Revision

subarray → 1 2 3 4 5

1 2 3 4 5
0 1 2 3 4

n = 5

$\frac{n(n+1)}{2}$

= 15.

0 1 0

0 1 2 1

0 1 2 3 2

0 1 2 3 4 3

0 1 2 3 4 5 4

1 2 1

1 2 3 2

1 2 3 4 3

1 2 3 4 5 4

2 3 2

2 3 4 3

2 3 4 5 4

3 4 3

3 4 5 4

4 5 4

Kadane's Algo.

```
Input: nums = [-2,1,-3,4,-1,2,1,-5,4]
Output: 6
```

sum < 0 → new part.

find.

ans = -∞ -2 X X 5 6 ✓

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

-5

1

sum = -∞ -2 X -2 X 3 5 6 1 5

↑

n = 9.

ans = -∞ -2 X X 5 6

sum = 0 -2 X -2

X 3 5 6 15

= [-2, 1, -3, 4, -1, 2, 1, -5, 4]

0 1 2 3 4 5 6 7 8

```
//logic

int ans = Integer.MIN_VALUE;        // should be max.
int sum = 0;        //current sum

for(int i = 0; i < n; i++){
    if(sum < 0){
        // start new
        sum = A[i];
    }
    else{
        // sum >= 0 -> add with prev
        sum += A[i];
    }

    ans = Math.max(ans, sum);
}

System.out.println(ans);
```

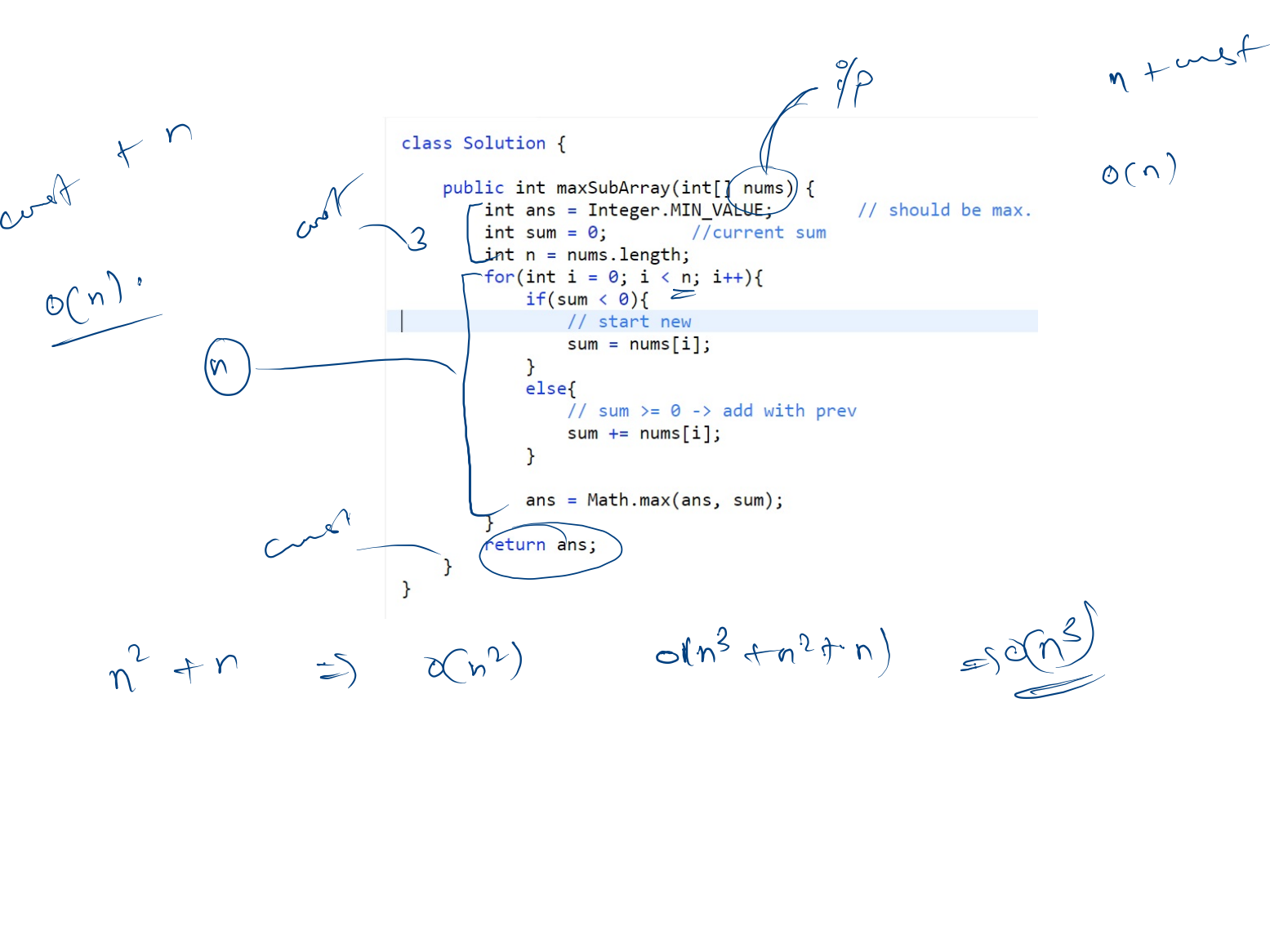
i = 9

9 < 9

```java
class Solution {

    public int maxSubArray(int[] nums) {
        int ans = Integer.MIN_VALUE;        // should be max.
        int sum = 0;        //current sum
        int n = nums.length;
        for(int i = 0; i < n; i++){
            if(sum < 0){
                // start new
                sum = nums[i];
            }
            else{
                // sum >= 0 -> add with prev
                sum += nums[i];
            }

            ans = Math.max(ans, sum);
        }
        return ans;
    }
}
```



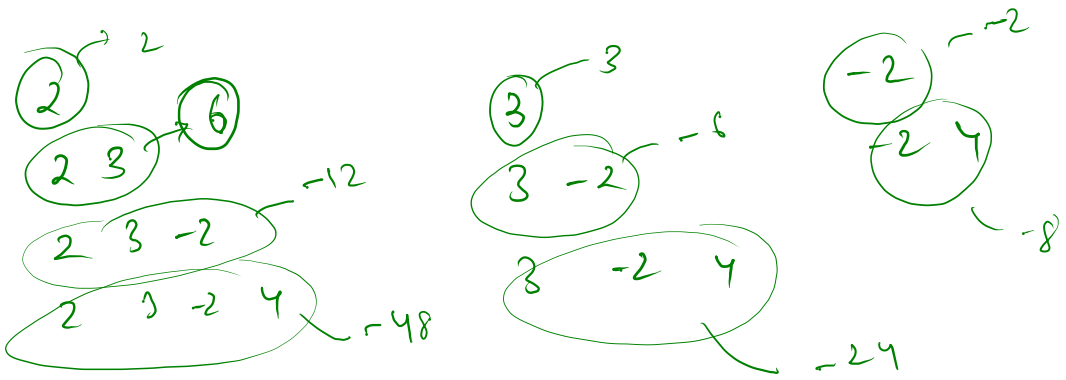
152. Maximum Product Subarray

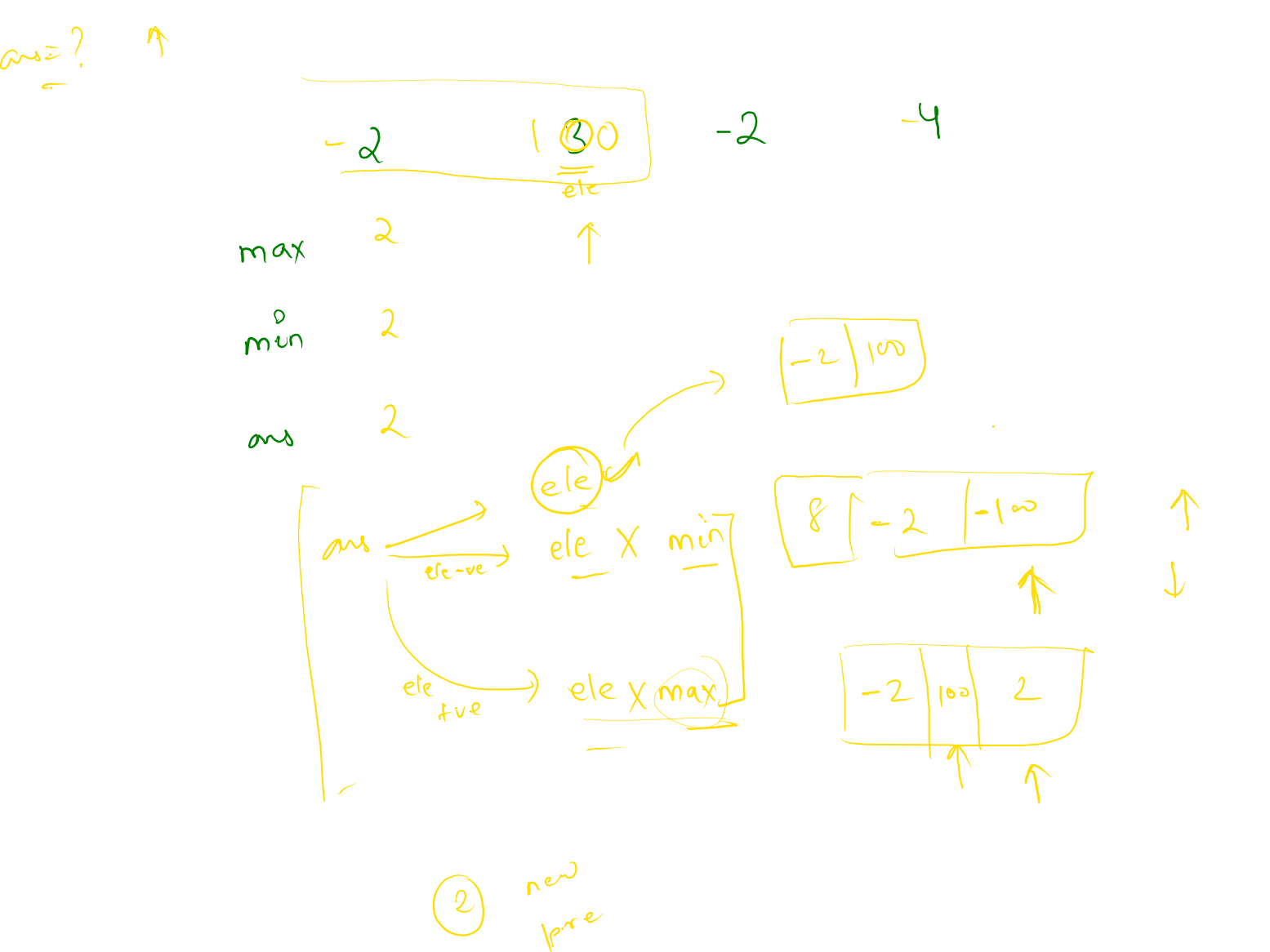
max prod. of sub.

```
Input: nums = [2,3,-2,4]
Output: 6
```

n = 4

$\frac{4(5)}{2} = 10.$




ans = ?
-2
1
3
0
ele
-2
-4
max
2
min
2
ans
2
-2 | 100
ele
ans
ele -ve
ele X min
ele +ve
ele X max
8 | -2 | -100
-2 | 100 | 2
2
new pre

if ans = $-\infty$

mps-

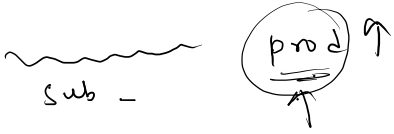
	2	3	-2	4
max	2	6	-2	9
min	2	3	-12	-48
ans	2	6	6	6

	ele	ex ma	ex min
max	3,	3X2	3X2

	-2 ,	-2X6 ,	-2X3
	4 ,	4X-2 ,	4X-12

ans = -∞ 2

	2	3	-2	4
	0	1	2	3
max	2	6	-2	4
		3	-12	-48
min	2	6	6	6
ans	2			



i	ele	ele * max	ele * min
1	3	6	6
2	-2	-12	-6
3	4	-8	-48

	2	3	-2	4	-1
max	2	6	-2	4	48
min	2	3	-12	-48	-4
ans =	2	6	6	6	48

2	3	-2

2 3 -2

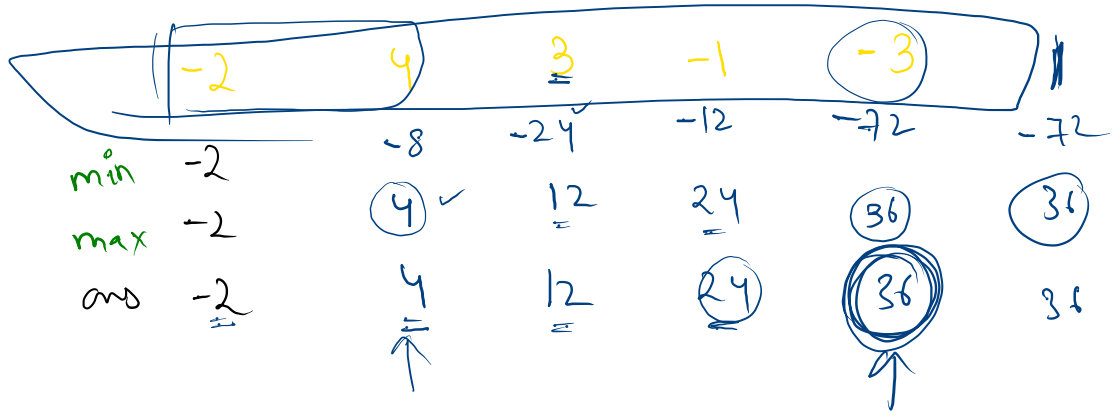
2 6

2 3 -2

3 -2

-2

i	A[i]	A[i] * max	A[i] * min
1	3	6	6
2	-2	-12	-6
3	4	-8	-48
4	-1	-4	48



i	A[i]	A[i]*max	A[i]*min
1	4	-8	-8
2	3	-24	12
3	-1	-12	24
4	-3	-72	36
	1		

-2 4 3 -3

-2

-2 4

-2 4 3

-2 4 3 -3

72

4

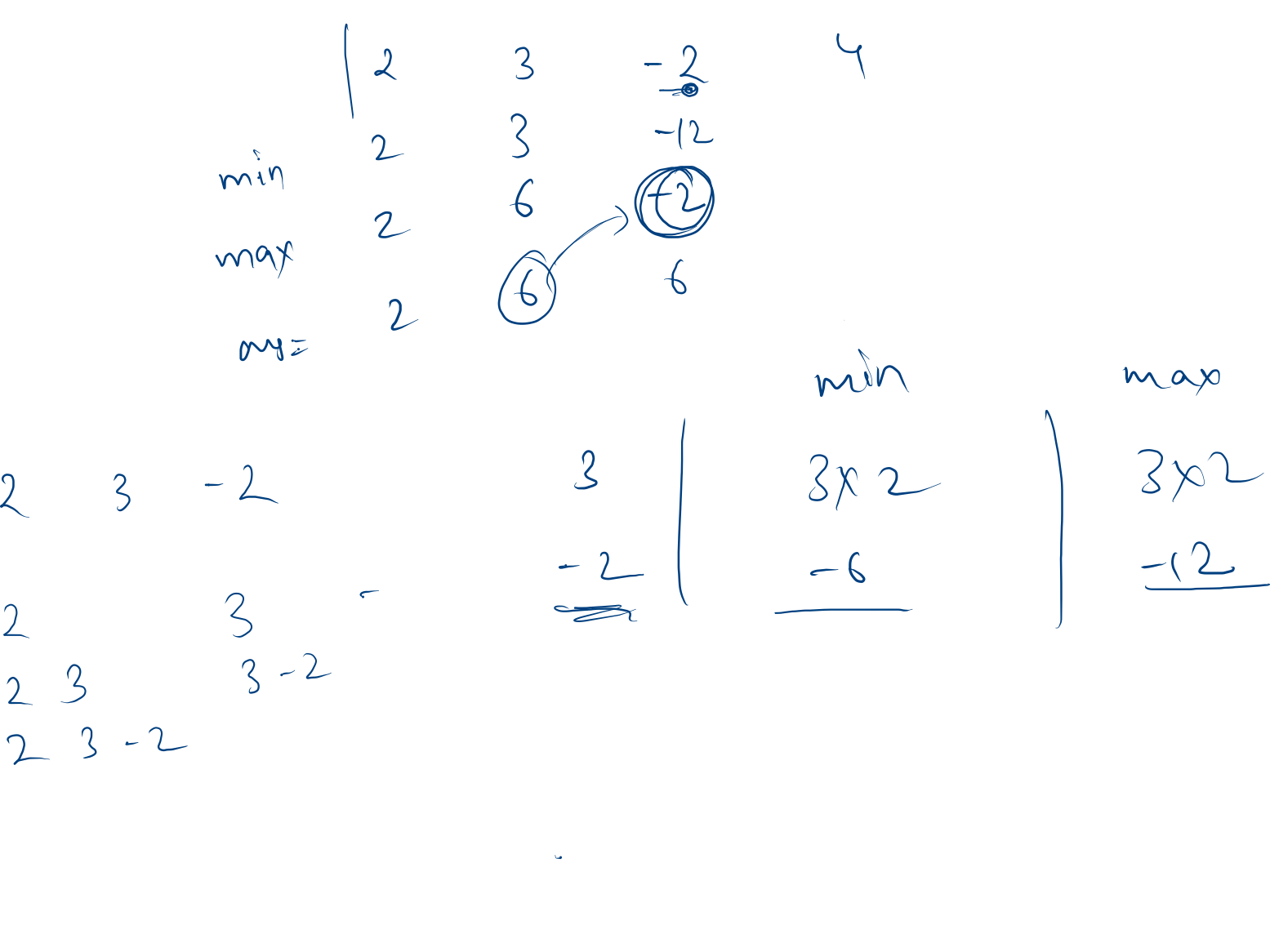
4 3

4 3 -3

3

3 -3

3


2 3 -2 4
2 3 -12
min 2 6 -2
max 2 6 6
ans 2
min
max
2 3 -2
3
-2
3x2
-6
3x2
-12
2
3
2 3
3 -2
2 3 -2

```java
class Solution {
    public int maxProduct(int[] A) {
        int min = A[0];
        int max = A[0];
        int ans = A[0];

        for(int i = 1; i < A.length; i++){
            int prevMin = min;

            min = Math.min(A[i], Math.min(A[i]*min, A[i]* max));
            max = Math.max(A[i], Math.max(A[i]*prevMin, A[i]* max));
            ans = Math.max(ans, max);
        }
        return ans;
    }
}
```

Need more
screen moc