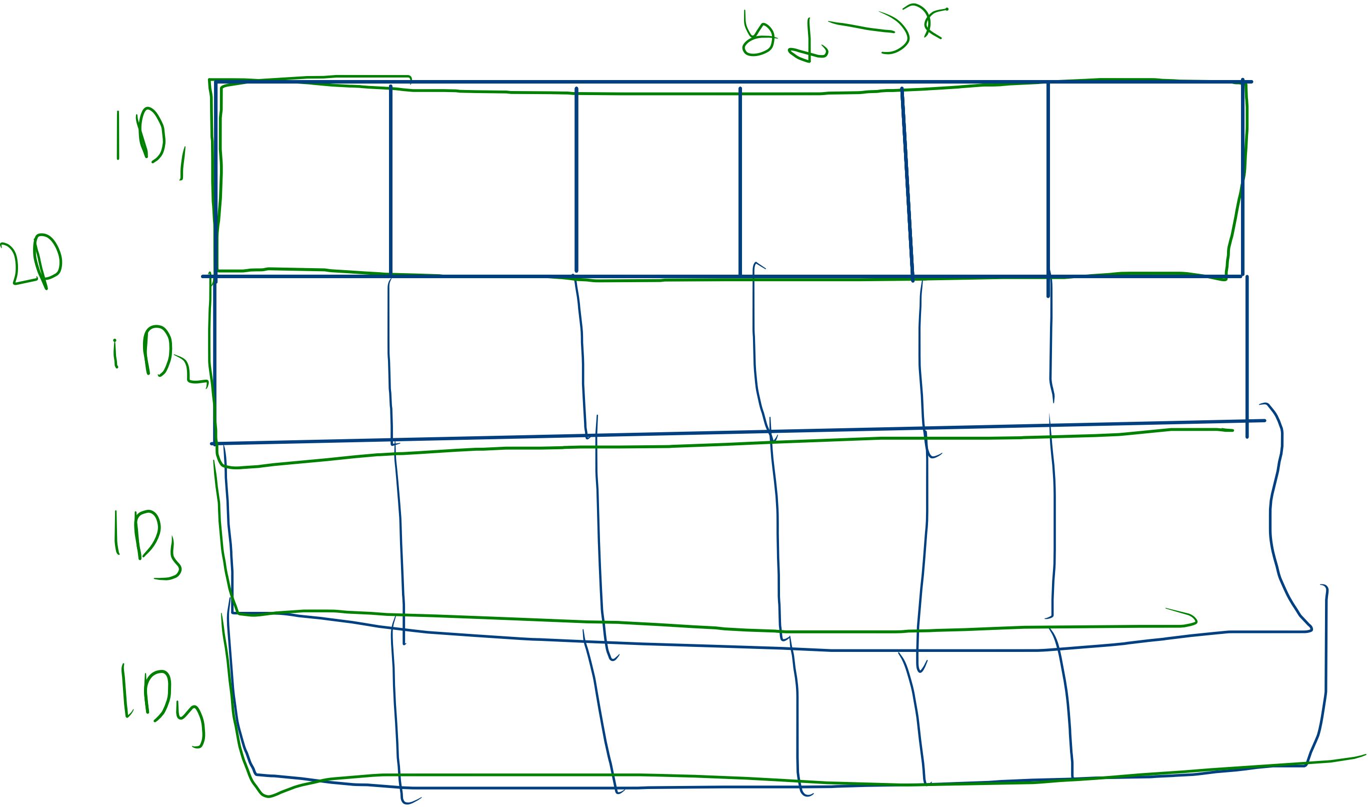
T.C

=

1D



```java
int[] arr = new int[size];
```



1D int[] arr = new int [5(2)]

2D int[][] arr = new int [row] [col]

row num ↓ col num ↓

arr[i] [j]

arr-

Col → row ↓	0	1	2	3	4
0	0,0	0,1	0,2	0,3	0,4
1	1,0	1,1	1,2	1,3	1,4
2	2,0	2,1	2,2	2,3	2,4
3	3,0	3,1	3,2	3,3	3,4
4	4,0	4,1	4,2	4,3	4,4

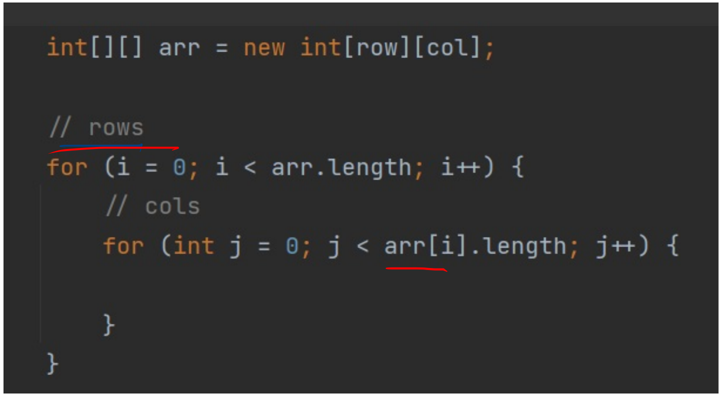
→ sq matrix

5X5 Total elmt = 5x5 = 25

0
1 →

Col → row ↓	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

5 × 5



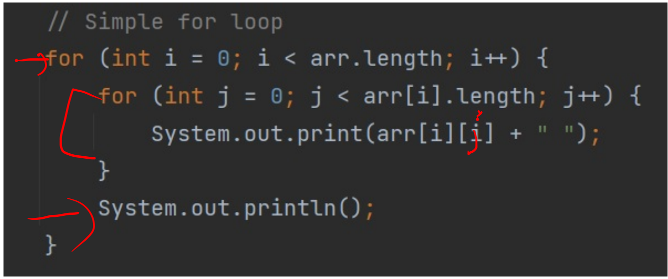
```
int[][] arr = new int[row][col];

// rows
for (i = 0; i < arr.length; i++) {
    // cols
    for (int j = 0; j < arr[i].length; j++) {

    }
}
```

arr(i)

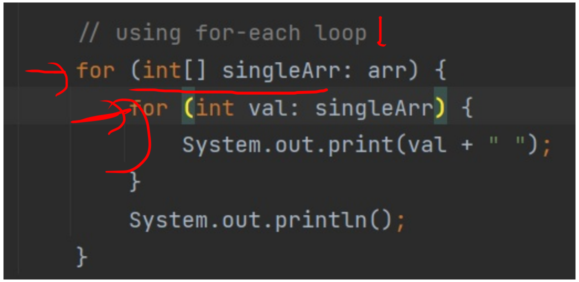
arr [i

Col → row ↓	0	1	2	3	4
0	1	2	3	4	5
1	6	7	8	9	10
2	11	12	13	14	15
3	16	17	18	19	20
4	21	22	23	24	25

5 × 5



```
// Simple for loop
for (int i = 0; i < arr.length; i++) {
    for (int j = 0; j < arr[i].length; j++) {
        System.out.print(arr[i][j] + " ");
    }
    System.out.println();
}
```

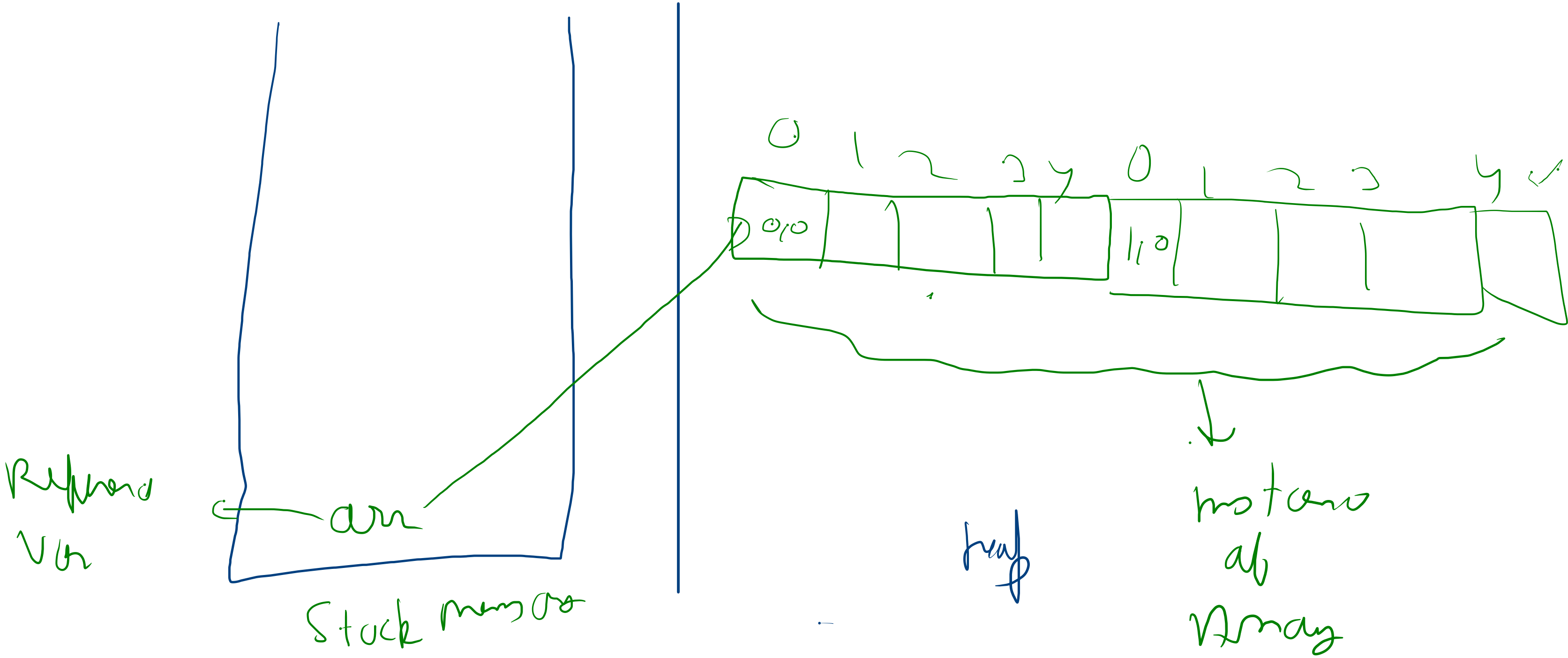


```
// using for-each loop
for (int[] singleArr: arr) {
    for (int val: singleArr) {
        System.out.print(val + " ");
    }
    System.out.println();
}
```

1 2 3 4 5

Memory Management of 2D Arr

```
int [] [] arr = new int [row] [col]
```


0 1 2 3 4 0 1 2 3 4
0,0
1,0
Reference Var
arr
Stack memory
heap
instance of Array