→ @codestorywithmik

(Instagram, Facebook)

CSwithMIK → Twitter

→ codestorywithMIK

Leetcode

Easy 61

- Phone interview problems
- Good Practice problems

Leetcode - 3264 Easy → you'll le

WeekendWithMIK

WeekendWithMIK

@WeekendWithMIK · [redacted]

Welcome to WeekendWithMIK 🎥✨ >

[redacted]

My Life behind the Scenes ...

Motivation :-

If someone is able to do it,
You can do it too.

You just need to put the effort
that might be missing.

Trust yourself, You are already
a champion – wake up and you
will see that nothing can stop you
if you have the will Power...

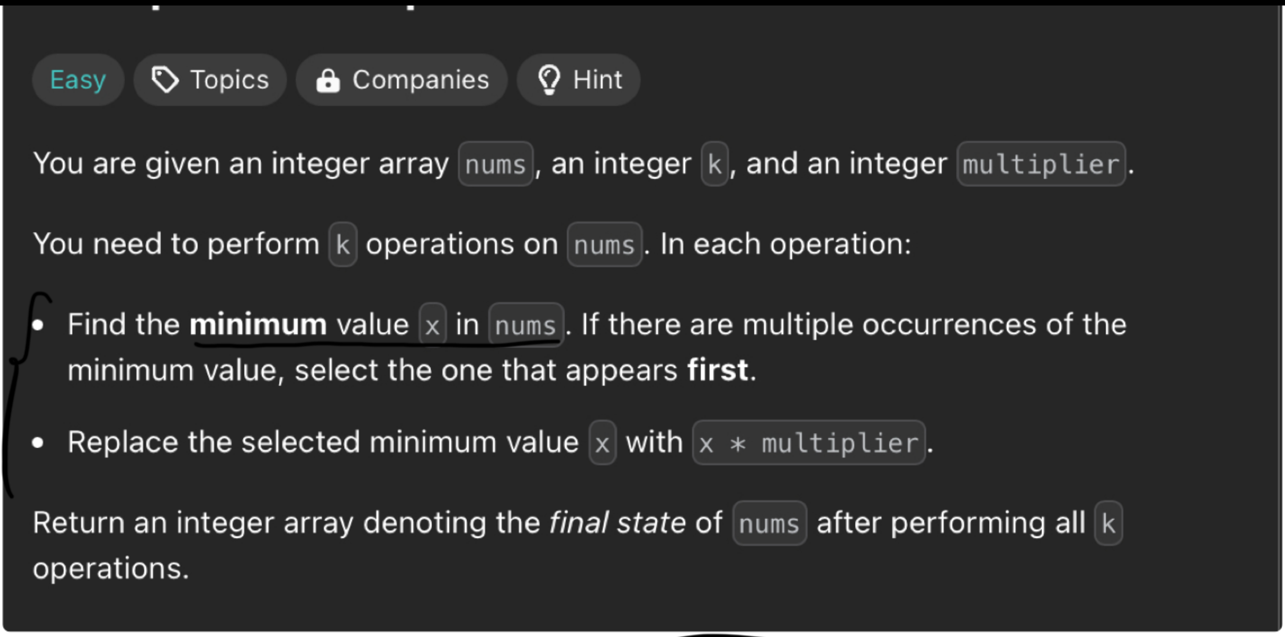
MIK...

Small Announcement

Introduction | DP On Grids | Part 1 | DP Concepts & Qns-29 | codestorywith...

3264. Final Array State After K Multiplication Operations I

Solved


Easy | Topics | Companies | Hint

You are given an integer array `nums`, an integer `k`, and an integer `multiplier`.

You need to perform `k` operations on `nums`. In each operation:

- Find the **minimum** value `x` in `nums`. If there are multiple occurrences of the minimum value, select the one that appears **first**.
- Replace the selected minimum value `x` with `x * multiplier`.

Return an integer array denoting the *final state* of `nums` after performing all `k` operations.

Example :- nums = [8, 4, 6, 5, 6]

k = ~~5~~ ~~4~~ ~~3~~ ~~2~~ ~~1~~ 0

multiplier = 2

Output :- [8, 4, 6, 5, 6] ✓

Approach ~ Brute Force
(Simulation).

$$\text{nums} = \{ \overset{0}{4}, \overset{1}{2}, \overset{2}{3}, \overset{3}{5}, \overset{4}{6} \}, \; K = \cancel{5}\ 4$$

mult = 2

min = 1 → *2 = 2

min = 2 → *2 = 4

⋮

```
K ← while (K--) {
        // Find min → O(n)
        min *= mult.
        Put it back in nums.
    }
```

T.C = O(K*n)

S.C = O(1).

Optimal Approach

$$\text{nums} = \{ \overset{0}{2}, \overset{1}{1}, \overset{2}{3}, \overset{3}{5}, \overset{4}{6} \}, \; K = 5$$

mult = 2

Min-heap = $\underline{O(1)}$

Min- Heap → {element, idx}

C++ users :-

#define P pair<int, int>

Pq → priority_queue < P , vector< P >, greater< P > Pq

```
for (i=0; i<n; i++) {   //O(n)
      pq.push ({nums[i], i}); → log(n)
}
```

O(nlogn).

```
while (k--) {

}
```
} O(Klogn)

T.C = O(nlogn + Klogn).

Build / Make Heap → Heapify

$O(N)$.

pr.<int, vec<int>, gr<int>> pq (begin (nums), end(n));

```
vector <pair <int, int>> vec ;
    for ( i = 0 ; i < n ; i++ ) {
            vec.push ( {nums[i], i} );
    }
```

# make_heap (begin (vec) , end (vec),
greater<>());

$O(n)$

vec.back();

```
while(k--) {

    // pop -> min Element

    pop_heap(begin(vec), end(vec), greater<>());
    pair<int, int> temp = vec.back();
    vec.pop_back();

        idx = temp.second;
        no  = temp.first;

        nums[idx] = no * multiplier;

    vec.push_back({nums[idx], idx});
    push_heap(begin(vec), end(vec),
                                greater<int>());
}

return nums;
```