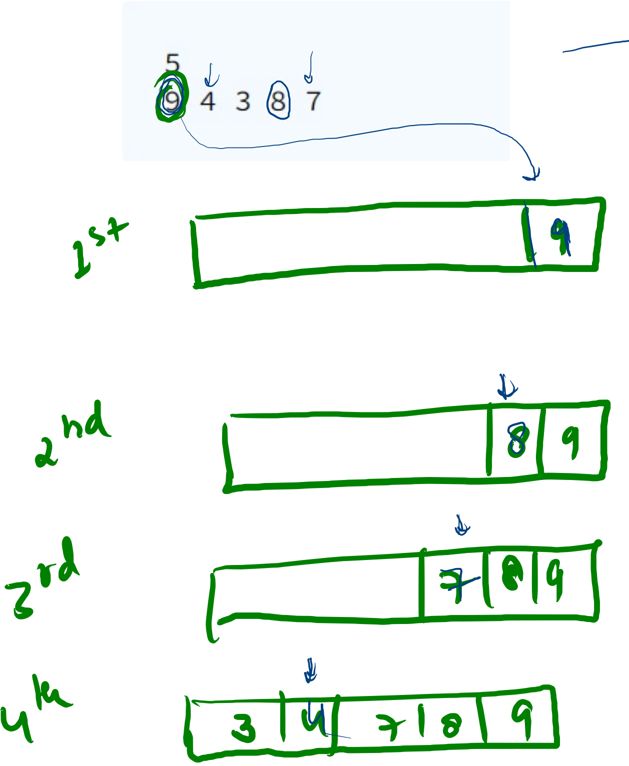
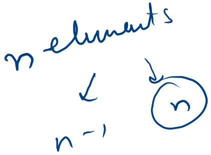
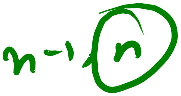
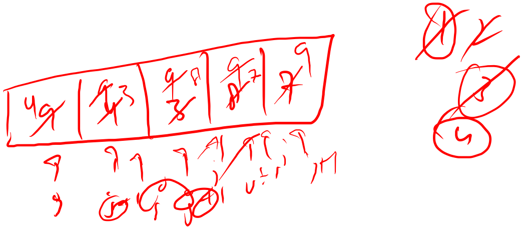
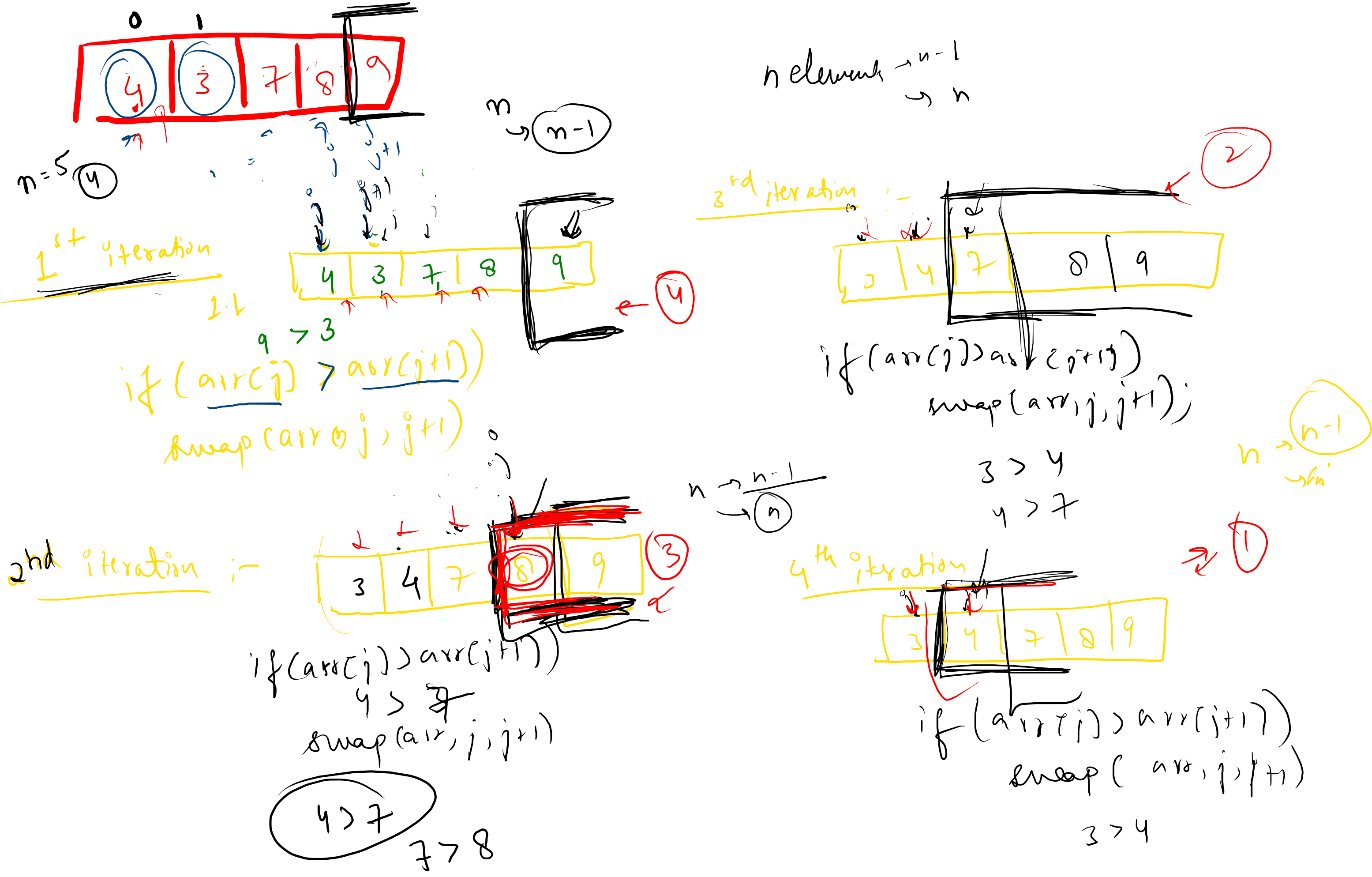
Bubble sort → Largest element gets its correct position first











if (arr(j) > arr(j+1))
swap (arr, j, j+1)

if (arr(j)>arr(j+1))
swap(arr, j, j+1);

3 > 4
4 > 7

if (arr(j)) arr(j+1))
4 > 7
swap (arr, j, j+1)

4>7
7 > 8

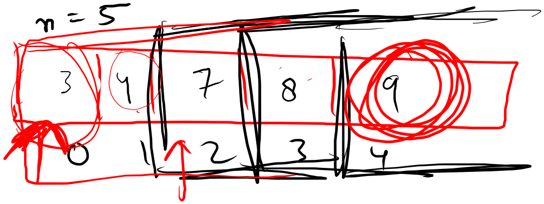
if (arr(j)) arr(j+1))
swap (arr, j, j+1)

3 > 4

```java
public static void swap(int arr[],int j,int i){
    int temp=arr[j];
    arr[j]=arr[i];
    arr[i]=temp;
}

public static void bubbleSort(int arr[]){
    int n=arr.length;

    for(int itr=1;itr<=n-1;itr++){
        for(int j=0;j<n-itr;j++){
            if(arr[j]>arr[j+1]){
                swap(arr,j,j+1);
            }
        }
    }
}
```



itr = 1 <= 4 (T)

j = ~~0~~ ~~1~~ ~~2~~ ~~3~~ 4 < 4

arr[j] > arr[j+1]

9 > 3

9 > 7

9 > 8

itr = 2 <= 4

j = ~~0~~ ~~1~~ ~~2~~ 3 < 3

4 > 3

4 > 7

7 > 8

itr = 3 <= 4

j = ~~0~~ ~~1~~ 2 < 2

3 > 4

4 > 7

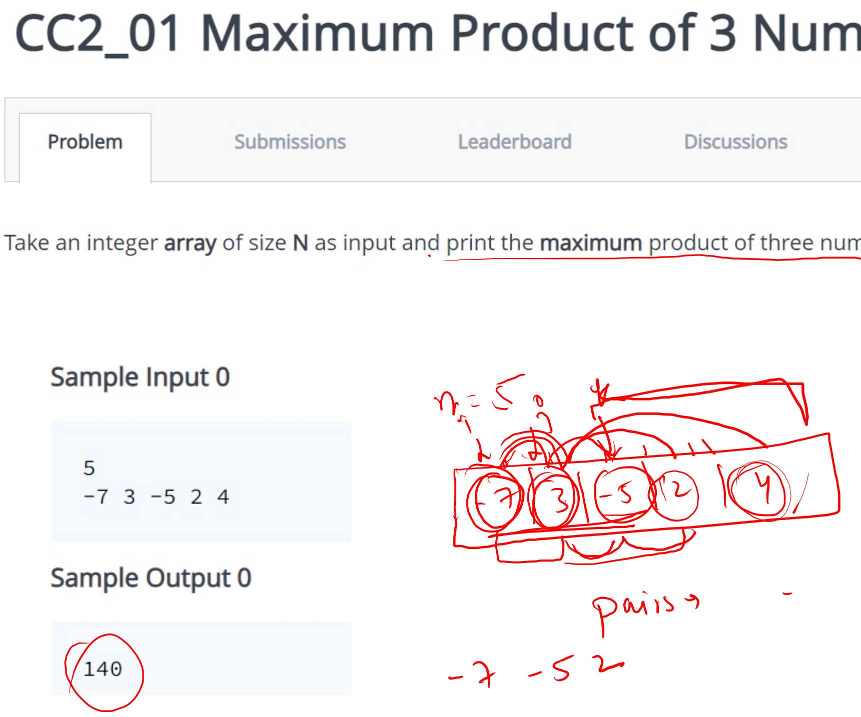
itr = 4 <= 4

j = ~~0~~ 1 < 1

3 > 4

~~itr = 5 <= 4~~

2 3



CC2_01 Maximum Product of 3 Numbers

Problem Submissions Leaderboard Discussions

print triplets → 2 3 5

Take an integer **array** of size **N** as input and print the **maximum** product of three numbers.

a

Sample Input 0

```
5
-7 3 -5 2 4
```

Sample Output 0

```
140
```

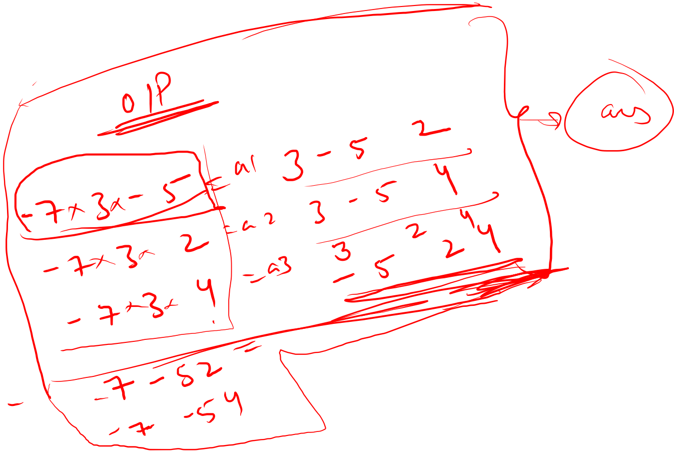
n = 5

0 4

-7 3 -5 2 4

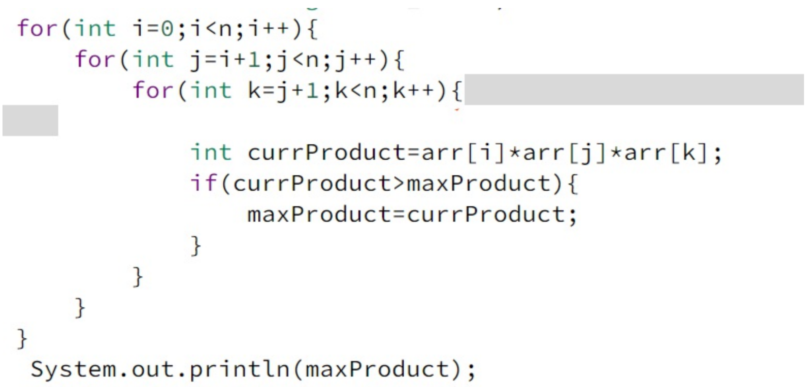
pairs →

-7 -5 2



Dry Run →

-7	3	-5	2	4



```
for(int i=0;i<n;i++){
    for(int j=i+1;j<n;j++){
        for(int k=j+1;k<n;k++){

            int currProduct=arr[i]*arr[j]*arr[k];
            if(currProduct>maxProduct){
                maxProduct=currProduct;
            }
        }
    }
}
 System.out.println(maxProduct);
```

i = 0 < 5 (T)

j = 2 < 5

-10

K = 3 < 5 (T)

arr[0] * arr[1] * arr[2]

-7 * 3 * -5 = 105

arr[0] * arr[2] * arr[3]

arr[0] * arr[2] * arr[4]

arr[0] * arr[1] * arr[3]

-7 * 3 * 2 = -42

arr[0] * arr[1] * arr[4]

-7 * 3 * 4

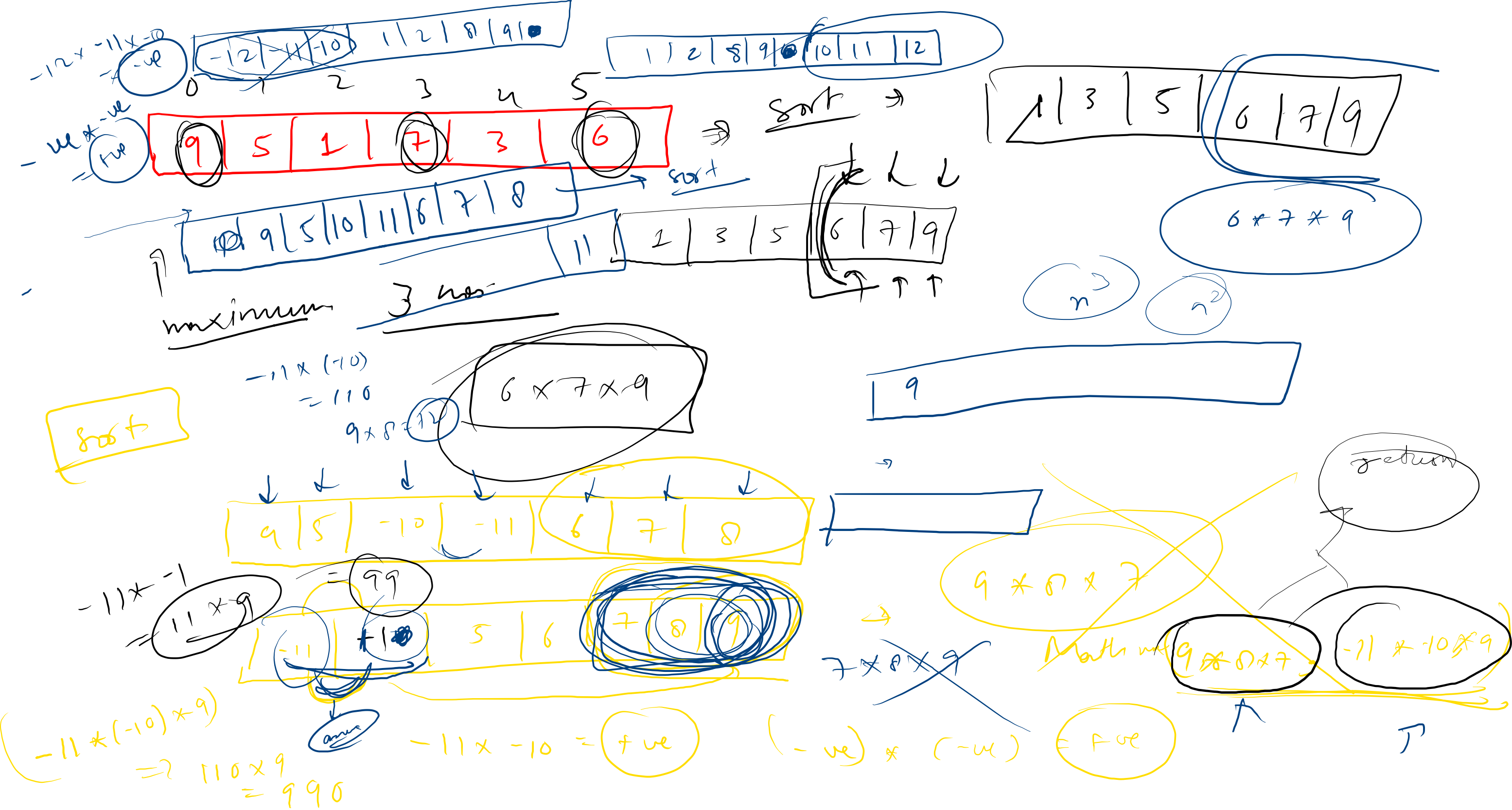
→ -84

arr [1] [3] [4]

~~[1] [4]~~

[2] [3] [3]

[1] [2] [4]


-12 x -11 x -10
-ve
-12 | -11 | -10 | 1 | 2 | 8 | 9
1 | 2 | 8 | 9 | 10 | 11 | 12
0 1 2 3 4 5
-ve * -ve
+ve
9 | 5 | 1 | 7 | 3 | 6
sort
1 | 3 | 5 | 6 | 7 | 9
9 | 5 | 10 | 11 | 6 | 7 | 8
11
sort
1 | 3 | 5 | 6 | 7 | 9
6 * 7 * 9
n^3
n^2
maximum
3 nos
-11 x (-10)
= 110
9x8=72
6 x 7 x 9
sort
9
9 | 5 | -10 | -11 | 6 | 7 | 8
-11 x -1
11 x 9
= 99
-11 | +10 | 5 | 6 | 7 | 8 | 9
9 * 8 * 7
7 * 8 * 9
return
Maximum
9 * 8 * 7
-11 * -10 * 9
ans
-11 * (-10) * 9)
=> 110 x 9
= 990
-11 x -10 = +ve
(-ve) * (-ve)
+ve