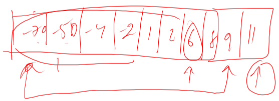


-70 X -50

2500

(last 2 number)

(starting 2 number)

[4,6] = (6-4+1)

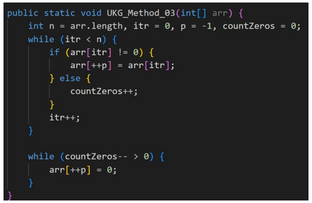
lG = 0
rG =

9 × 6 = 54

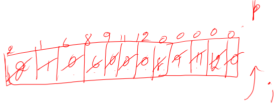
num1 = arr[1] * arr[0]
num2 = arr[0] * arr[1]

11, 15

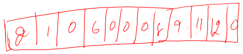
15-11+1 = 5



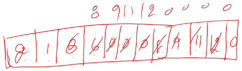
```
public static void UKG_Method_03(int[] arr) {
    int n = arr.length, itr = 0, p = -1, countZeros = 0;
    while (itr < n) {
        if (arr[itr] != 0) {
            arr[++p] = arr[itr];
        } else {
            countZeros++;
        }
        itr++;
    }

    while (countZeros-- > 0) {
        arr[++p] = 0;
    }
}
```



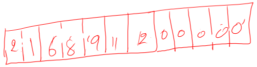
2
1
6
8
9
11
12
0
0
0
0

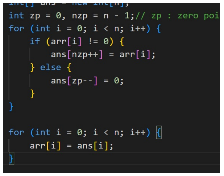


real copy :



temp → 1

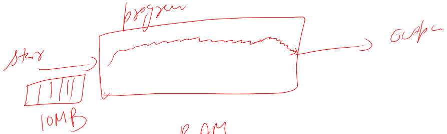


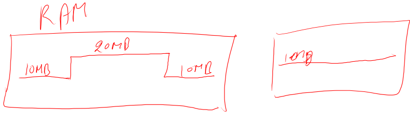


```
int[] ans = new int[n];
int zp = 0, nzp = n - 1;// zp : zero poi
for (int i = 0; i < n; i++) {
    if (arr[i] != 0) {
        ans[nzp++] = arr[i];
    } else {
        ans[zp--] = 0;
    }
}

for (int i = 0; i < n; i++) {
    arr[i] = ans[i];
}
```

zp

nzp





1	2	10	15	18	20

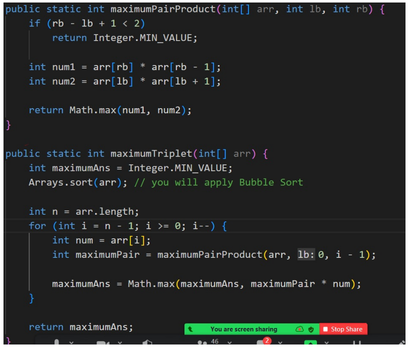
$1 + (a+b) = 35$

$\boxed{a+b = 34}$

$a + b + c = \text{target}$

$\boxed{a+b = \text{target} - c}$

$\boxed{a+b = (t-x)'}$



```java
public static int maximumPairProduct(int[] arr, int lb, int rb) {
    if (rb - lb + 1 < 2)
        return Integer.MIN_VALUE;

    int num1 = arr[rb] * arr[rb - 1];
    int num2 = arr[lb] * arr[lb + 1];

    return Math.max(num1, num2);
}

public static int maximumTriplet(int[] arr) {
    int maximumAns = Integer.MIN_VALUE;
    Arrays.sort(arr); // you will apply Bubble Sort

    int n = arr.length;
    for (int i = n - 1; i >= 0; i--) {
        int num = arr[i];
        int maximumPair = maximumPairProduct(arr, lb: 0, i - 1);

        maximumAns = Math.max(maximumAns, maximumPair * num);
    }

    return maximumAns;
}
```

0	1	2	3	4	5	6	7
-70	-30	-10	1	2	10	15	29

[a,b] = 6-4+1

(a,b) = 6-4-1

[a,b)
(a,b] } → 6-4

num = 15

ans = 8,70,000 (-70 × -30 × 29)

tar = 30

4	6	8	9	7	10	12	13	16	17	22
	↑	↑								↑

4 + 9 + 17

4 + 10 + 16

6 +

i = 0	N
i = 1	N
i = 2	N
i = 3	N

1 → 4N

10 → 4N

100 → 4N

1000 → 4N

1 cr → 4N

1 → X

X → $\underline{\underline{X}}$

1 → N

2 → 2N

3 → 3N

4 → 4N

10 → [illegible]