2654. Minimum Number of Operations to Make All Array Elements Equal to 1

Solved

Medium | Topics | Companies | Hint

You are given a **0-indexed** array `nums` consisiting of **positive** integers. You can do the following operation on the array **any** number of times:

- Select an index `i` such that `0 <= i < n - 1` and replace either of `nums[i]` or `nums[i+1]` with their gcd value.

Return *the **minimum** number of operations to make all elements of* `nums` *equal to* `1`. If it is impossible, return `-1`.

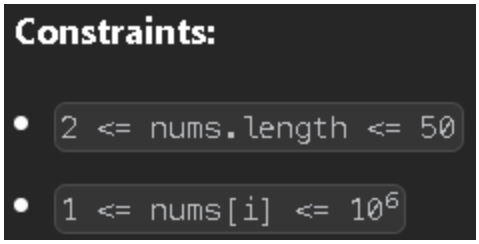
The gcd of two integers is the greatest common divisor of the two integers.

Example 1:

```
Input: nums = [2,6,3,4]
Output: 4
Explanation: We can do the following operations:
- Choose index i = 2 and replace nums[2] with gcd(3,4) = 1.
Now we have nums = [2,6,1,4].
- Choose index i = 1 and replace nums[1] with gcd(6,1) = 1.
Now we have nums = [2,1,1,4].
- Choose index i = 0 and replace nums[0] with gcd(2,1) = 1.
Now we have nums = [1,1,1,4].
- Choose index i = 2 and replace nums[3] with gcd(1,4) = 1.
Now we have nums = [1,1,1,1].
```

Example 2:

```
Input: nums = [2,10,6,14]
Output: -1
Explanation: It can be shown that it is impossible to make
all the elements equal to 1.
```



Python:

```
class Solution:
    def minOperations(self, nums: List[int]) -> int:
        n = len(nums)
        # check if there is 1 in `nums`
        ones = nums.count(1)
        # if so, we can make all the other elements equal to 1 with one operation each
        # e.g. [2,6,1,4] -> [2,1,1,4] -> [1,1,1,4] -> [1,1,1,1]
        # the number of operation to make all equal to 1 is simply n - number of 1s
        if ones: return n - ones
        res = inf
        # try finding the shortest subarray with a gcd equal to 1.
        for i in range(n):
            # subarray starting from i
            g = nums[i]
            # try each element after i
            for j in range(i + 1, n):
                # to calculate gcd
                g = gcd(g, nums[j])
                # if the gcd is 1
                if g == 1:
                    # then we calculate the min ops
                    res = min(res, j - i)
        # no result -> return -1
        if res == inf: return -1
        # otherwise, return res + n - 1
        # i.e. the min ops to turn the shortest subarray to 1 +
        #      use that 1 to convert n - 1 elements to 1
        return res + n - 1
```

JavaScript:

```
var minOperations = function (nums) {
    const n = nums.length;
    let num1 = 0;
    let g = 0;

    const gcd = (a, b) => {
```

```
        while (b !== 0) {
            const temp = b;
            b = a % b;
            a = temp;
        }
        return a;
    };

    for (const x of nums) {
        if (x === 1) {
            num1++;
        }
        g = gcd(g, x);
    }

    if (num1 > 0) {
        return n - num1;
    }
    if (g > 1) {
        return -1;
    }

    let minLen = n;
    for (let i = 0; i < n; i++) {
        let currentGcd = 0;
        for (let j = i; j < n; j++) {
            currentGcd = gcd(currentGcd, nums[j]);
            if (currentGcd === 1) {
                minLen = Math.min(minLen, j - i + 1);
                break;
            }
        }
    }
    return minLen + n - 2;
};
```

Java:

```
class Solution {
    public int minOperations(int[] nums) {
         int n = nums.length;
        // check if there is 1 in `nums`
        int ones = 0;
        for (int num : nums) {
            if (num == 1) {
```

```
            ones++;
        }
    }
    // if so, we can make all the other elements equal to 1 with one operation each
    // e.g. [2,6,1,4] -> [2,1,1,4] -> [1,1,1,4] -> [1,1,1,1]
    // the number of operation to make all equal to 1 is simply n - number of 1s
    if (ones > 0) {
        return n - ones;
    }
    int res = Integer.MAX_VALUE;
    // try finding the shortest subarray with a gcd equal to 1.
    for (int i = 0; i < n; i++) {
        // subarray starting from i
        int g = nums[i];
        // try each element after i
        for (int j = i + 1; j < n; j++) {
            // to calculate gcd
            g = gcd(g, nums[j]);
            // if the gcd is 1
            if (g == 1) {
                // then we calculate the min ops
                res = Math.min(res, j - i);
            }
        }
    }
    // no result -> return -1
    if (res == Integer.MAX_VALUE) {
        return -1;
    }
    // otherwise, return res + n - 1
    // i.e. the min ops to turn the shortest subarray to 1 +
    //      use that 1 to convert n - 1 elements to 1
    return res + n - 1;
  }

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