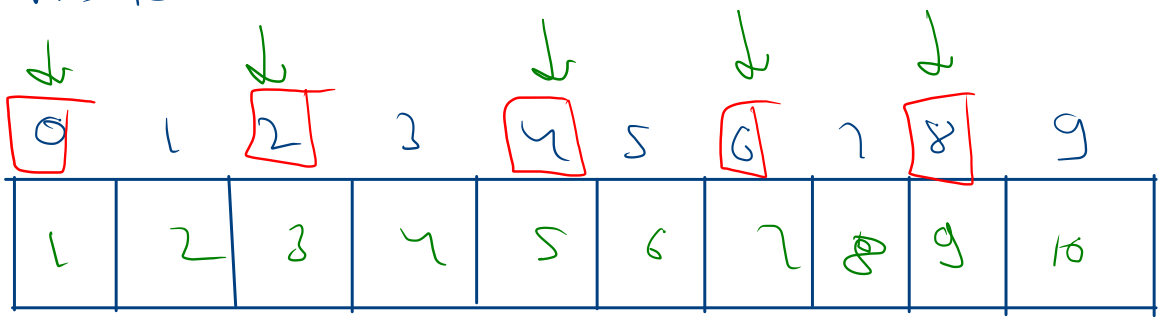
Print Alternate Array Elements Linewise (1 july)

I/P -> n -> num

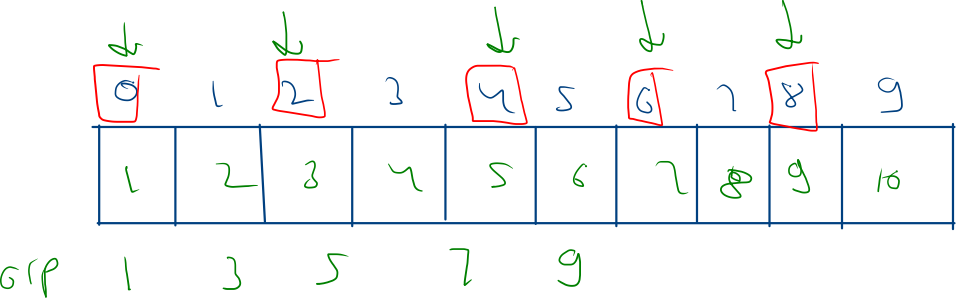
int arr -> n input in arr

Sample n = 10



i%2 == 0 -> even

O/P 1 3 5 7 9



i%2==0
↳ even

if(i%2==0){
sout(arr[i])
}

1 usage Eshan-Agarwal *

```java
public static void alternatePrint(int[] arr) {

    int n = arr.length;
    for (int i = 0; i < n; i++) {
        if (i % 2 == 0) {
            System.out.println(arr[i]);
        }
    }

}
```

Rotation of Arr

0	1	2	3	4
1	2	3	4	5

Arr =

r = 1	5	1	2	3	4
r = 2	4	5	1	2	3
r = 3	3	4	5	1	2
r = 4	2	3	4	5	1
r = 5	1	2	3	4	5
r = 6	5	1	2	3	4
r = -1	2	3	4	5	1

⇒ r = r % n
if (r < 0) {
⇒ r = r + n
}

-1 % 5 = -1 + 5 = 4

N = 256 1 to N

for (i = 0 to 256) {

basic =>

update cur cur

}

smart =>

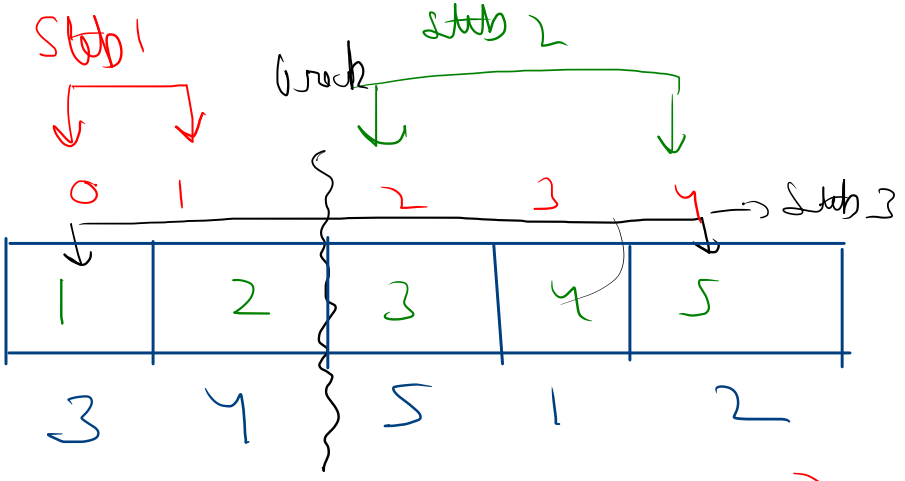
N = N - 1

if (N < 0) {

N = N + 1

}

→ 1 to n



arr =

r = 3

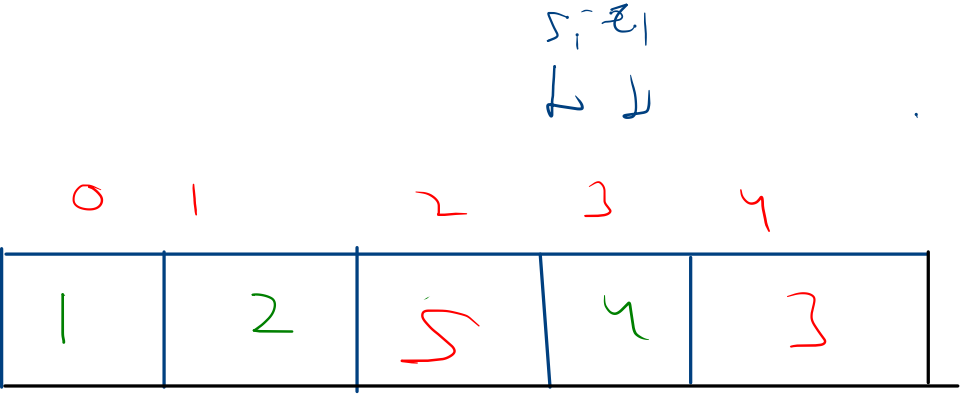
Logic → Stub 1 reverse (arr , 0 , n-1-r)

Stub 2 reverse (arr , n-r , n-1)

Stub 3 reverse (arr , 0 , n-1)

n-1-r =
5-1-3
= 4-3
= 1

n-r = 5-3 = 2



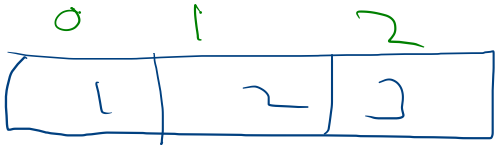

```
1 usage   new *
public static void rotateArr(int[] arr, int r) {
    int n = arr.length;
    // code here
    r = r % n;
    if (r < 0) {
        r += n;
    }
    reverse(arr, si: 0, ei: n - 1 - r);
    reverse(arr, si: n - r, ei: n - 1);
    reverse(arr, si: 0, ei: n - 1);
    printArr(arr);
}
```

Count Odd Pair (2 July)

```
1 usage  new *
public static void  countOddPair(int[] arr) {
    int n = arr.length;
    for (int i = 0; i < n; i++) {
        for (int j = i + 1; j < n; j++) {
            int sum = arr[i] + arr[j];
            if (sum % 2 == 1) { // odd
                System.out.println(arr[i] + " " + arr[j]);
                System.out.println(arr[j] + " " + arr[i]);
            }
        }
    }

}
```

for (i to n-1) {
for (j = i+1 to n-1)



Pairs => arr =

1 2

2 1

2 3

3 2

Sum = 1+2 = 3

2+3 = 5

Sum = arr(i) + arr(j)

if (sum % 2 == 1) {

Sout (arr(i) + " " + arr(j))

Sout (arr(j) + " " + arr(i))

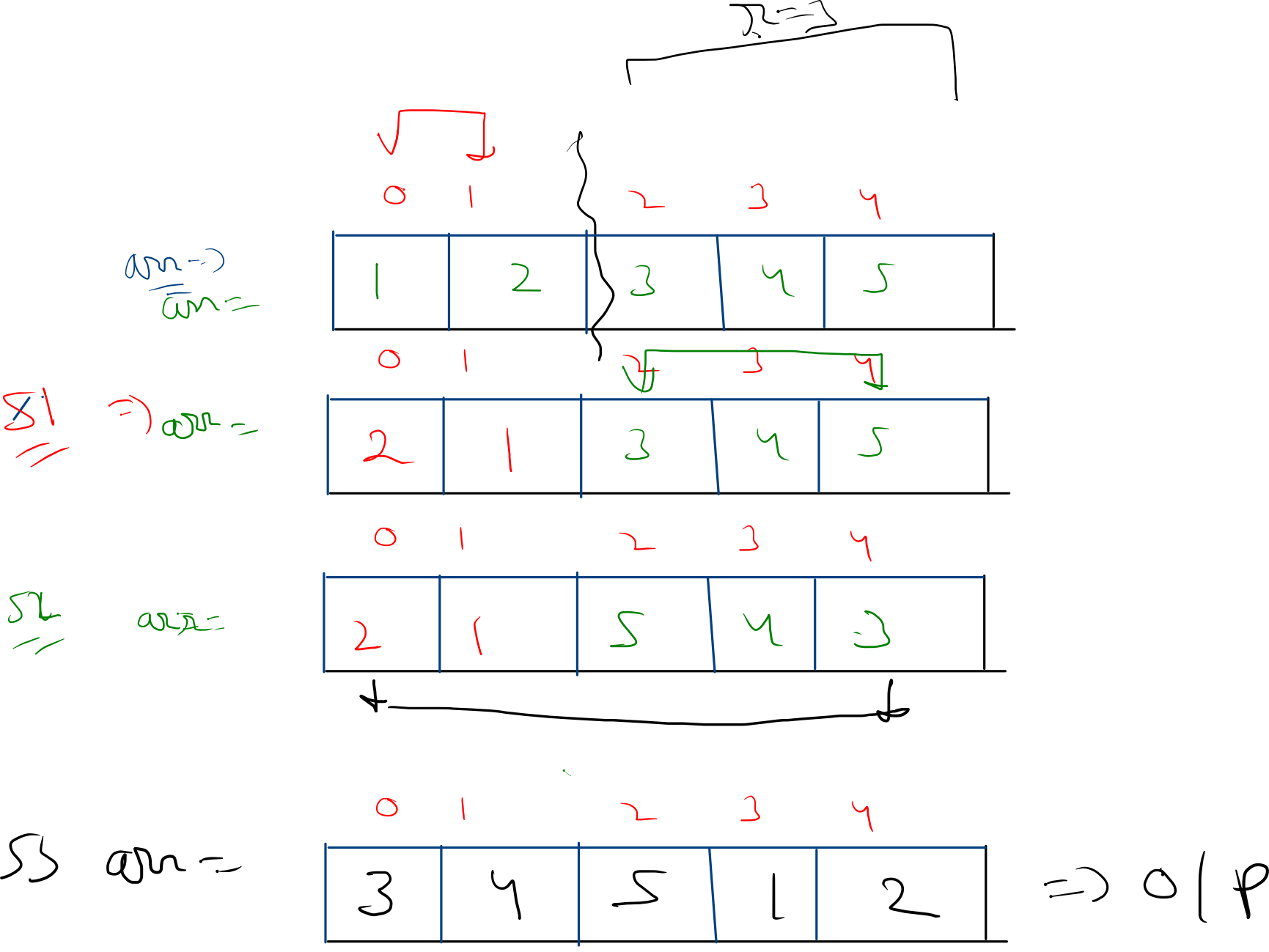
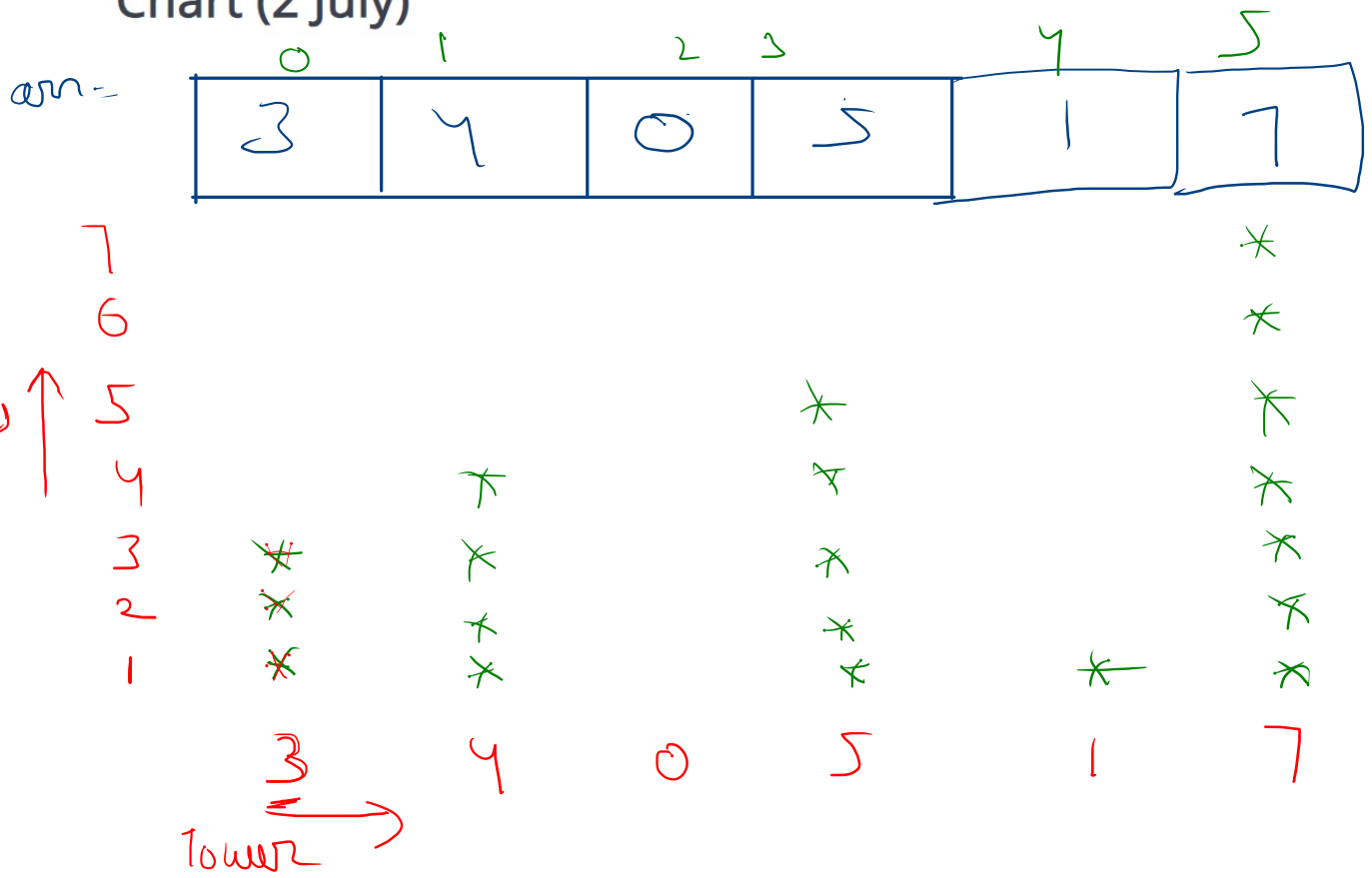
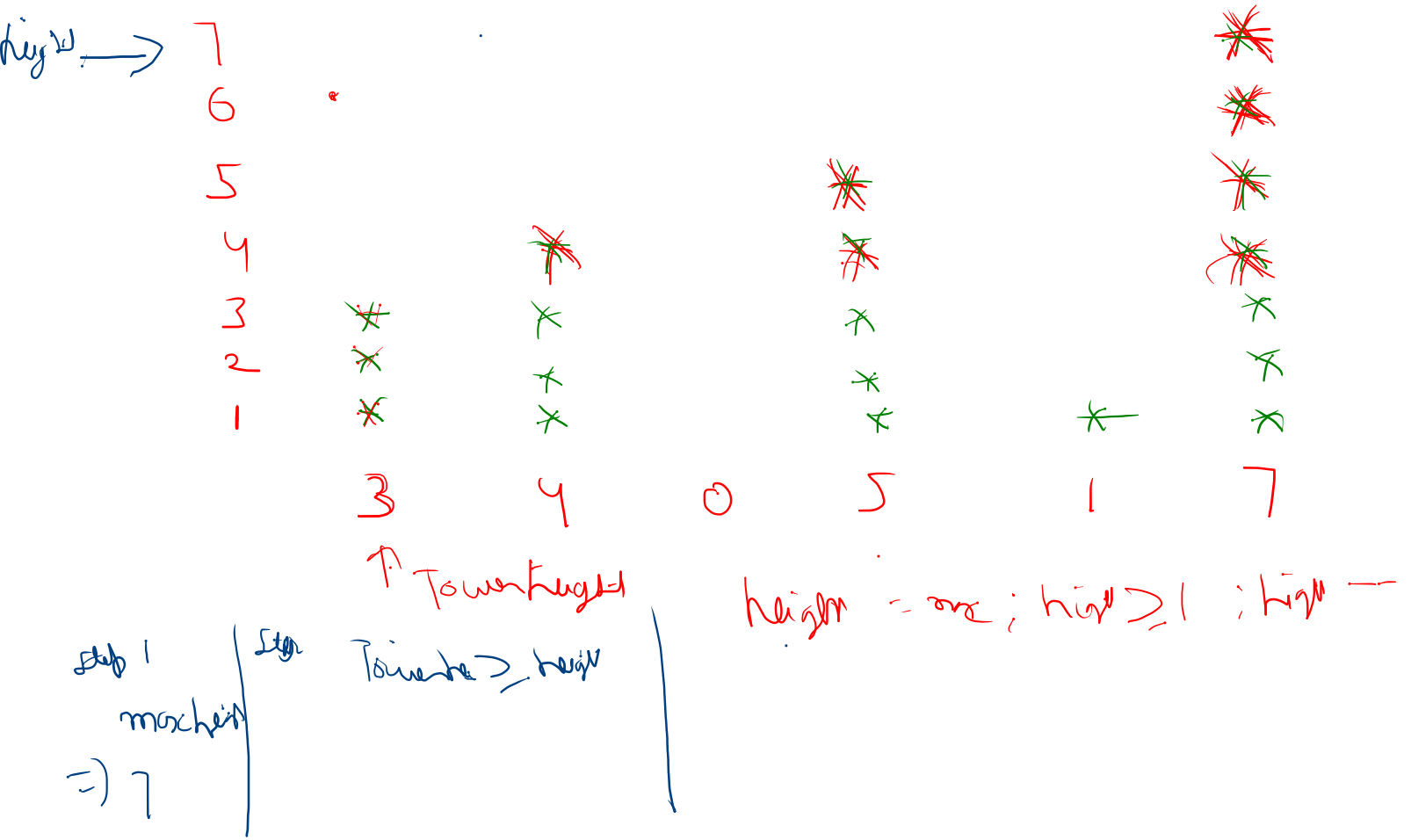
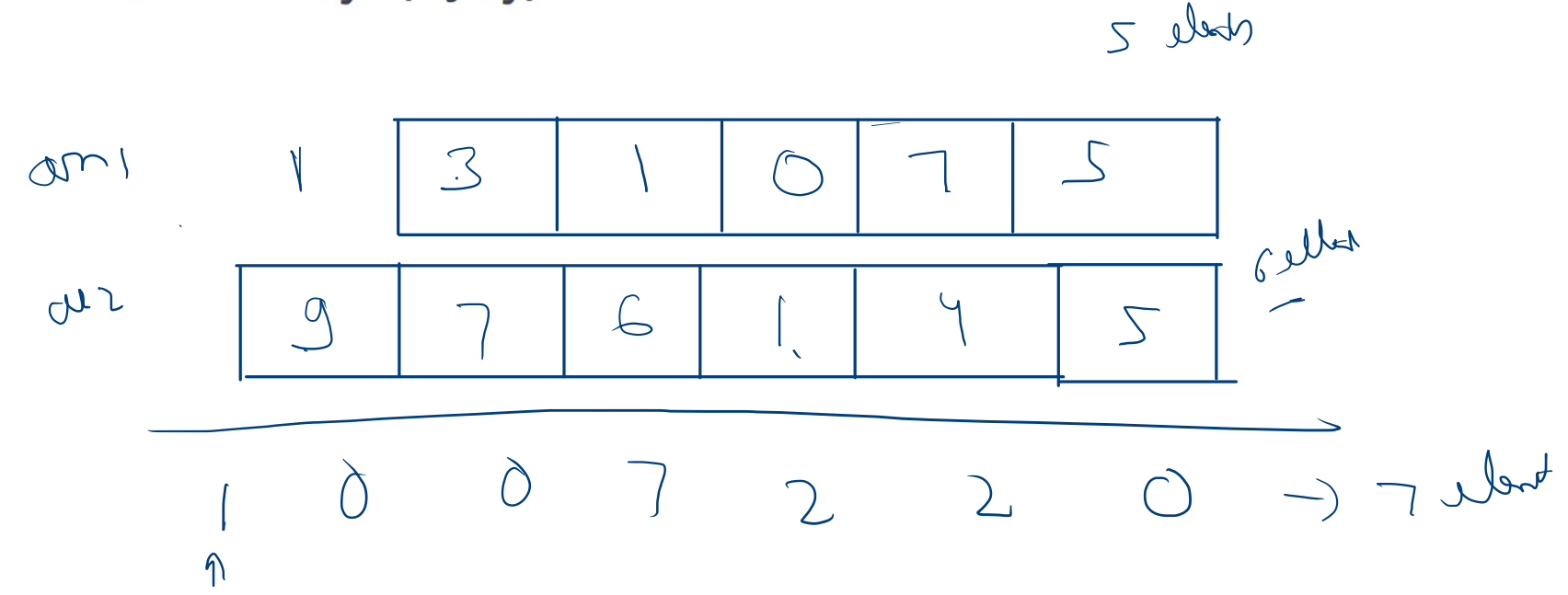

0 1 2 3 4
1 2 3 4 5
0 1 2 3 4
2 1 3 4 5
0 1 2 3 4
2 1 5 4 3
0 1 2 3 4
3 4 5 1 2

Chart (2 July)

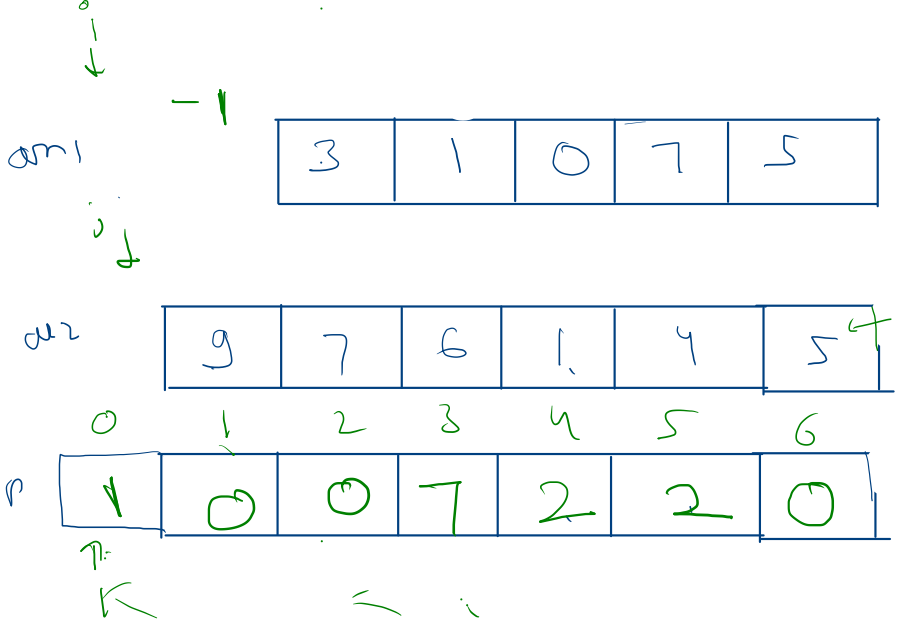



height
7
6
5
4
3
2
1
3
4
0
5
1
7
Tower height
height = 0 ; height ≥ 1 ; height
step 1
max height
=) 7
Step
Tower > height

Sum of Arrays (2 July)



O(n) ⇒ size (arr1.length, arr2.length) → max + 1



Sum = arr1[i] + arr2[j] + carry

carry = sum / 10

sum = sum % 10

arr[k] = sum

i--; j--; k--

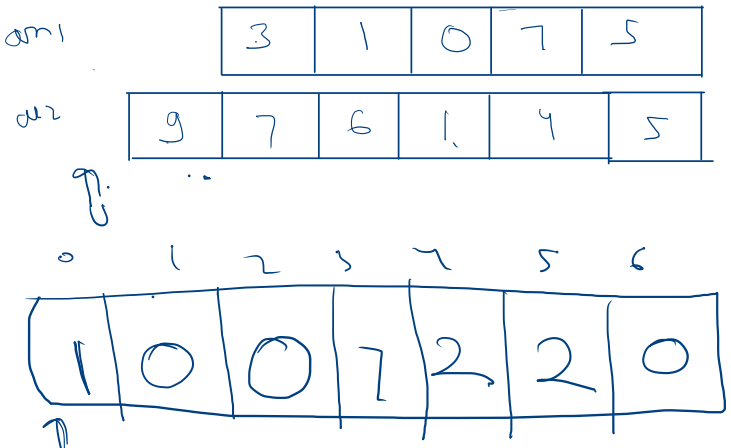
Carry = 1

0 0 + 1

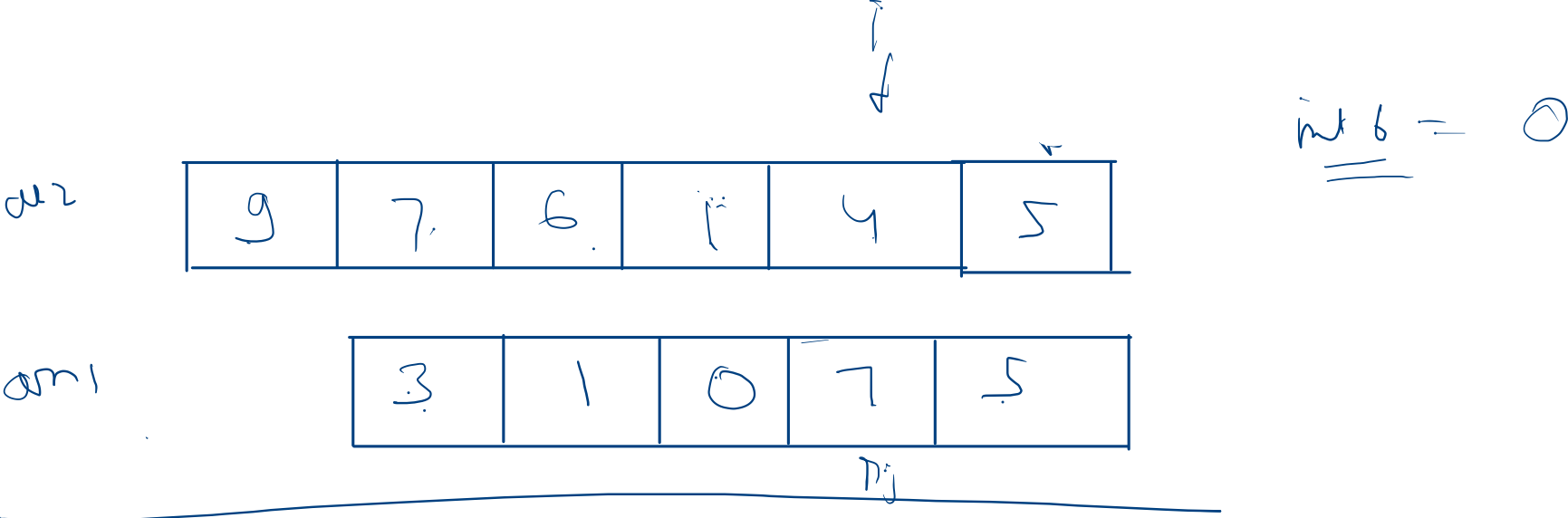
Sum = 9 + 1 = 10

c = 1

Sum = 0



```java
public static void addTwoArr(int[] arr1, int[] arr2) {
    int n1 = arr1.length;
    int n2 = arr2.length;
    int sizeAns = (n1 > n2) ? n1 + 1 : n2 + 1; // Ternary Operator
    int[] ans = new int[sizeAns];
    int i = n1 - 1;
    int j = n2 - 1;
    int k = ans.length - 1;
    int carry = 0;
    while(i >= 0 || j >= 0) {
        int sum  = carry;
        if (i >= 0) {
            sum += arr1[i];
        }
        if (j >= 0) {
            sum += arr2[j];
        }
        carry = sum / 10;
        sum = sum % 10;
        ans[k] = sum;
        i--; j--; k--;
    }
    if (carry > 0) {
        ans[k] = carry;
    }
    for(i = 0; i < ans.length; i++) {
        int val = ans[i];
        if (i == 0 && val > 0) {
            System.out.print(val + " ");
        } else {
            System.out.print(val + " ");
        }
    }
```


arr2
9
7
6
i
4
5
arr1
3
1
0
7
5
j
int b = 0
if(arr2[i] == arr1[j]) {
b=1
}
if(b==1) {
val2 = arr2[i] + 1
}
diff = val2 - arr1[j] - b