

You are given a **0-indexed** integer array nums .

Return **the maximum value over all triplets of indices** (i, j, k) such that $i < j < k$. If all such triplets have a negative value, return 0 .

The **value of a triplet of indices** (i, j, k) is equal to $(\text{nums}[i] - \text{nums}[j]) * \text{nums}[k]$.

Example :- $\text{nums} = [12, 6, 1, 2, 7]$

Output :- 77
 $(12 - 1) * 7$

77.

Thought Process

Brute Force

...

$i < j < k$

$\begin{matrix} i & j & k \\ 0 & 1 & 2 \\ \text{nums} = [12, 6, 1, 2, 7] \end{matrix}$

we will take 3 for loops for i, j, k each.

i will start from index 0, j will start from after i &
(j will start from $i+1$)

k will start from after i & j since $i < j < k$.
(K will start from $j+1$)

Ex: the ftrs will move like this $i < j < k$

numb = $[\overset{i}{12}, \overset{j}{6}, \overset{k}{1}, \overset{3}{2}, \overset{4}{7}]$ $(12, 6, 1)$

numb = $[\overset{i}{12}, \overset{j}{6}, \overset{k}{1}, \overset{3}{2}, \overset{4}{7}]$ $(12, 6, 2)$

numb = $[\overset{i}{12}, \overset{j}{6}, \overset{k}{1}, \overset{3}{2}, \overset{4}{7}]$ $(12, 6, 7)$

numb = $[\overset{0}{12}, \overset{1}{6}, \overset{2}{1}, \overset{3}{2}, \overset{4}{7}]^k$

once k gets out of bounds
 j will move.

numb = $[\overset{i}{12}, \overset{j}{6}, \overset{k}{1}, \overset{3}{2}, \overset{4}{7}]$ $(12, 1, 2)$

numb = $[\overset{i}{12}, \overset{j}{6}, \overset{k}{1}, \overset{3}{2}, \overset{4}{7}]$ $(12, 1, 7)$

numb = $[\overset{0}{12}, \overset{1}{6}, \overset{2}{1}, \overset{3}{2}, \overset{4}{7}]^k$

again k oob j will move.

numb = $[\overset{i}{12}, \overset{j}{6}, \overset{k}{1}, \overset{3}{2}, \overset{4}{7}]$ $(12, 2, 7)$

numb = $[\overset{i}{12}, \overset{j}{6}, \overset{k}{1}, \overset{3}{2}, \overset{4}{7}]$

again k oob j will move.

numb = $[\overset{i}{12}, \overset{j}{6}, \overset{k}{1}, \overset{3}{2}, \overset{4}{7}]$

when j is here

K will be 0 or that
means j can move
only till here .

once j & k completes i moves .

& it will repeat again like before .

numb = [1⁰, 2¹, 6², 1³, 2⁴, 7]
.....
.....
.....

The max possible combination we get acc
to formula given in question will be our
answer .

for ex: $(12 - 1) * 7 = \underline{\underline{77}}$

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

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

```
        for( int K = j+1 ; K < n ; K++) { ↵
```

result = max (result, (num[i]-num[j])
* num[K]);

}

} ↵

? ↵

return result ;

TC = O(n³)

SC = O(1).

Slight Improvement

$$\underbrace{(\text{nums}[i] - \text{nums}[j]) * \text{nums}[k]}$$

maximise

$$\text{nums} = [12, 6, 1, 2, 7]$$

we would want $\text{nums}[k]$ to be bigger since its getting multiplied and we want max .

Similarly, we would want $\text{nums}[i]$ to be larger since its getting subtracted by $\text{nums}[j]$ so getting a big value won't affect the no by that much after subtraction & also since we want max we want to reduce the effect of subtraction. [we want to maximise the diff]

\therefore for any given j we want to find the possible largest i & k .

$$\text{nums} = [12, 6, \underbrace{1}_{\substack{\text{maximum} \\ i}}, \underbrace{2}_{\substack{\text{maximum} \\ k}}, 7]$$

from j 's left side we will find max i & from j 's right side we will find max k . [since $i < j < k$]

\therefore for every given j we can ~~process~~ find leftmax $_i$ & rightmax $_k$.

| | 0 | 1 | 2 | 3 | 4 |
|-----------------|---|---|---|---|---|
| leftMax $_i$ = | 0 | | | | |
| rightMax $_k$ = | | | | | |

$$\text{nums} = [12, 6, 1, 2, 7]$$

→ there is no element at left hand side of index 0
 \therefore take leftmax $_i[0] = 0$.

now let's say j is at index 1, iske left hand side mein max kaun hoga? hum already jitna max dekhte aayein hai ya to voh max hoga.
ya to ahi iss j ke left hand side jo element hai

| | 0 | 1 | 2 | 3 | 4 |
|----------------|---|---|---|---|---|
| leftMax $_i$ = | 0 | | | | |

$$\text{nums} = [12, 6, 1, 2, 7]$$

$$\text{leftmax}_i[j] = \max(\text{leftmax}[j-1], \text{nums}[j-1]);$$

$$\text{Ex: } (0, 12) \Rightarrow 12.$$

$$\text{nums} = [12, 6, 1, 2, 7]$$

\nwarrow

$$(12, 6) \Rightarrow 12.$$

| | 0 | 1 | 2 | 3 | 4 |
|----------------|---|----|----|---|---|
| leftMax $_i$ = | 0 | 12 | 12 | 1 | |

$$\text{nums} = [12, 6, 1, 2, 7]$$

$$\text{leftMaxi} = \begin{array}{c|c|c|c|c} 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 12 & 12 & 12 & \end{array}$$

$(1, 12) \Rightarrow 12$

$$\text{nums} = [12, 6, 1, 2, 7]$$

$$\text{leftMaxi} = \begin{array}{c|c|c|c|c} 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 12 & 12 & 12 & 12 \end{array}$$

$(2, 12) \Rightarrow 12$

Similarly, we will find for right hand side.

since we need to find right hand side max.

\therefore we will do traversal from right to left.

$$\text{nums} = [12, 6, 1, 2, 7]$$



$$\text{leftMaxi} = \begin{array}{c|c|c|c|c} 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 12 & 12 & 12 & 12 \end{array}$$

$$\text{rightMark} = \begin{array}{c|c|c|c|c} 0 & 1 & 2 & 3 & 4 \\ \hline & & & & 0 \end{array}$$

$\text{leftMaxi}[j] = \max(\text{leftMaxi}[j-1], \text{nums}[j-1]);$

for last index i.e 4 there is no right side element \therefore func.

Similarly,

now let's say j is at index 3, iske right hand side mein max kaun hoga? hum already jitna max dekhte aayein hai ya to voh max hoga. ya to ahi iss j ke right hand side jo element hai

| | | | | |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |
| 0 | | | | |

$$\text{nums} = [12, 6, 1, 2, 7]$$

$$\text{nums} = [12, 6, 1, 2, 7]$$

\leftarrow traversal

$$\text{leftMaxi} = \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 12 & 12 & 12 & 12 \\ \hline \end{array}$$

$$\text{rightMaxi} = \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 2 & 3 & 4 \\ \hline & & & 7 & 0 \\ \hline \end{array}$$

\rightarrow

$$\text{leftMaxi}[j] = \max(\text{leftMaxi}[j-1], \text{nums}[j-1]);$$

$$\text{rightMaxi}[j] = \max[\text{rightMaxi}[j+1], \text{nums}[j+1]];$$

(0, 7) \Rightarrow 7.

$$\text{nums} = [12, 6, 1, 2, 7]$$

\leftarrow traversal

$$\text{leftMaxi} = \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 2 & 3 & 4 \\ \hline 0 & 12 & 12 & 12 & 12 \\ \hline \end{array}$$

$$\text{rightMaxi} = \begin{array}{|c|c|c|c|c|} \hline 0 & 1 & 2 & 3 & 4 \\ \hline & & 7 & 7 & 0 \\ \hline \end{array}$$

\rightarrow

(2, 7) \Rightarrow 7

$$\text{leftMaxi}[j] = \max(\text{leftMaxi}[j-1], \text{nums}[j-1]);$$

$\text{nums} = [12, 6, 1, 2, 7]$

| 0 | 1 | 2 | 3 | 4 |
|---|----|----|----|----|
| 0 | 12 | 12 | 12 | 12 |

$$\text{leftMaxi}[j] = \max(\text{leftMaxi}[j-1], \text{nums}[j-1]);$$

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| | 7 | 7 | 7 | 0 |

$$(1, 7) \Rightarrow 7$$

$\text{nums} = [12, 6, 1, 2, 7]$

| 0 | 1 | 2 | 3 | 4 |
|---|----|----|----|----|
| 0 | 12 | 12 | 12 | 12 |

$$\text{leftMaxi}[j] = \max(\text{leftMaxi}[j-1], \text{nums}[j-1]);$$

| 0 | 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 7 | 7 | 7 | 7 | 0 |

$$(6, 7) \Rightarrow 7$$

for (int j = 1 ; j < n ; j++) {

 result = max (result , (leftMaxi[j] - nums[j])

* rightMaxK[j]);

}

return result ;

TC : O(n)

SC : O(n)

Avoiding extra Space...

$$(\underbrace{\text{nums}[i] - \text{nums}[j]}_{\text{maximize diff.}}) * \text{nums}[k]$$

$\text{nums} = [\begin{smallmatrix} 0 & 1 & 2 & 3 & 4 \\ 12 & 6 & 1 & 2 & 7 \end{smallmatrix}]$

In Approach-2, we saw we need to maximise diff by making $\text{nums}[i]$ larger so that subtraction with $\text{nums}[j]$ won't have that much effect.

\therefore we need $\max_{i=0}^n \text{diff} = 0$;
 $\max_{i=0}^n = 0$.

\hookrightarrow to maximize diff, we need largest possible $\text{nums}[i]$.

$\text{result} = 0$;

$\text{result} = \max(\text{result}, \max_{i=0}^n \text{diff} * \text{nums}[k])$;

$\text{nums} = [1^0, 2^1, 6^2, 1^3, 2^4, 7]$

$$\text{maxDiff} = 0 ; \quad \text{result} = 0$$

$$\text{maxi} = 0 ; \quad \underline{1^2}$$

$$\text{result} = \max(\text{result}, \text{maxDiff} * \text{nums}[k]);$$

$$\text{maxDiff} = (\underbrace{\text{nums}[i]}_{\downarrow} - \text{nums}[j])$$

we want this max .

$$\therefore \text{maxDiff} = (\text{maxi} - \text{nums}[k])$$

$$\text{maxi} = \max(\text{maxi}, \text{nums}[k]);$$

$$0 - 12 = -12$$

$$(0, 12) \Rightarrow 12$$

$$\text{already } \text{maxDiff} = 0 > -12$$

update maxi .

\therefore no need to update .

Similarly do it for all .

$$TC = O(n)$$

$$SC = O(1)$$

APPROACH - 2 CODE

```
1 class Solution {
2 public:
3     long long maximumTripletValue(vector<int>& nums) {
4         int n = nums.size();
5
6         vector<int> leftMaxi(n);
7         vector<int> rightMaxk(n);
8
9         for(int j = 1; j < n; j++) {
10             leftMaxi[j] = max(leftMaxi[j-1], nums[j-1]);
11         }
12
13         for(int j = n-2; j >= 0; j--) {
14             rightMaxk[j] = max(rightMaxk[j+1], nums[j+1]);
15         }
16
17         long long result = 0;
18         for(int j = 1; j < n; j++) {
19             result = max(result, (long long)(leftMaxi[j] - nums[j])*rightMaxk[j]);
20         }
21
22         return result;
23     }
24 }
25 };
```

APPROACH - 3 CODE

```
1 class Solution {
2 public:
3     long long maximumTripletValue(vector<int>& nums) {
4         int n = nums.size();
5
6         long long result = 0;
7
8         long long maxDiff = 0;
9         long long maxi = 0;
10
11         for(int k = 0; k < n; k++) {
12             result = max(result, maxDiff * nums[k]);
13             maxDiff = max(maxDiff, maxi - nums[k]);
14             maxi = max(maxi, (long long)nums[k]);
15         }
16
17         return result;
18     }
19 }
20 };
```

