Bar chart

Rows = max from array

Col = arr.length

arr[0] = 3

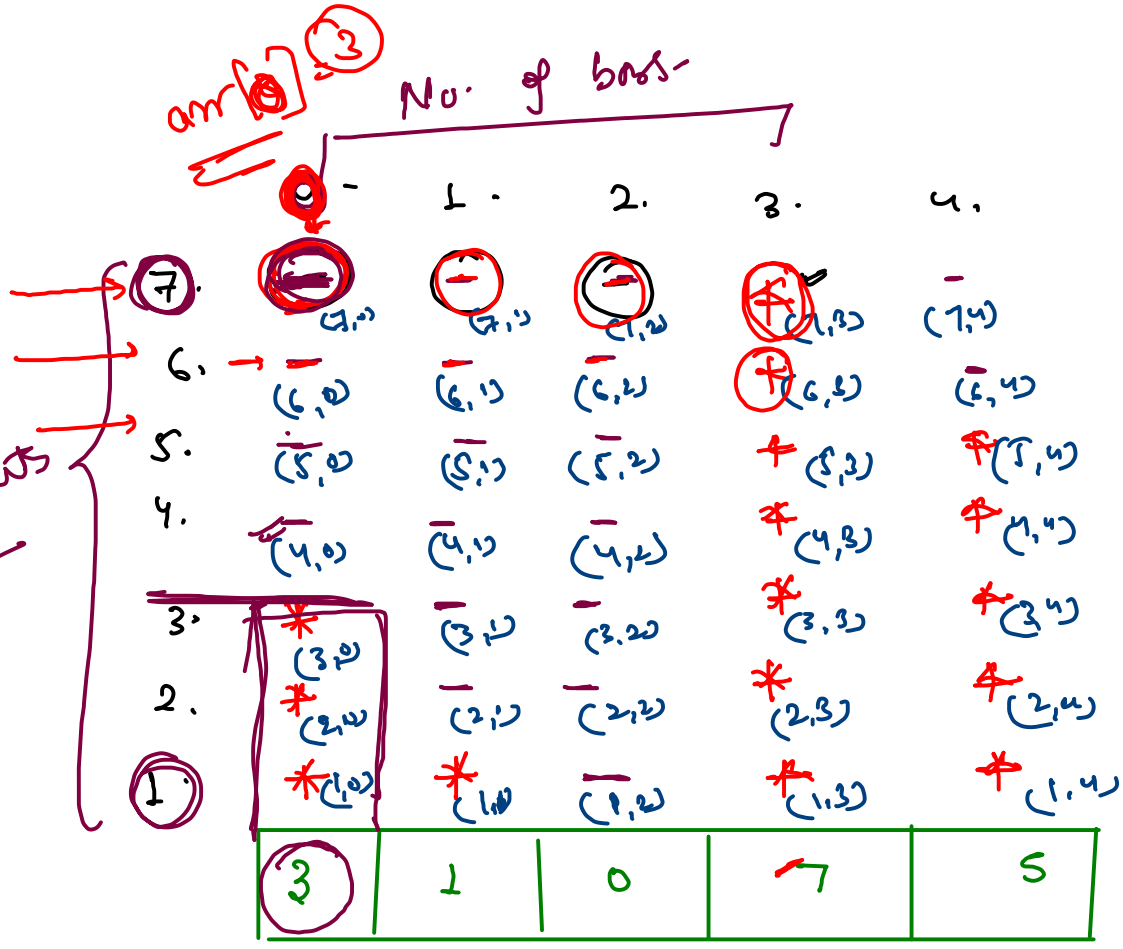
No. of bars

0 1 2 3 4

7 6 5 4 3 2 1

heights

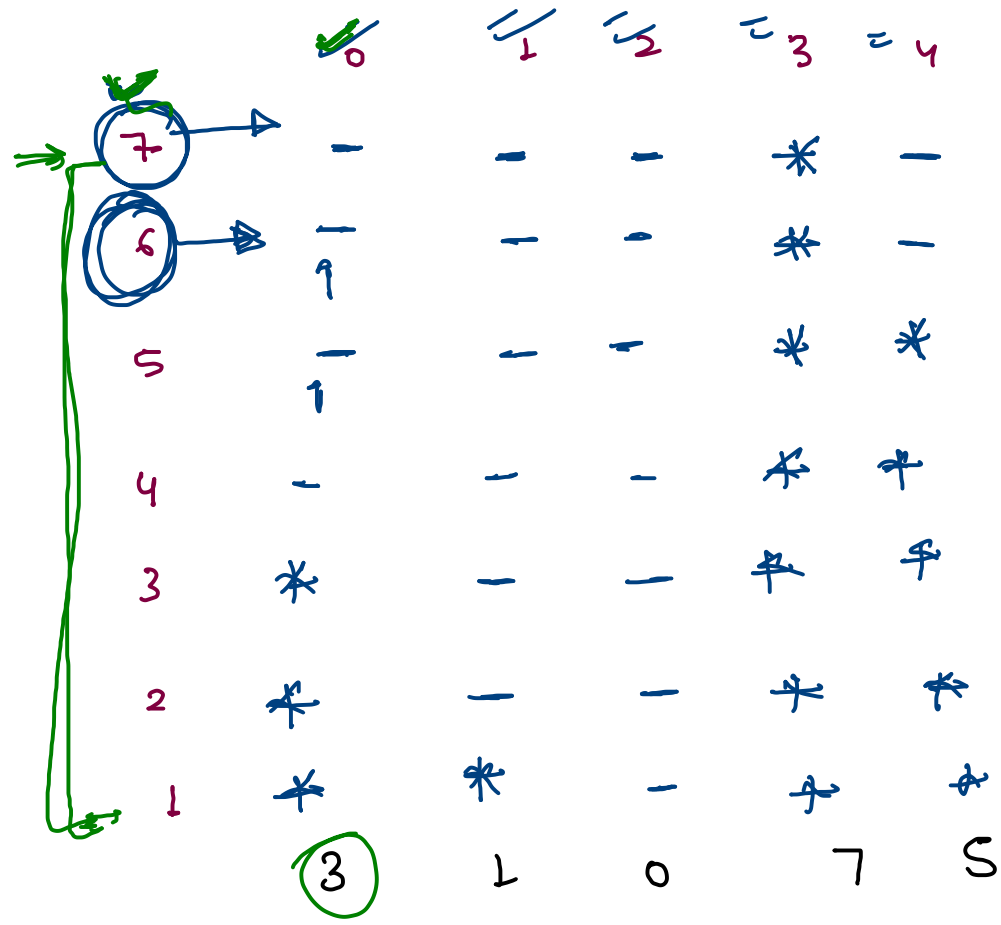
Fit max to h = 1



max = 7

```java
public static void printBarChart(int[] arr) {
    // find max from array
    int mx = max(arr);
    // print barchart
    for(int ht = mx; ht >= 1; ht--) {
        for(int i = 0; i < arr.length; i++) {
            if(ht <= arr[i]) {
                // print star
                System.out.print("*\t");
            } else {
                // print space
                System.out.print("\t");
            }
        }
        // hit enter
        System.out.println();
    }

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



Bonus Question

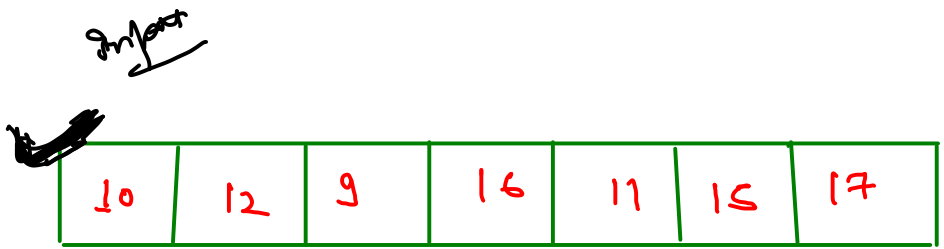
Single Iteration on array.

→ first max

Second max

→ first min

Second min



max = 17

second max = 16

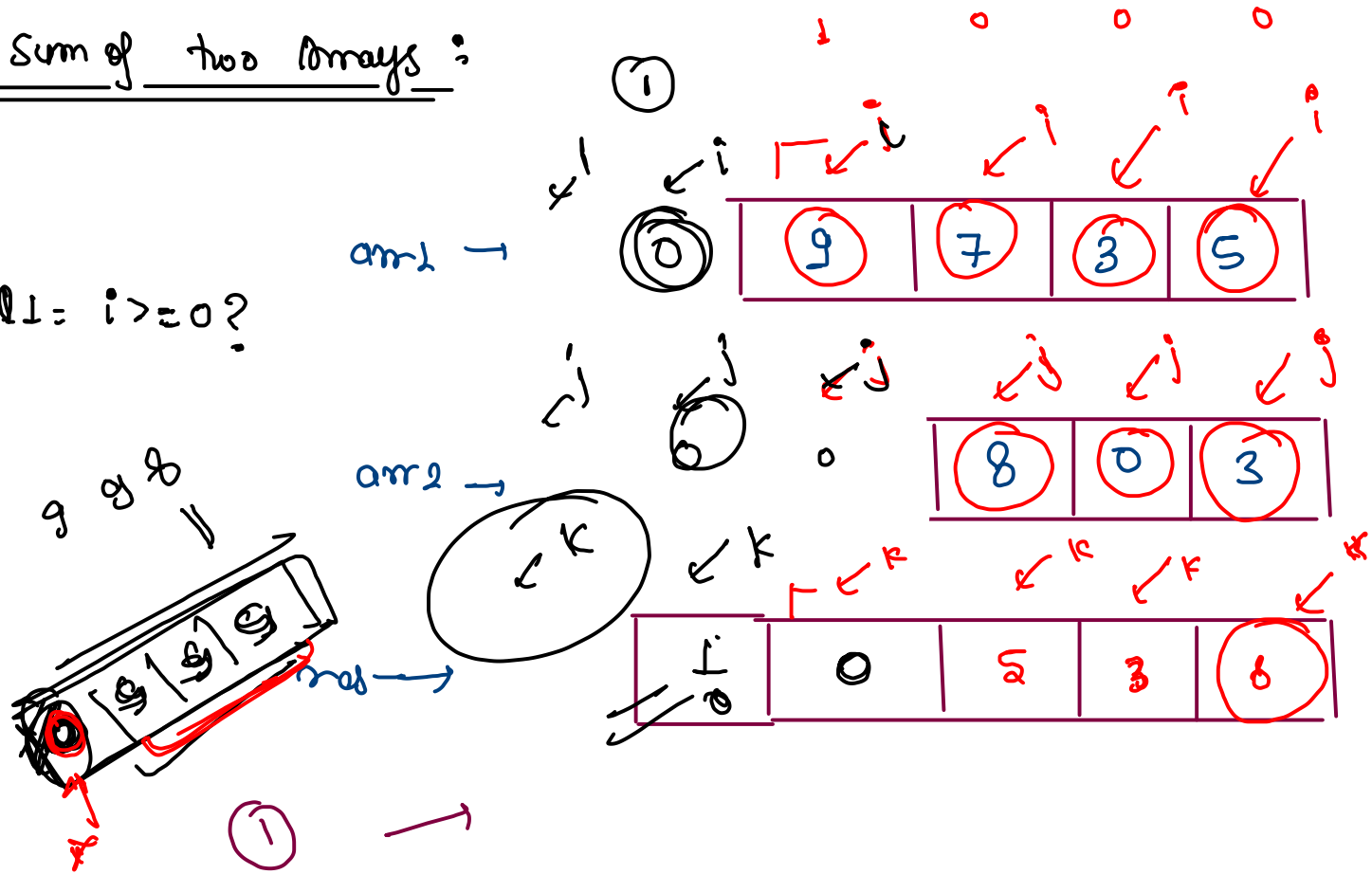
min = 9

second min = 10

Sort ~~}~~

only single iteration

Sum of two Arrays :



val1 = arr1[i] ⓐ

val2 = arr2[j] ⓑ

sum = val1 + val2 + carry

res[k] = sum % 10;

carry = sum / 10;

i--;

j--;

k--;

arr1 → size → s_1

arr2 → size → s_2,

$s_1 > s_2$

res → $s_1 + 1$

$s_2 > s_1$

res → $s_2 + 1$

3 1 0 7 5

1 1 1 1 1

ternary operator

val = condition ? Statement 1 : Statement 2;

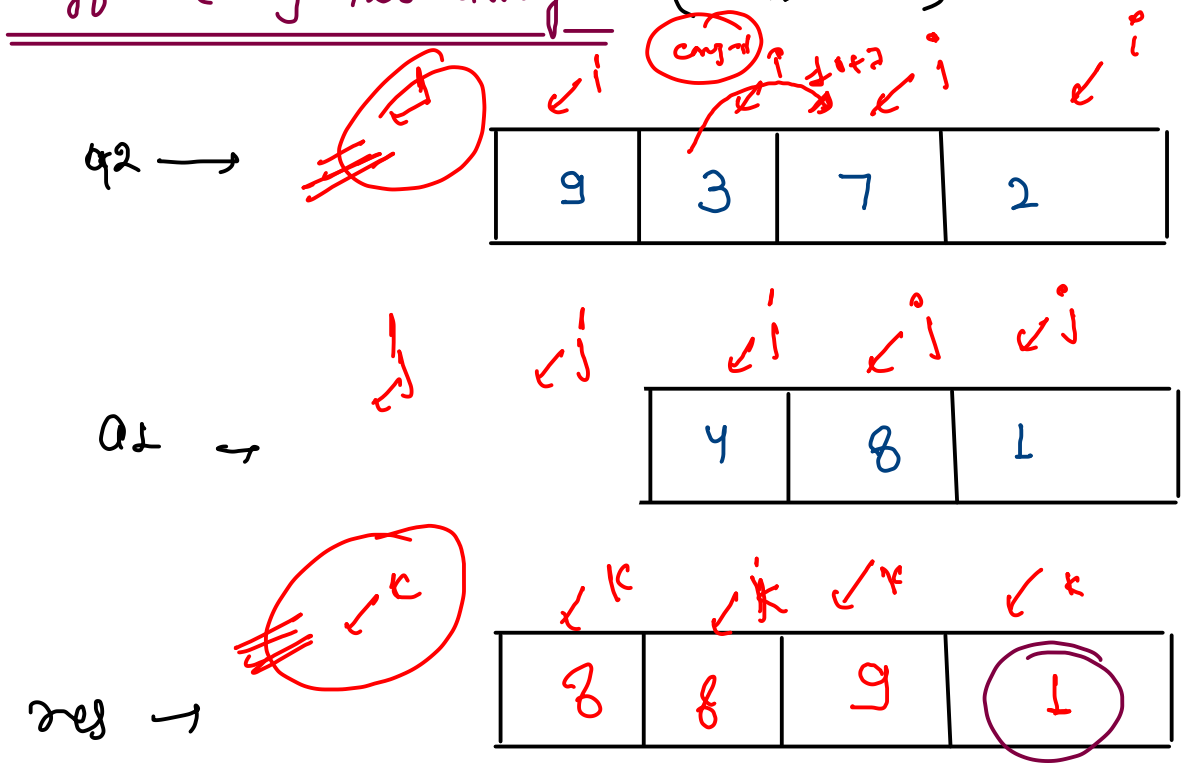
```
if(condition == True){
    Statement 1
} else {
    Statement 2
}
```

i = -1

val = i >= 0 ? arr[i] : 0;

val = 0

difference of two array - (a2 - a1)




```
res[k] = val;
i--;
j--;
k--;
```

```
carry = 0 - 1
```

```
val2 = arr2[i]; = 2 7 9
val1 = arr1[j]; = 1 8 0
```

2 - 4

```
val = val2 + carry - val1
```

Effective value.

```
if (val < 0) {
    val += 10;
    carry = -1;
} else {
    carry = 0 ; //reset carry
}
```

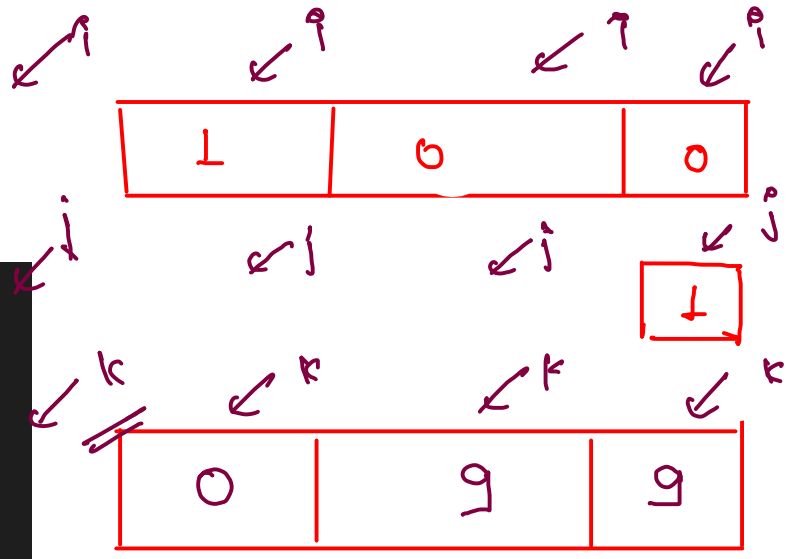
```
for(int k = s2 - 1; k >= 0; k--) {
    int val2 = arr2[i];
    int val1 = arr1[j];

    int val = (val2 + carry) - val1;

    if(val < 0) {
        val += 10;
        carry = -1;
    } else {
        // reset carry
        carry = 0;
    }

    res[k] = val;
    i--;
    j--;
}
```

invalid strings



carry = 0 -1 -1 0

val2 = 1 0 1

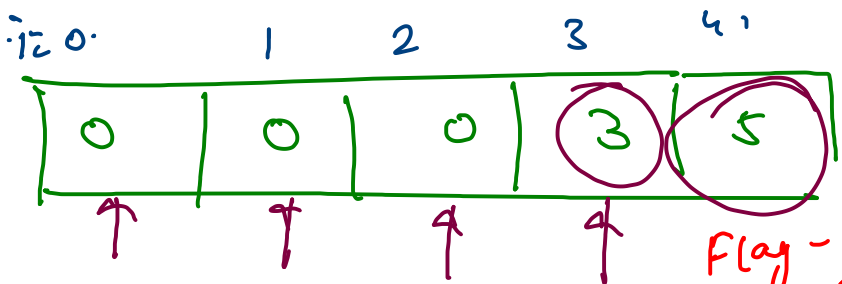
val1 = 1 0 0

val: -1 9 -1 9 0

flag = false; → leading zero available.

True.

```
for(int i=0; i< res.length; i++){
    if(res[i] != 0) {
        Sys.o(res[i]);
```

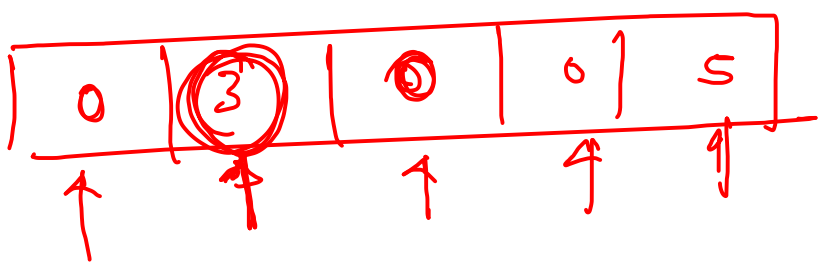


leading zero

Flag - ~~False~~ True

3

}



Zero → leading zero che
True → Sys(arr[i]);
false

Non Zero → Flag = True] so that it can figure out that there is no leading zero

3 0 0 5

Rotate an Array. (arr, k)

Rotate a Number

arr → 10, 20, 30, 40, 50

k = 3

+ve 'k'

k=1 50 10 20 30 40

k=2 40 50 10 20 30

k=3 30 40 50 10 20

-ve 'k'

k=-1 k=-1 → 20 30 40 50 10

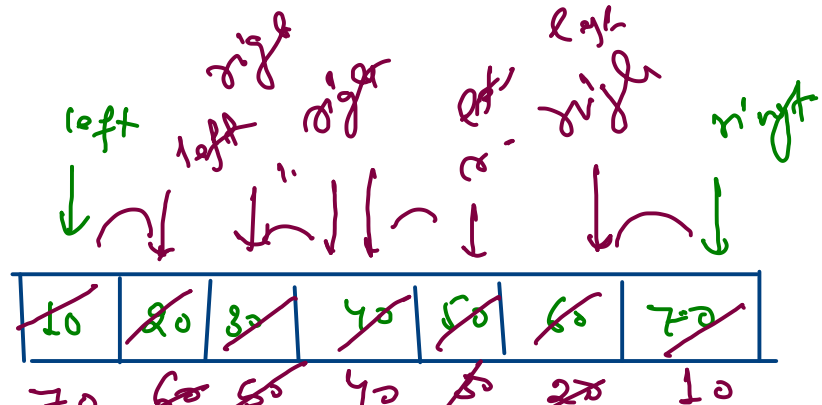
k=-2 → 30 40 50 10 20

k=-3 → 40 50 10 20 30

2 Question ~~Space~~ Not allowed,

time complexity → O(n)

Reverse an Array - →



from left & Right →

```
int left=0;
int righ = arr.length-1;
```

→ Half (left < right)

```
For( datatype of arr  variable name :  name of array)}
        Syso(variable.nam);
}
```

Rotate an array

(k)

k=0

10	20	30	40	50

k=3

k+ve (back to front)

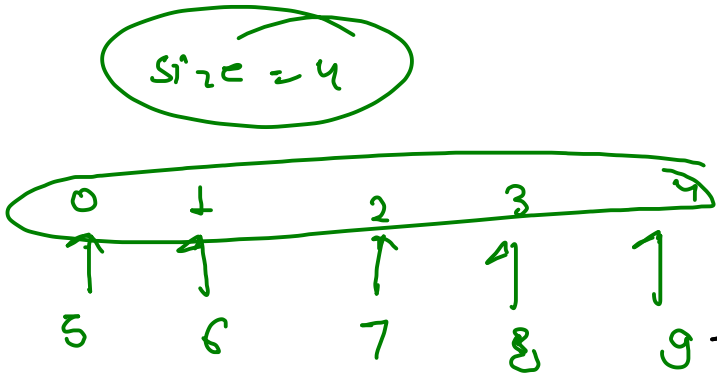
k=0 10

k=1	50	10	20	30	40
k=2	40	50	10	20	30
k=3	30	40	50	10	20
k=4	20	30	40	50	10
10=1 k=5	10	20	30	40	50
k=2 k=6	50	10	20	30	40
k=3 k=7	40	50	10	20	30

if(k = k + arr.length);

k -ve (front to back)

k=4 k=1	20	30	40	50	10
k=3 k=2	30	40	50	10	20
k=3	40	50	10	20	30
k=1 k=4	50	10	20	30	40
k=5	10	20	30	40	50
k=6	20	30	40	50	10
k=7	30	40	50	10	20



10

k = k % arr.length;

if(k < 0) {

k = k + arr.length;

}

Make 'k' in valid Range.

k = k % arr.length;

if(k < 0) {

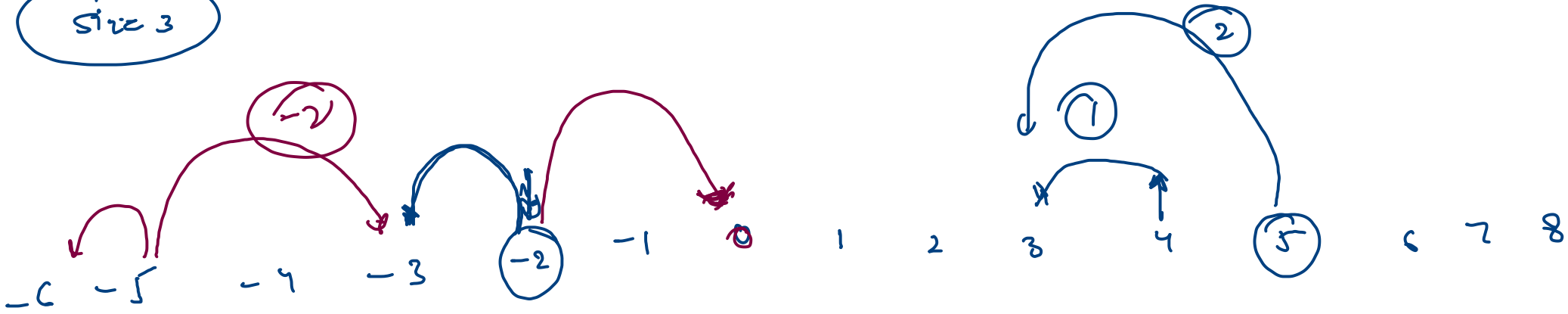
k = k + arr.length;

}

a → (-v)

|a| % b = sign of a (rest)

Size 3



4 % 3 = 1

5 % 3 = 2

5 - 5 % 3

simple 3

k.

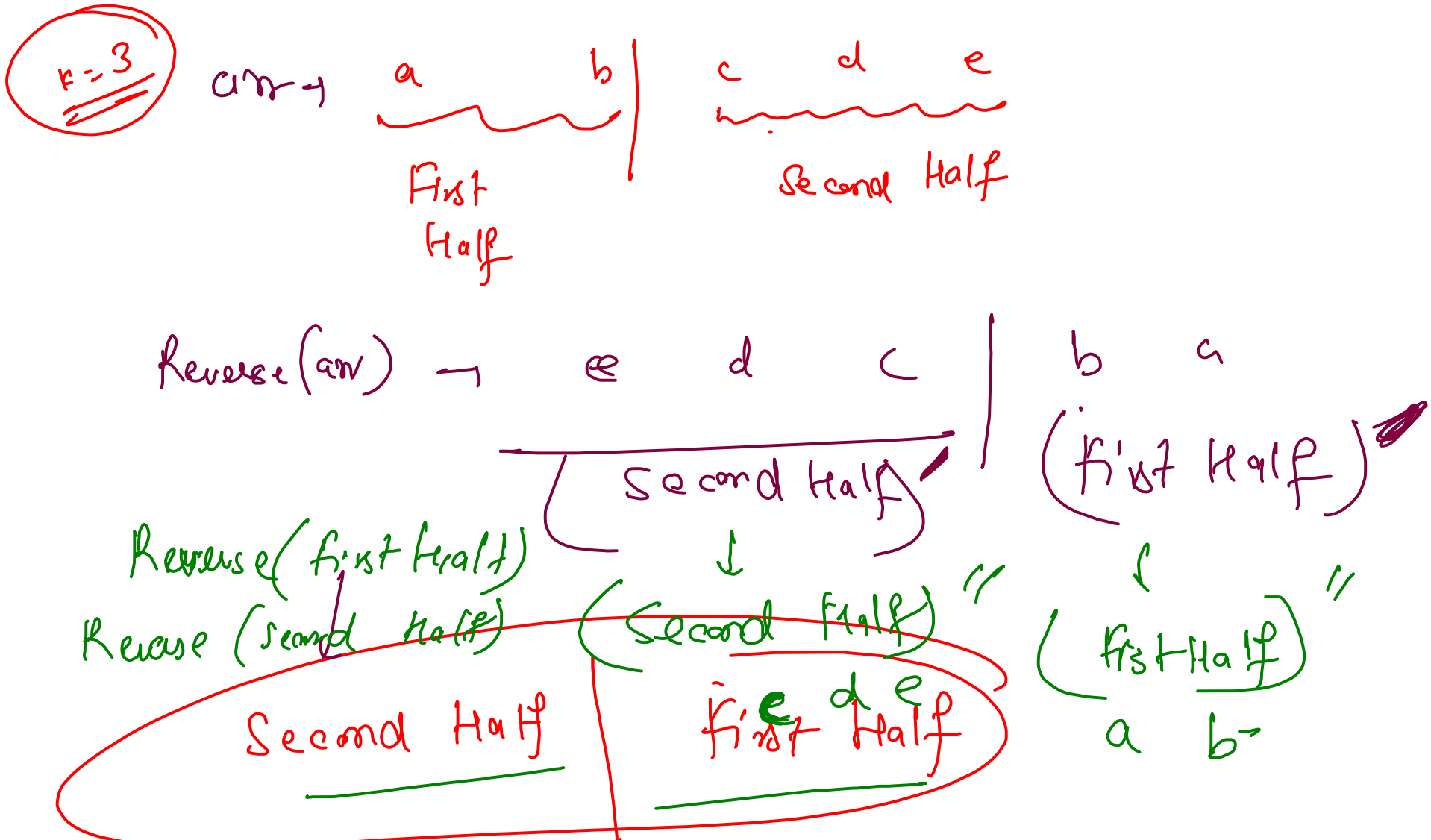
-5

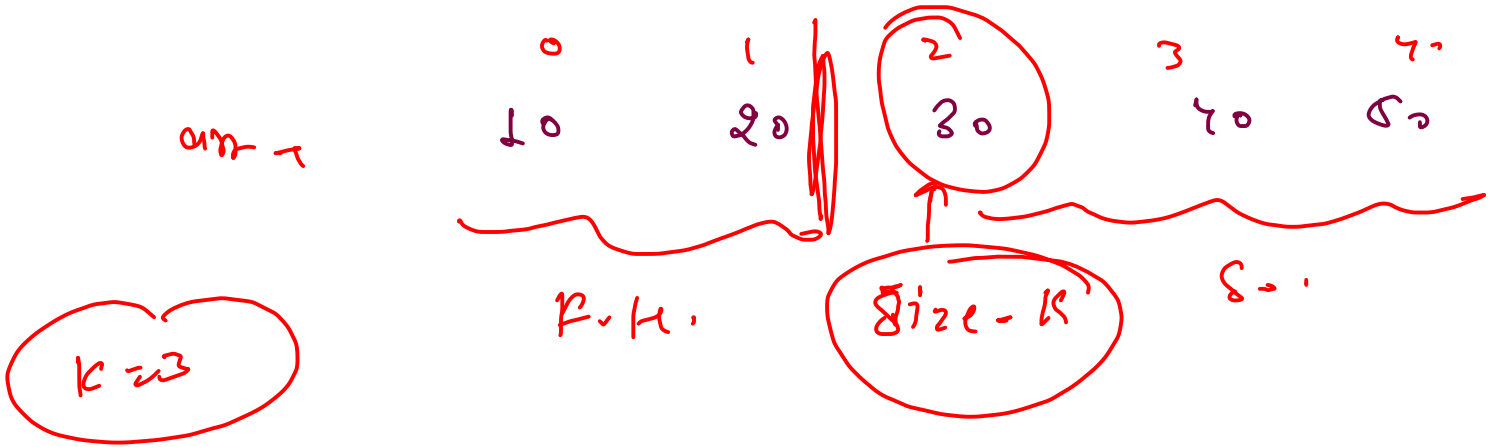
|-4| % 3 = 1

|2| % 3 = -2

-5 % 3 = 1

5 % 3 = -2


F:-3
arr → a b | c d e
First Half
Second Half
Reverse(arr) → e d c | b a
Second Half
First Half
Reverse(first Half)
Reverse (second Half)
Second Half
First Half
Second Half
First Half
a b
e d
a b

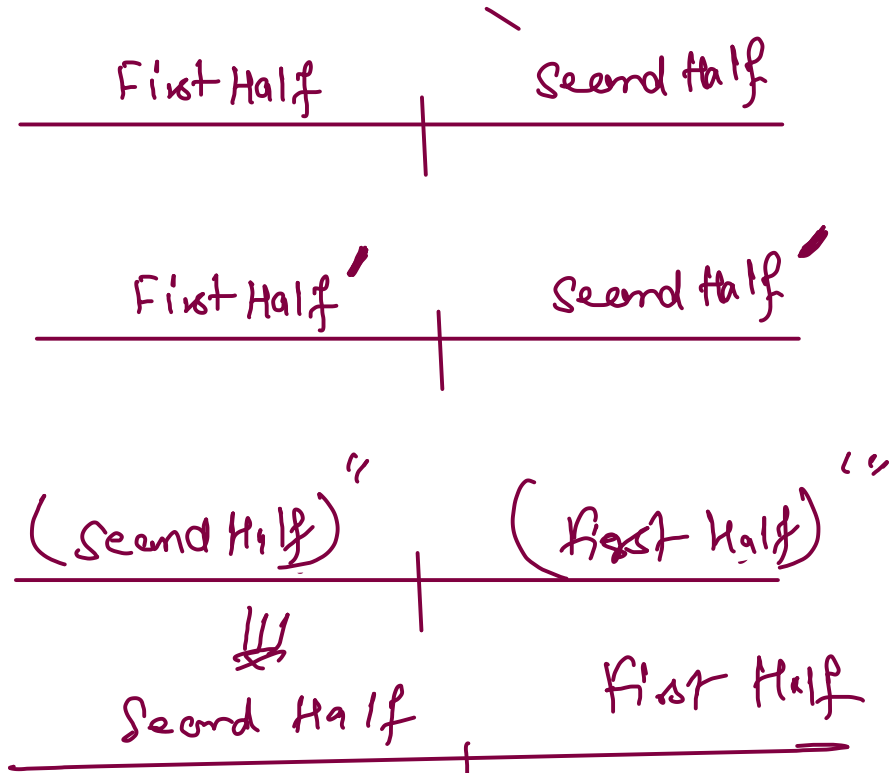


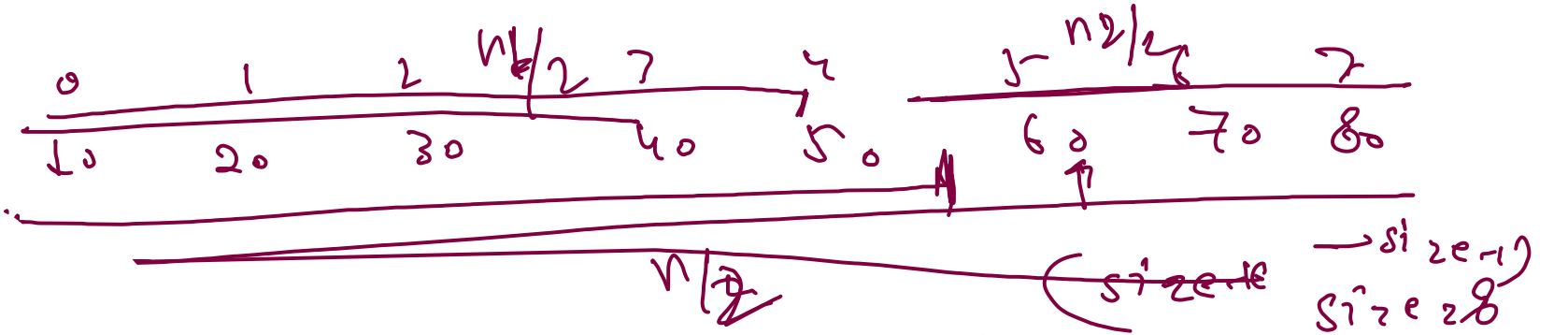
steps:

~~Reverse(arr, 0, size-k-1);~~

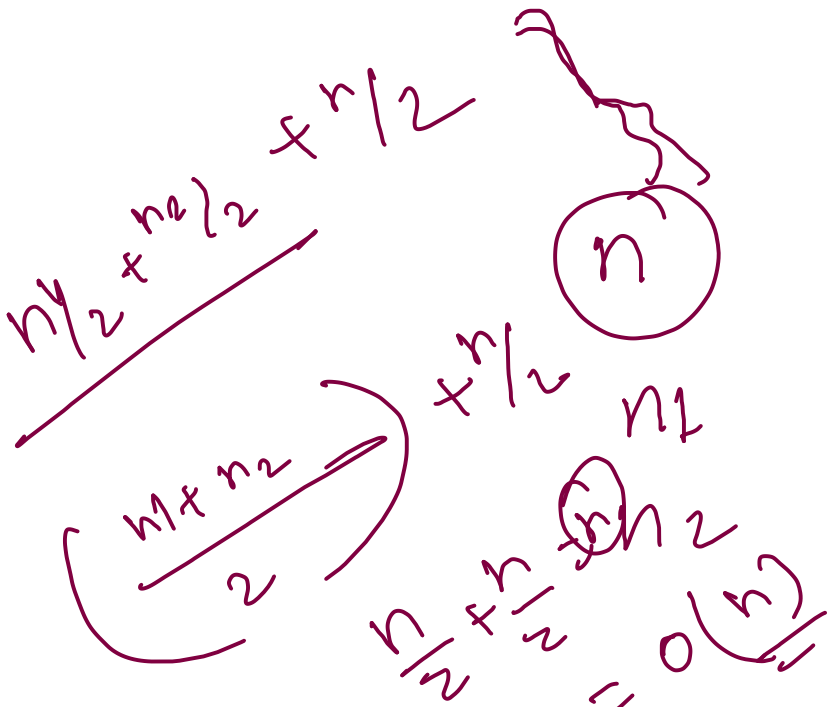
~~Reverse(arr, size-k, size-1);~~

~~Revese(arr, 0, size-1);~~





$\Rightarrow$ size-1c-1 — $1c=3$



$n1+n2 = n$

$n+n = 2n$