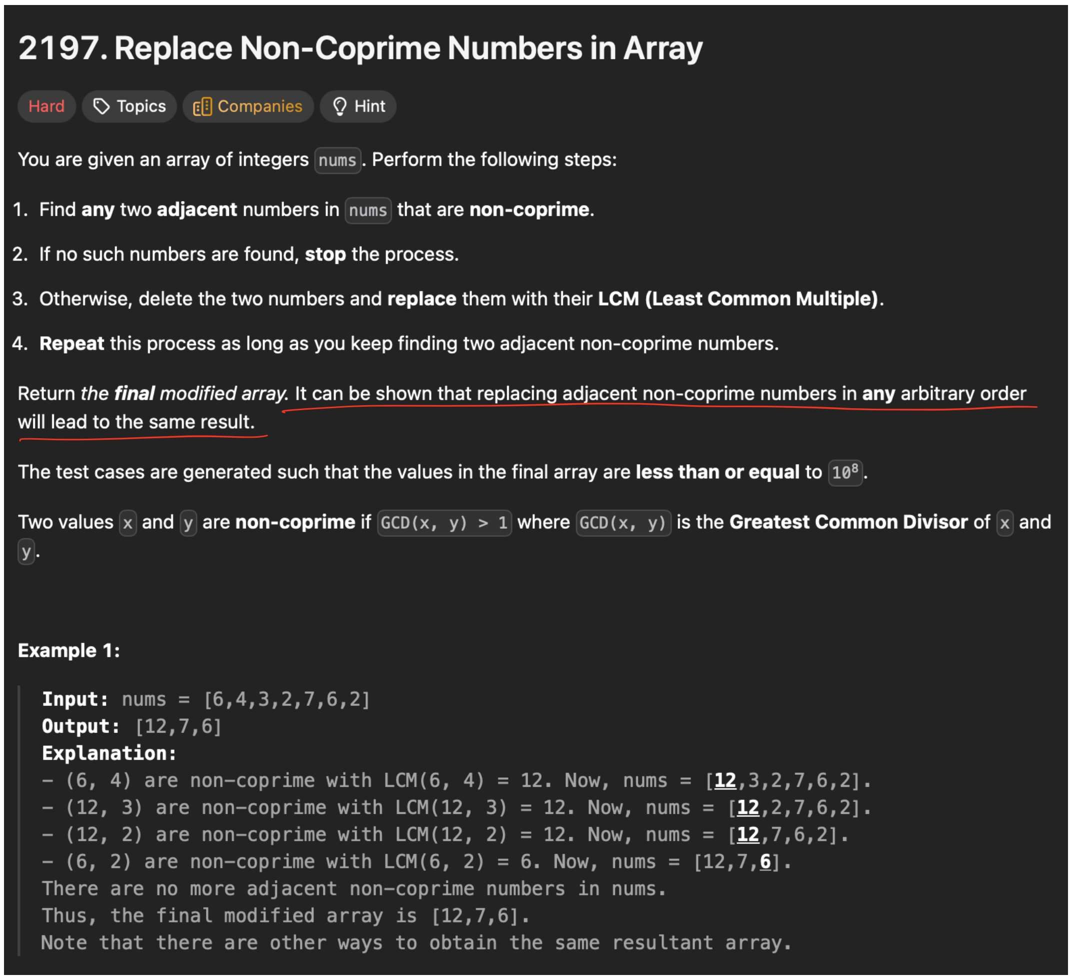


2197. Replace Non-Coprime Numbers in Array

Hard · Topics · Companies · Hint

You are given an array of integers `nums`. Perform the following steps:

1. Find **any** two **adjacent** numbers in `nums` that are **non-coprime**.
2. If no such numbers are found, **stop** the process.
3. Otherwise, delete the two numbers and **replace** them with their **LCM (Least Common Multiple)**.
4. **Repeat** this process as long as you keep finding two adjacent non-coprime numbers.

Return *the **final** modified array.* It can be shown that replacing adjacent non-coprime numbers in **any** arbitrary order will lead to the same result.

The test cases are generated such that the values in the final array are **less than or equal** to `10^8`.

Two values `x` and `y` are **non-coprime** if `GCD(x, y) > 1` where `GCD(x, y)` is the **Greatest Common Divisor** of `x` and `y`.

Example 1:

```
Input: nums = [6,4,3,2,7,6,2]
Output: [12,7,6]
Explanation:
- (6, 4) are non-coprime with LCM(6, 4) = 12. Now, nums = [12,3,2,7,6,2].
- (12, 3) are non-coprime with LCM(12, 3) = 12. Now, nums = [12,2,7,6,2].
- (12, 2) are non-coprime with LCM(12, 2) = 12. Now, nums = [12,7,6,2].
- (6, 2) are non-coprime with LCM(6, 2) = 6. Now, nums = [12,7,6].
There are no more adjacent non-coprime numbers in nums.
Thus, the final modified array is [12,7,6].
Note that there are other ways to obtain the same resultant array.
```

My rough idea:

1° 硬算 → probably Time Limit Exceed.

2° assign dictionary to each number:

Eg. $6 = \{2:1, 3:1\}$

$16 = \{2:4\}$

$\rightarrow GCD(6, 16) = \{2:\min(1,4), 3:\min(1,0)\}$

$= \{2:1\}$

$= 2 \rightarrow$ non-coprime

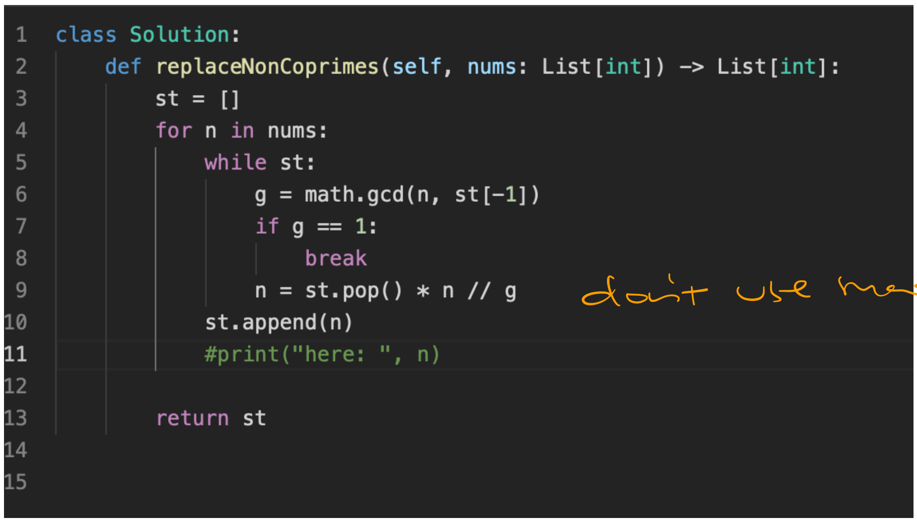
$\rightarrow 6, 16 \Rightarrow LCD(6,16) = \{2:\max(1,4), 3:\max(1,0)\}$

= {2:4 , 3:1}

= 48

5° Stack

if order doesn't matter, always operate the [0] and [1], till the end.



```
class Solution:
    def replaceNonCoprimes(self, nums: List[int]) -> List[int]:
        st = []
        for n in nums:
            while st:
                g = math.gcd(n, st[-1])
                if g == 1:
                    break
                n = st.pop() * n // g
            st.append(n)
            #print("here: ", n)

        return st
```

don't use math.lcd, it's time consuming.