Hollow Sq

```
      1           5
1     *  *  *  *  *
2     *           *
3     *           *
4     *           *
5     *  *  *  *  *
```

```
i==1 || i==n  ⟶  print n times *
i!=1 || i!=n  ⟶
                  j==1 || j==n
                      ↳ print *
```

Core Subjects

1). OOPS → Youtube or GFG Blogs

2). Operating System

3). Database Management System (DBMS)

4). Basics of Computer Network

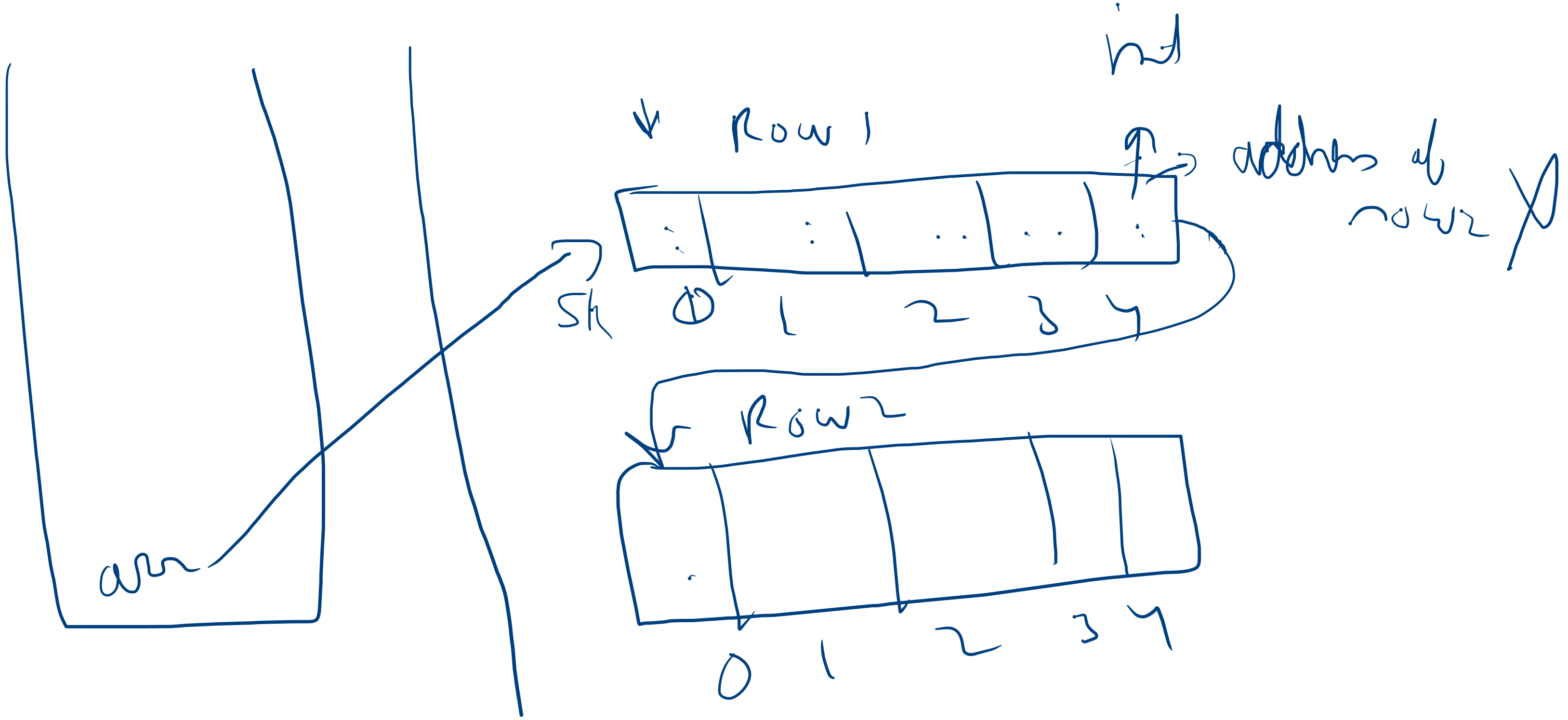
✓✓ imp

✓ DS Algo → ✓ Core Subjects → ✓ Technology

✓ web dev

blockchain

✓ AI (ML)


arr
5k
Row 1
0
1
2
3
4
int
address of
row2
Row2
0
1
2
3
4

UP Solving ? → v v Imp

↓

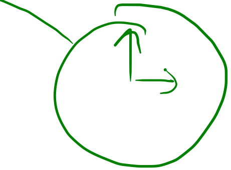
present

q1 ✓

q2 X

2hrs

1hr



⇒ Sol

30min

Sol^n

Logic only ✓

read code X X

Competitive Programming !

↓

+1

Contest

2hrs

3hrs

upsolve

Problem Solving skill

① Sub arrays ? : → Continuous part of original array having 1 or more elements

Total = $\frac{n(n+1)}{2}$

0	1	2	3	4	5	6
5	3	10	12	9	8	7

4

$\frac{7 \times \cancel{8}}{2} = 28$

5
5 3
5 3 10
5 3 10 12
5 3 10 12 9
5 3 10 12 9 8
5 3 10 12 9 8 7

3
3 10
3 10 12
3 10 12 9
3 10 12 9 8
3 10 12 9 8 7

10
10 12
10 12 9
10 12 9 8
10 12 9 8 7

12
12 9
12 9 8
12 9 8 7

9
9 8
9 8 7

8
8 7

7

Subset of array → one or more elements present in non-continuous manner but their order remain same

Total = $2^n = 2^7 = \underline{\underline{128}}$

Subset

0	1	2	3	4	5	6
5	3	10	12	9	8	7

idx y below Idx 3

→ 3 12 8 7

→ 5

→ 10 9 8

→ 3 12 9

Q: Given an array

→ Print all subarrays

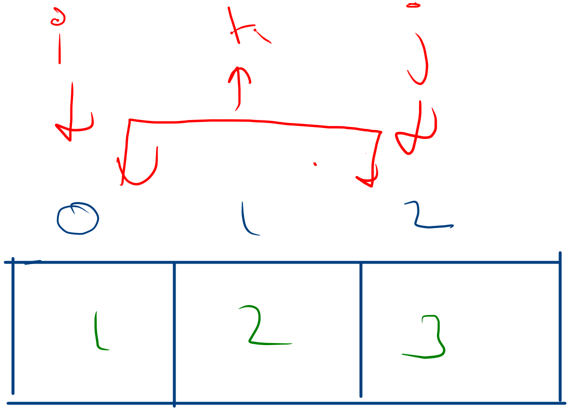
i = 0 to n-1

↓

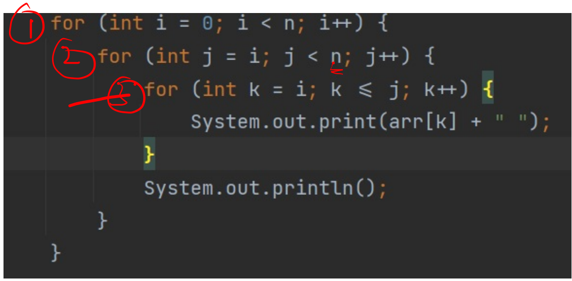
j = i to n-1

↓

k = i to j



One



```
for (int i = 0; i < n; i++) {
    for (int j = i; j < n; j++) {
        for (int k = i; k ≤ j; k++) {
            System.out.print(arr[k] + " ");
        }
        System.out.println();
    }
}
```

1

1 2

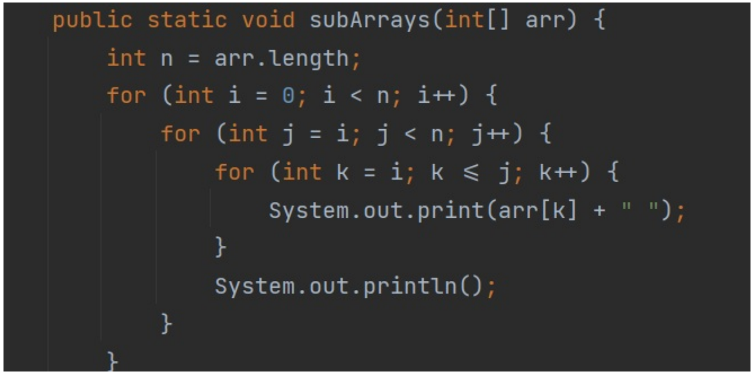
1 2 3

2

2 3

3

Never count I/P array in Space comp



```java
public static void subArrays(int[] arr) {
    int n = arr.length;
    for (int i = 0; i < n; i++) {
        for (int j = i; j < n; j++) {
            for (int k = i; k ≤ j; k++) {
                System.out.print(arr[k] + " ");
            }
            System.out.println();
        }
    }
```

$i = 0$ to $n-1$

$j = 0$ to $n-1$

$k = 0$ to $n-1$

$TC \Rightarrow O(\boxed{n^3})$

$SC \Rightarrow O(1)$

Q: Rotate an Array

arr[]

0	1	2	3	4
1	2	3	4	5

k = 3

Rotat array 3 time 3 4 5 1 2

k = -1
2 3 4 5 1

k = -2
3 4 5 1 2

k = 1 5 1 2 3 4

k = 2 4 5 1 2 3

k = 3 3 4 5 1 2

k = 4
2 3 4 5 1

k = 5 1 2 3 4 5

k = 6 → 5 1 2 3 4

k = 7 → 4 5 1 2 3

i/. if +ve num k

$\Rightarrow k \% n$

ii/. if -ve num k

$\Rightarrow (-k \% n) + n$

arr →	0	1	2	3	4	
	1	2	3	4	5	
k = 1	5	1	2	3	4	✓
k = 2	4	5	1	2	3	✓
k = 3	3	4	5	1	2	✓
k = 4	2	3	4	5	1	✓
k = 5	1	2	3	4	5	✓
k = 6	5	4	3	2	1	
k = 7	4	5	3	2	1	

rotate k times

↳ k +ve or -ve

% of two

k = 6 ⇒ 6 % 5 = 1

k = 7 ⇒ 7 % 5 = 2

arr ↓	0	1	2	3	4
	1	2	3	4	5
k=-1	2	3	4	5	1
k=-2	3	4	5	1	2
k=-3	4	5	1	2	3
k=-4	5	1	2	3	4
k=-5	1	2	3	4	5
k=-6	2	3	4	5	1

% of -ve num

$k=-2 \Rightarrow -2\%5+5$

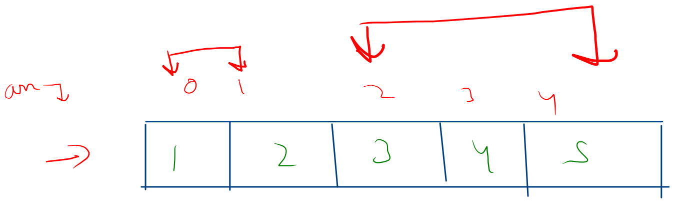
$= -2+5$

$= 3$

$k=-6 \Rightarrow -6\%5+5$

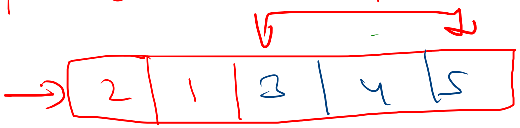
$-1+5$

$= 4$

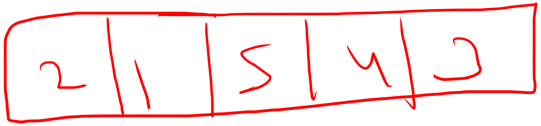


k=3   3 4 5 1 2

step 1 0 to n-1-k → reverse



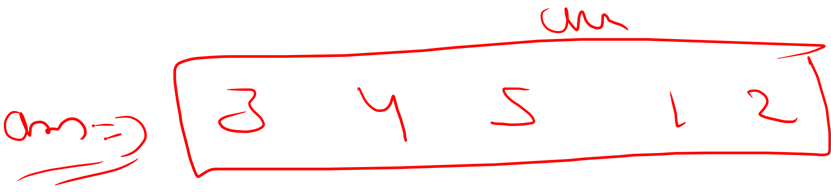
step 2 reverse n-k to n-1



① k = k % n

② if (k<0) k = k+n

step 3 → Reverse whole arr



arr
arr =>
3
4
5
1
2