2197. Replace Non Coprime Numbers in Array

Problem Description

Given an array `A` of positive integers, the task is to replace the adjacent non-coprime numbers with their Least Common Multiple (LCM), until it's not possible to do any more merges. A pair of numbers `(a, b)` are called non-coprime if their Greatest Common Divisor (GCD) is greater than 1. The final array should be as short as possible.

Example:

Input: nums = [4, 6, 7, 30]

Output: [4, 6, 7, 30]

Explanation:

- The first pair of numbers (4, 6) have GCD of 2, so they are non-coprime. Replace them with their LCM, which is 12.
- The resulting array is [12, 7, 30]. There are no more adjacent non-coprime numbers, so the array is final.

Approach

1. Initialize two pointers, `i` as a read pointer, and `j` as a write pointer, both starting from the first element of the input array `nums`.
2. Iterate over the array with the read pointer `i`. For each number `num` at position `i`, try to merge it with the previous number `ans[j-1]`, as long as they are non-coprime. The merging is done by calculating the LCM of `ans[j-1]` and `num`, replacing `ans[j-1]` with the LCM, and popping the last element of `ans`.
3. Push the merged number or the original number to the back of `ans`.
4. Repeat steps 2-3 for all elements in the input array.
5. Return the resulting array `ans` as the output.

Time Complexity

The time complexity of the algorithm is O(NlogM), where N is the number of elements in the input array, and M is the maximum number in the input array. This is because the time complexity of calculating GCD(a, b) is log(min(a, b)).

Example Walkthrough

Let's walk through the provided example:

Input: nums = [4, 6, 7, 30]

1. Initialize read pointer `i` and write pointer `j`
2. Iterate over the input array with the read pointer `i`
   - `i` = 0, num = 4, push 4 to `ans`: ans = [4]
   - `i` = 1, num = 6, GCD(4, 6) > 1, so merge 4 and 6 by calculating LCM(4, 6) = 12, ans = [12]
   - `i` = 2, num = 7, GCD(12, 7) = 1, push 7 to `ans`: ans = [12, 7]
   - `i` = 3, num = 30, GCD(7, 30) = 1, push 30 to `ans`: ans = [12, 7, 30]
3. The final array is [12, 7, 30], which is our output.

Python Solution

```
python
from math import gcd

class Solution:
    def replaceNonCoprimes(self, nums):
        ans = []

        for num in nums:
            while ans and gcd(ans[-1], num) > 1:
                num = (ans[-1] * num) // gcd(ans[-1], num)
                ans.pop()
            ans.append(num)

        return ans
```

Java Solution

```
java
import java.util.ArrayList;
import java.util.List;

class Solution {
    public List<Integer> replaceNonCoprimes(int[] nums) {
        List<Integer> ans = new ArrayList<>();

        for (int num : nums) {
            while (!ans.isEmpty() && gcd(ans.get(ans.size() - 1), num) > 1) {
                num = lcm(ans.get(ans.size() - 1), num);
                ans.remove(ans.size() - 1);
            }
            ans.add(num);
        }

        return ans;
    }

    private int gcd(int a, int b) {
        if (b == 0) {
            return a;
        } else {
            return gcd(b, a % b);
        }
    }

    private int lcm(int a, int b) {
        return a * b / gcd(a, b);
    }
}
```

JavaScript Solution

```
javascript
class Solution {
    replaceNonCoprimes(nums) {
        const ans = [];

        for (const num of nums) {
            while (ans.length > 0 && this.gcd(ans[ans.length - 1], num) > 1) {
                const temp = ans.pop();
                ans.push(this.lcm(temp, num));
            }
            ans.push(num);
        }

        return ans;
    }

    gcd(a, b) {
        if (b === 0) {
            return a;
        } else {
            return this.gcd(b, a % b);
        }
    }

    lcm(a, b) {
        return a * b / this.gcd(a, b);
    }
}
```

C++ Solution

```
cpp
#include <algorithm>
#include <vector>

class Solution {
public:
    std::vector<int> replaceNonCoprimes(std::vector<int>& nums) {
        std::vector<int> ans;

        for (int num : nums) {
            while (!ans.empty() && std::gcd(ans.back(), num) > 1) {
                num = std::lcm(ans.back(), num);
                ans.pop_back();
            }
            ans.push_back(num);
        }

        return ans;
    }
};
```

C# Solution

```
csharp
using System;
using System.Collections.Generic;

public class Solution {
    public IList<int> ReplaceNonCoprimes(int[] nums) {
        List<int> ans = new List<int>();

        foreach (int num in nums) {
            while (ans.Count > 0 && Gcd(ans[ans.Count - 1], num) > 1) {
                num = Lcm(ans[ans.Count - 1], num);
                ans.RemoveAt(ans.Count - 1);
            }
            ans.Add(num);
        }

        return ans;
    }

    private int Gcd(int a, int b) {
        if (b == 0) {
            return a;
        } else {
            return Gcd(b, a % b);
        }
    }

    private int Lcm(int a, int b) {
        return a * b / Gcd(a, b);
    }
}
```