

```
Arrays.sort(arr);
```

```
int distinctCount = 0;
```

```
int distinctElem = Integer.MIN_VALUE;
```

```
for(int i = n-1; i >= 0; i-){
    if(arr[i] != distinctElem){
        distinctCount++;
        distinctElem = arr[i];
    }
    if(distinctCount == 3){
        System.out.println(arr[i]);
        return;
    }
}
```

```
// distinct elems fewer than 3
System.out.println(arr[n-1]);
```

count = 0 1 2 3
 elem = ~~0~~ 8 7 3

count == 3
 3 = 5 3 T

0/8 → 3

c = 1 2
 c = 2 1

↓ (2)

arr =

7	3	3	1	7	8
---	---	---	---	---	---

 n = 6

arr =

0	1	2	3	4	5
1	3	3	7	7	8

i = 5 ≥ 0

if (8 != -∞) T

i = 3 ≥ 0

if (7 != 7) F

i = 4 ≥ 0

if (7 != 8) T

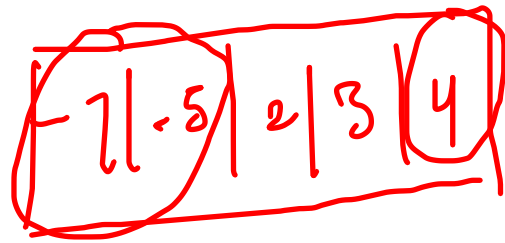
i = 2 ≥ 0

if (3 != 7) T

[TC → O(n log n) + n]

Maximum of 3 products

Arrays.sort(arr)

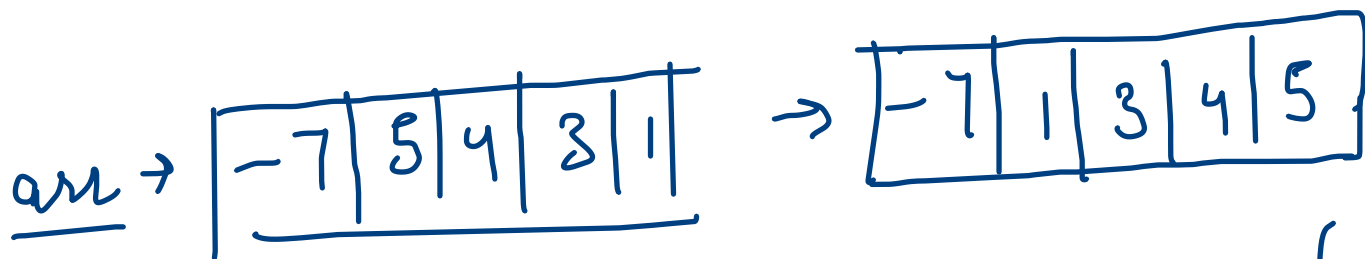


1 2 3 4 5
-7 3 -5 2 4

int p1 = arr[n-1] * arr[n-2] * arr[n-3];

int p2 = arr[0] * arr[1] * arr[n-1];

return Math.max(p1, p2);



$$p1 = 5 \times 4 \times 3 = 60$$

$$p2 = -7 \times 1 \times 5 = -35$$

$$\text{max}(60, -35)$$

$$\text{ans} \rightarrow \underline{60}$$

$$2 \times 3 \times 4 = 24$$

$$-7 \times -5 \times 2 = 70$$

$$-7 \times -5 \times 4 = 140$$

$$-5 \times -4 \times 3 = 60$$

$$-3 \times -4 \times 5 = \underline{\underline{60}}$$

$$-1 \times -2 \times 5 = -10$$