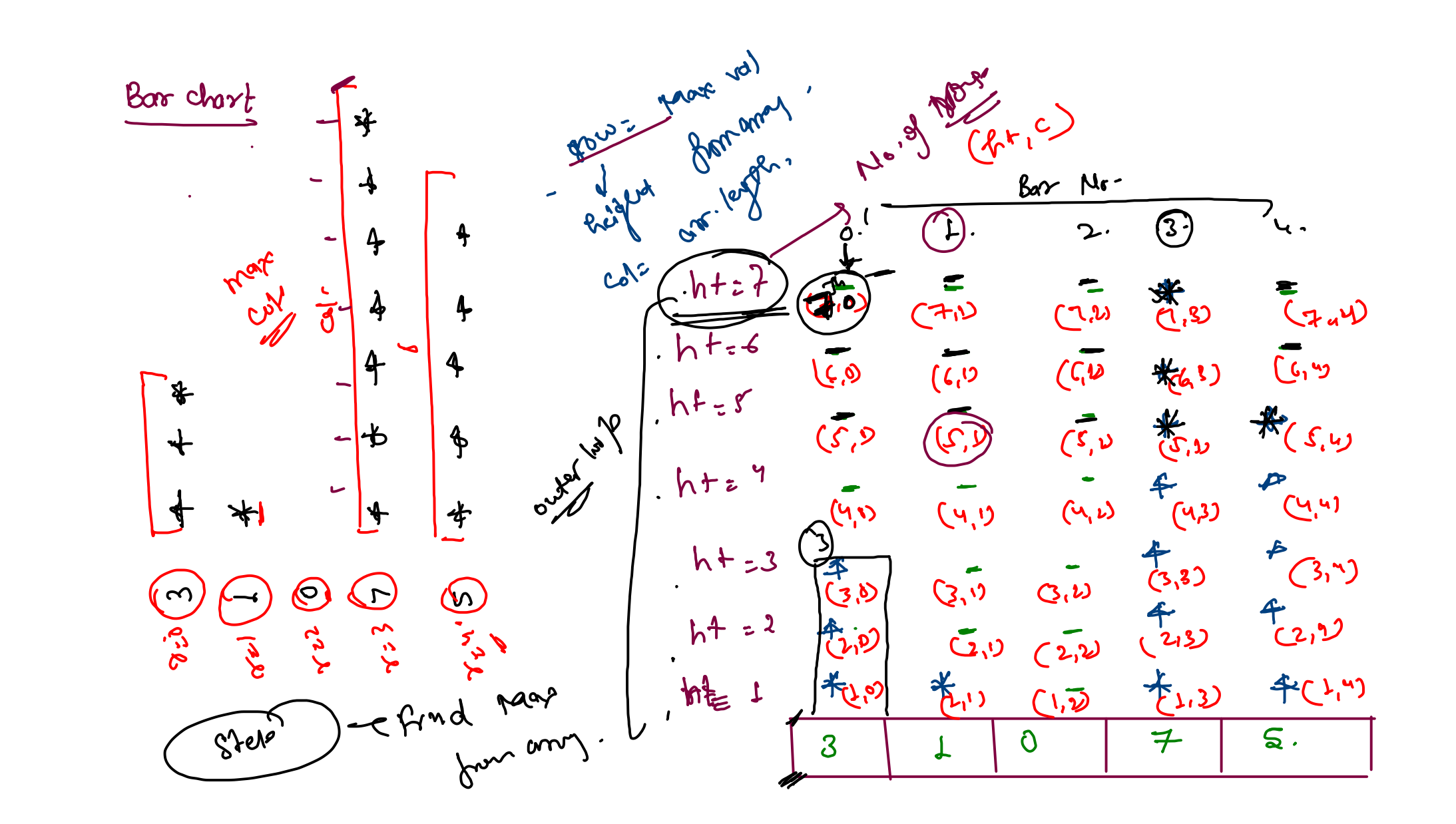

Bar chart
3
1
0
7
5
max col
max ht
Steps
find max from array.
row = max val / height from array, col = arr. length,
No. of bars
(ht, c)
Bar No.
outer loop
ht = 7
ht = 6
ht = 5
ht = 4
ht = 3
ht = 2
ht = 1
3
1
0
7
5

RAM → (i) Stack → Primitive Data type

(ii) Heap → Non primitive data.

Data types → (i) Primitive D.T.

(ii) Non primitive D.T. → Except primite d. Type, all are non primite :=

```java
public static Scanner scn = new Scanner(System.in);

public static void takeInput(int[] arr) {
    for(int i = 0; i < arr.length; i++) {
        arr[i] = scn.nextInt();
    }
}

Run | Debug
public static void main(String[] args) {
    int n = scn.nextInt();
    int[] arr = new int[n];
    takeInput(arr);
}
```

Primitive D.T.

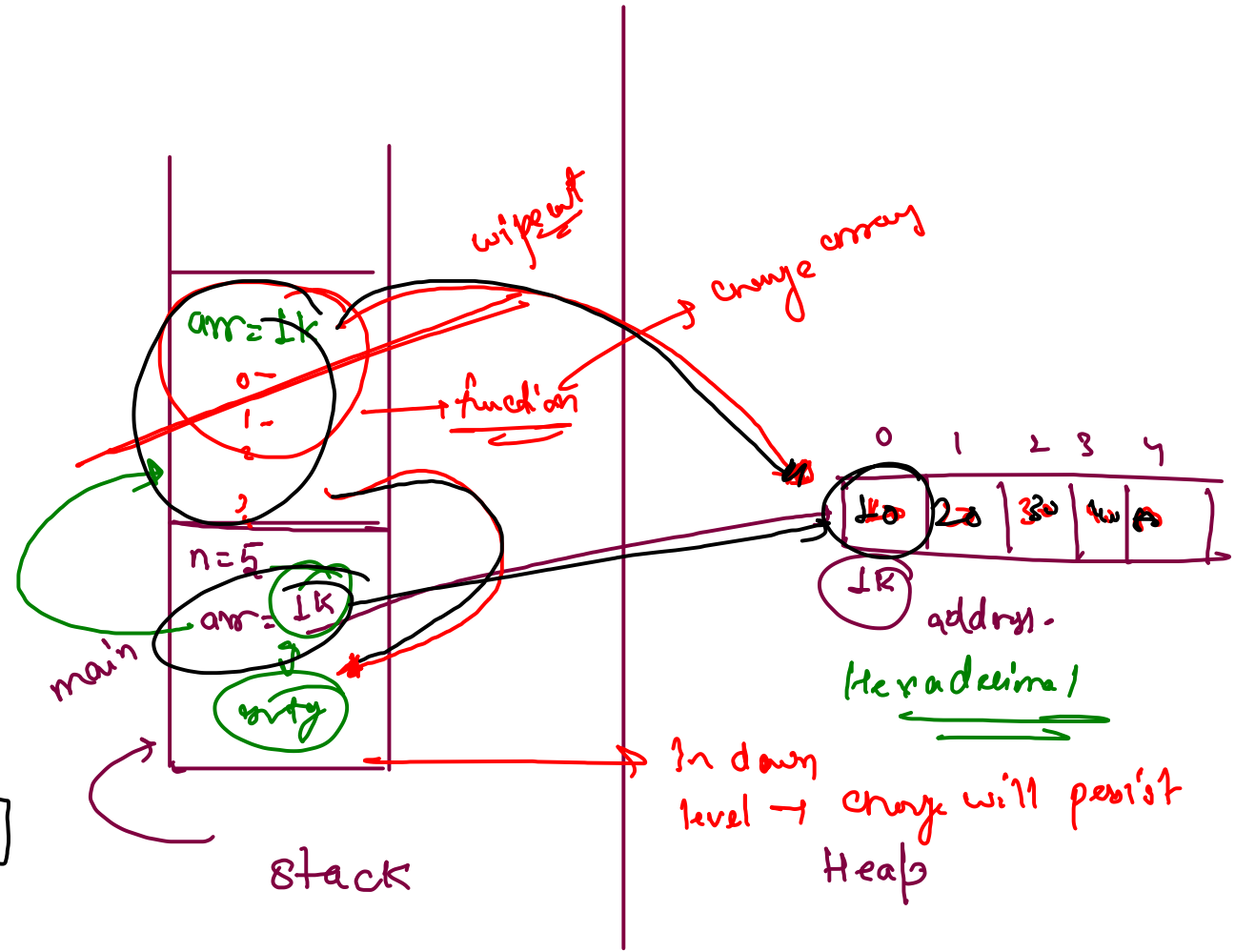
(i) Byte.
(ii) Short
(iii) Int
(iv) long
(v) float
(vi) Double
(vii) char
(viii) booleam

```
int[] arr;

arr = new int[5];
```

10
20
30
40
50

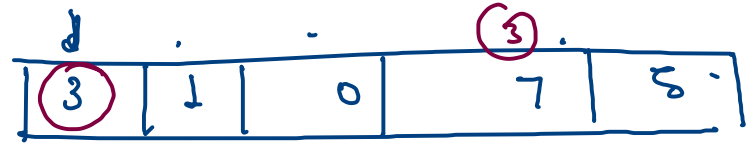
```java
public static Scanner scn = new Scanner(System.in);

public static void takeInput(int[] arr) {
    for(int i = 0; i < arr.length; i++) {
        arr[i] = scn.nextInt();
    }
}

Run | Debug
public static void main(String[] args) {
    int n = scn.nextInt();
    int[] arr = new int[n];
    takeInput(arr);
}
```

arr → [10 | 20 | 30 | 40 | 50]

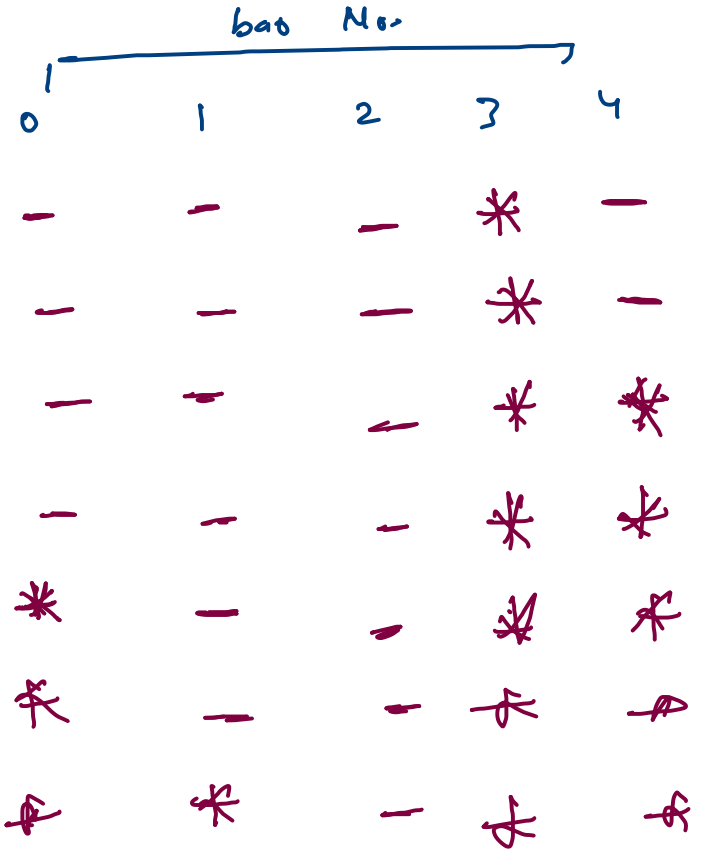


Integer.MIN_VALUE } $-\infty$

arr →



Row = Max from array. ⑦

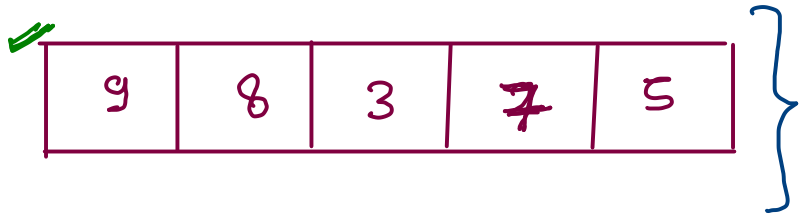



```java
public static void printBarChart(int[] arr) {
    // find max from array
    int max = max(arr);

    for(int ht = max; ht >= 1; ht--) {
        for(int i = 0; i < arr.length; i++) {
            if(arr[i] >= ht) {
                // print star
                System.out.print("*\t");
            } else {
                // print space
                System.out.print("\t");
            }
        }
        // hit enter
        System.out.println();
    }

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

Sum of two arrays.

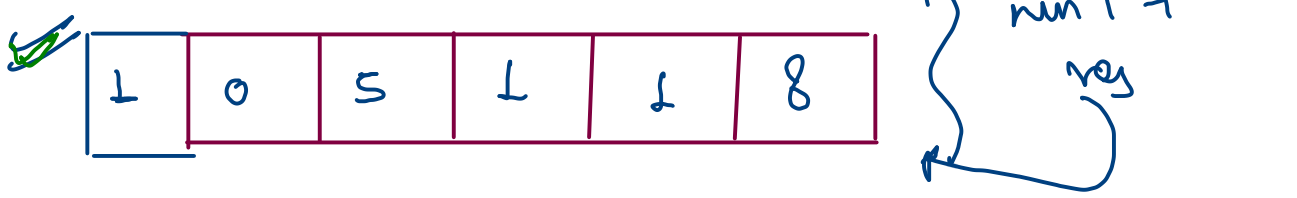
$+10^9$
$=$
-10^9

s1 → arr1 →



s2 → arr2 →

res =



s1 ≥ s2

res → s1.f1

s2 > s1

res → s2.f1

res = 0
i = 0; i < arr.length
for res += arr[i]

if (a * e = a.) then e is Identity for operator '*'

↑ operator.

power = 1; 1o 10

↑ identity.

pow * = 10;

$a + e = a$

$e = 0$ 0 is identity + operator.

$a - e = a$

$e = 0$, 0 is identity - operation.

$a \times e = a$

$e = 1$, 1 is identity for 'x' operator.

$a / e = a$

$e = 1$ 1 is identity for division operator.

$\min(a, e) = a.$

int

$e = ?$

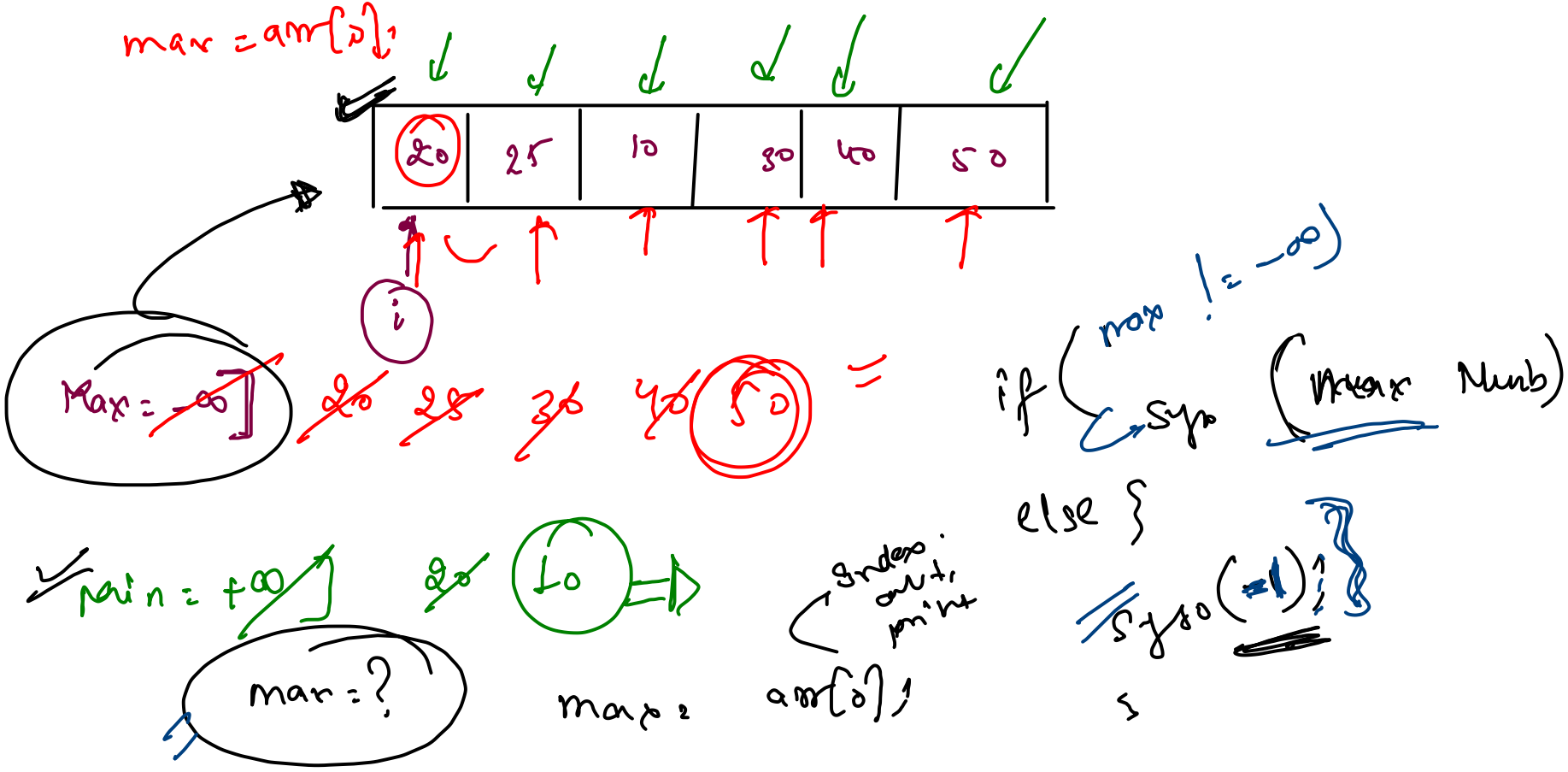
$e = +\infty$ e = Integer.MAX_VALUE

$e = +\infty$

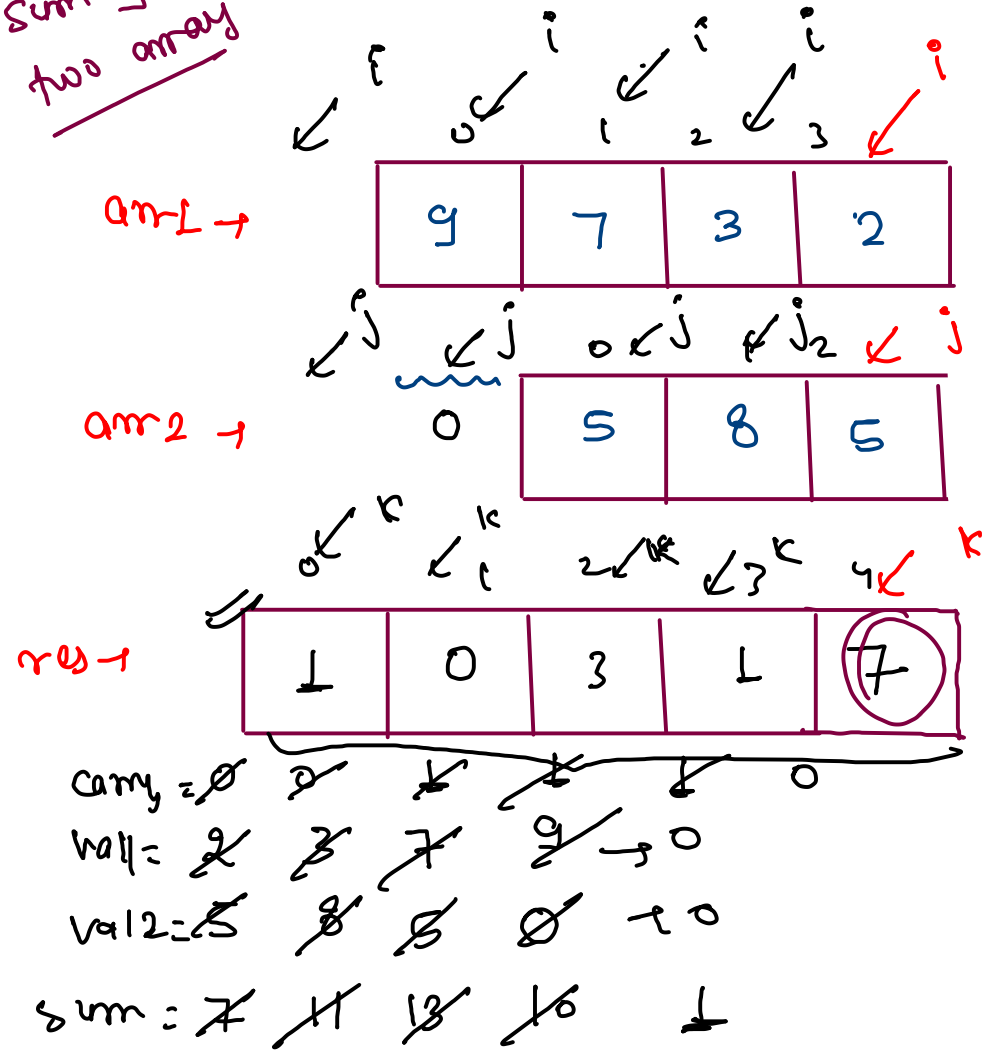
$\max(a, e) = a$

$e = ?$

$e = -\infty$, e = Integer.MIN_VALUE


max = arr[0];
20
25
10
30
40
50
i
Max = -∞
20
25
30
40
50
=
if (max != -∞)
Sys
Max Numb
else {
Sys0(-1);
}
min = +∞
20
10
max = ?
Order out print
max = arr[0];
[ ]
max = -∞;
return max;

Sum of two array:




```
int carry = 0
int val1 = arr1[i]
int val2 = arr2[j]
int sum = val1 + val2 + carry

res[k] = sum % 10;
carry = sum / 10;

i--;
j--;
k--;
```

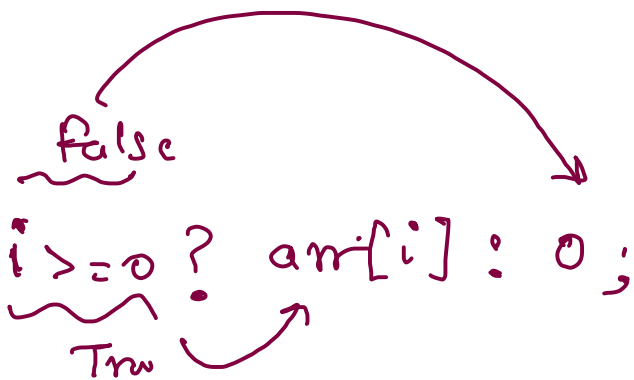
i = 3 2 1 0
j = 2 1 0 -1
k = 4 3 2 1

ternary operator →

```
int val = condition ? Statement ① : Statement ②
```

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

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

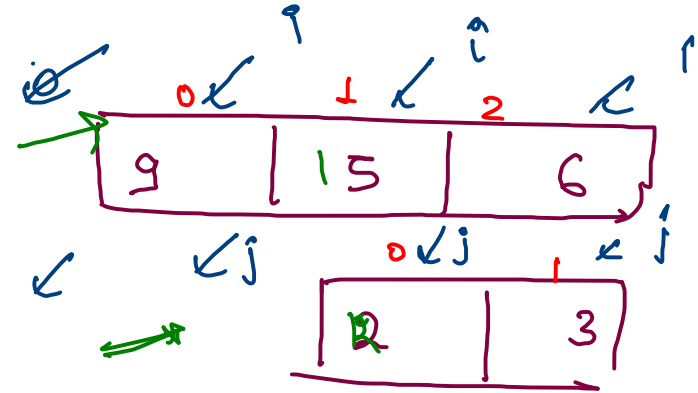


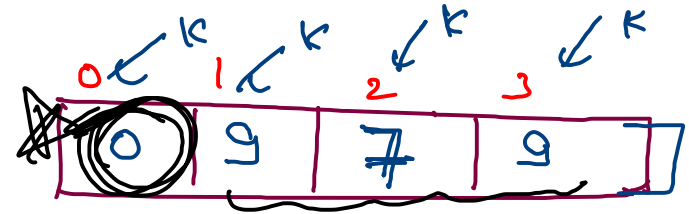
```java
public static void printSumOfArray(int[] arr1, int[] arr2) {
    int s1 = arr1.length;
    int s2 = arr2.length;

    int rsize = s1 > s2 ? s1 + 1 : s2 + 1;
    int[] res = new int[rsize];

    int i = s1 - 1;
    int j = s2 - 1;
    int carry = 0;
    for(int k = rsize - 1; k >= 0; k--) {
        int val1 = i >= 0 ? arr1[i] : 0;
        int val2 = j >= 0 ? arr2[j] : 0;

        int sum = val1 + val2 + carry;
        res[k] = sum % 10;
        carry = sum / 10;
        i--;
        j--;
    }

    // print array
```

s1 = 3

s2 = 2

rsize = 4





carry = 0 0 0 0

val1 = 6 5 9 0

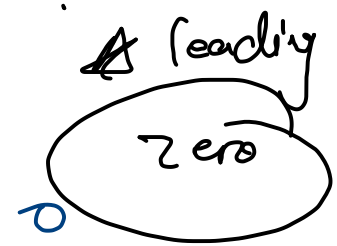
val2 = 3 2 0 0

sum = 9 7 9 0

i = 2 1 0

j = 1 0 -1

k = 3 2 1

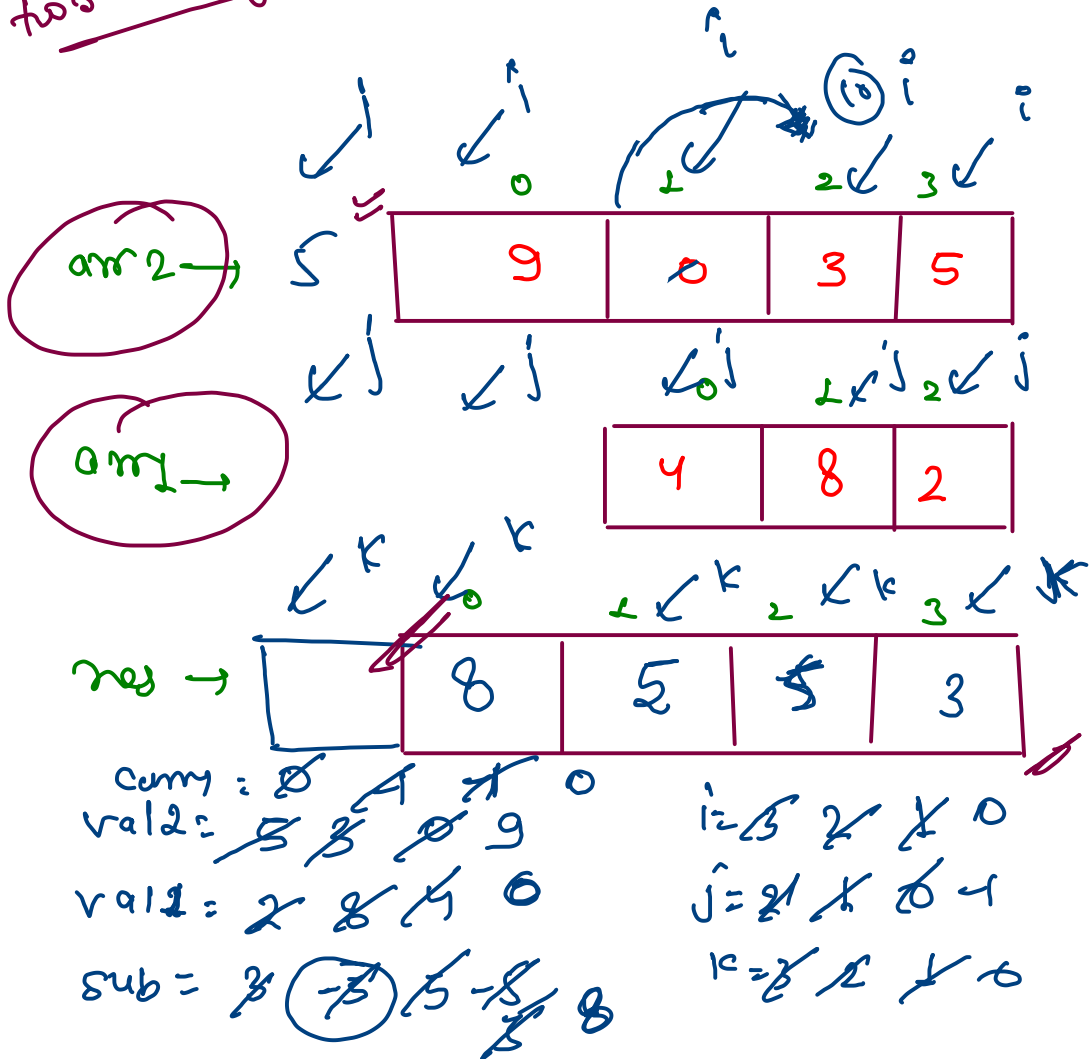


9

7

9

diff of two arrays:

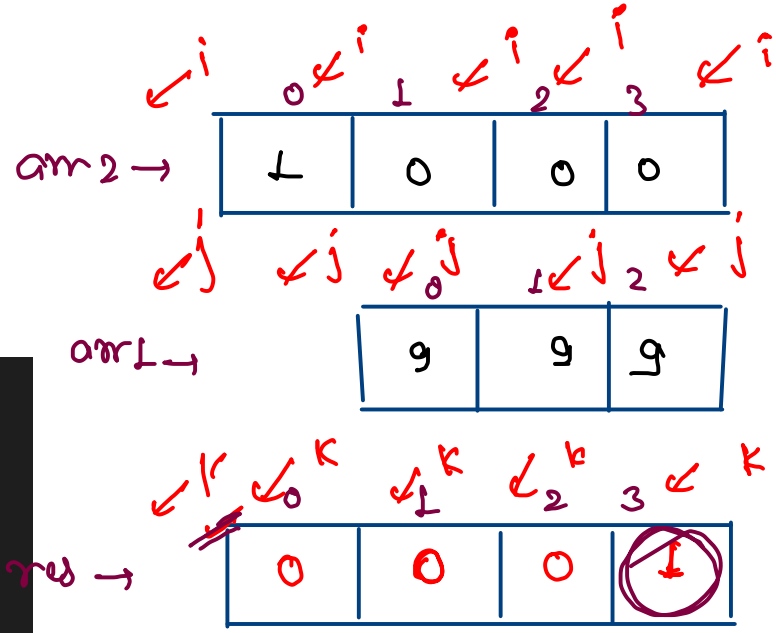


borrow

```
carry = 0
val2 = arr2[i]
val1 = arr1[j]
sub = (val2 + carry) - val1
if(sub < 0) {
    sub += 10;
    carry = -1;
} else {
    carry = 0;
}
res[k] = sub;
i--;
j--;
k--;
```

Reset is important

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

    int sub = (val2 + carry) - val1;

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

    res[k] = sub;

    i--;
    j--;
}

// print res
```



carry = 0 0 0 1

val2 = 0 0 0 1

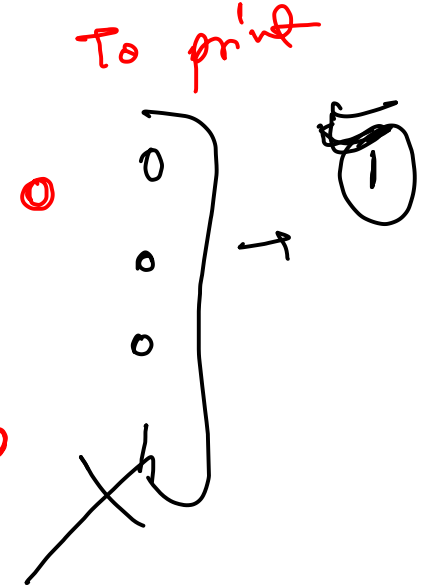
val1 = 9 9 9 0

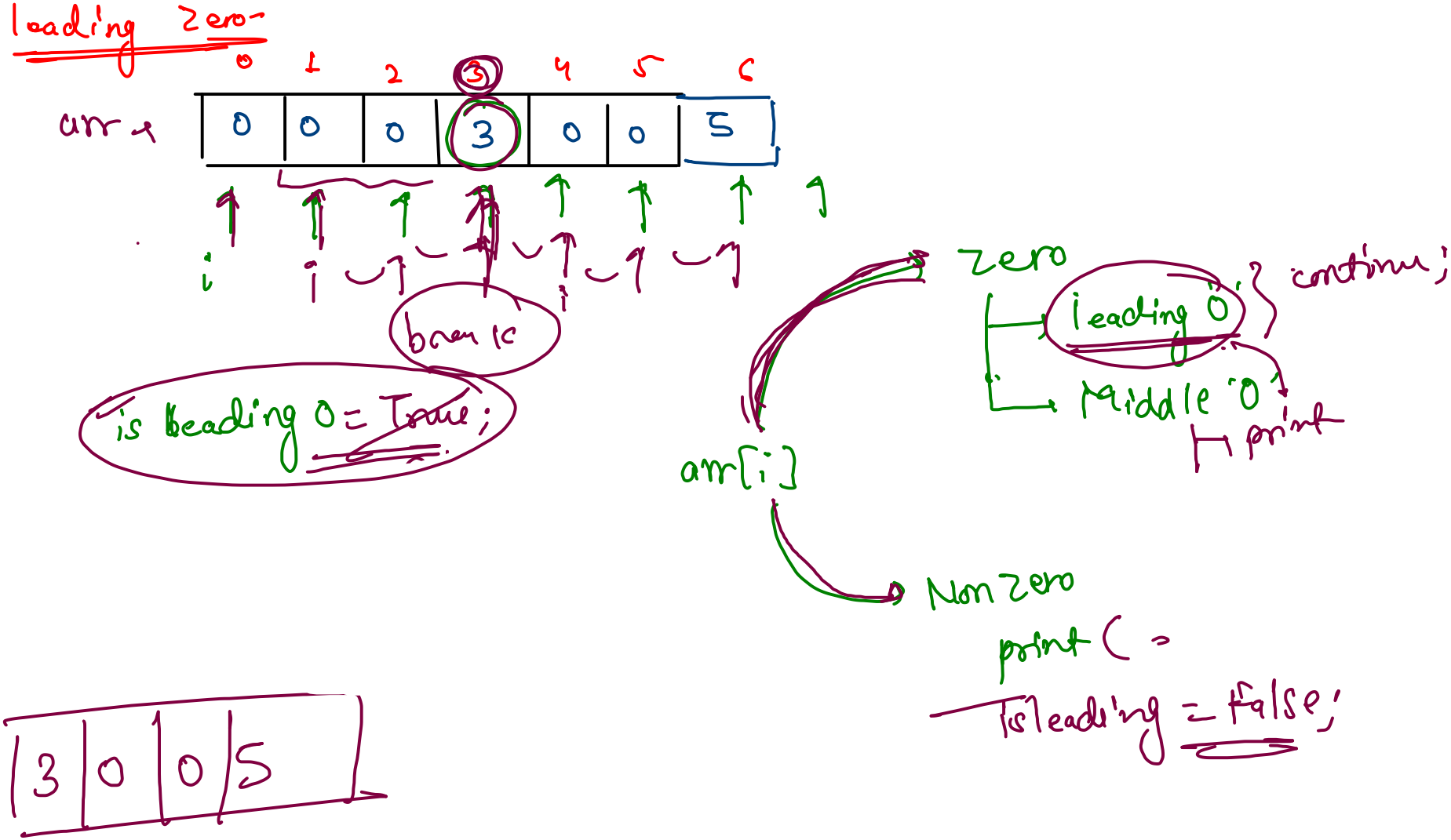
sub = -9 1 -10 0 -10 0 0

i = 3 2 1 0 -1

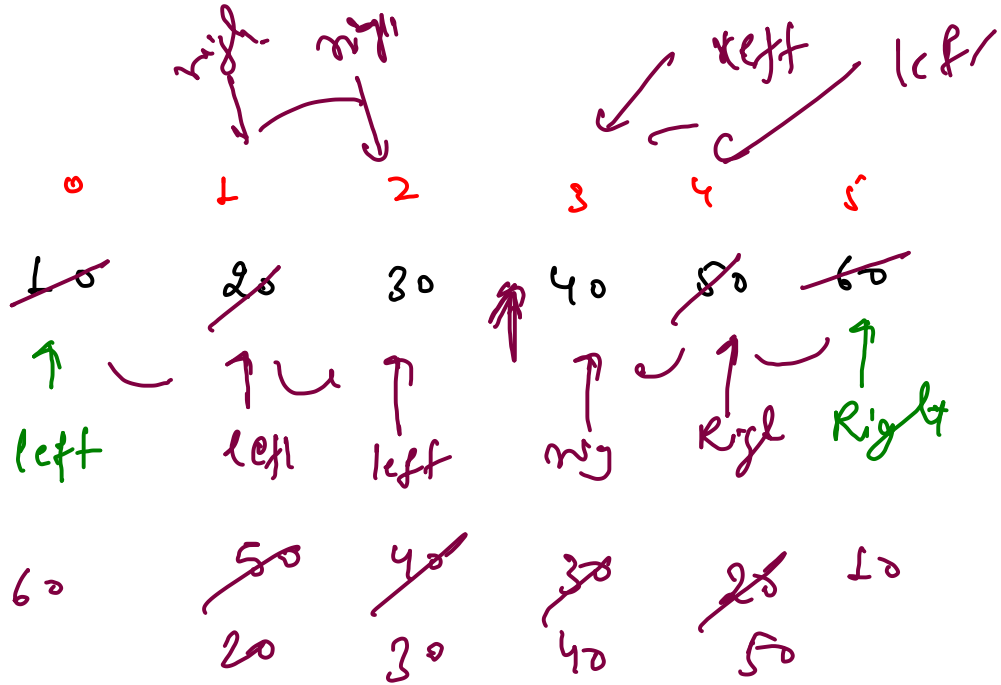
j = 2 1 0 -1 -2

k = 3 2 1 0 -1




leading zero
0 1 2 3 4 5 6
arr →
0 0 0 3 0 0 5
i
break
is leading 0 = true;
zero
leading 0
continue;
Middle '0'
print
arr[i]
Non zero
print (
isleading = false;
3 0 0 5

Reverse an array.




```
while ( left < right)

Swap;

left++;
right--;
```