[LeetCode](https://leetcode.com/problems/greatest-common-divisor-traversal/description/?envType=daily-question&envId=2024-02-25)

<https://github.com/AdityaKonda6/-50DaysOfCoding>

<https://leetcode.com/problems/greatest-common-divisor-traversal/description/?envType=daily-question&envId=2024-02-25>

<https://www.linkedin.com/in/aditya-adi-konda/>

 Day 6 of [#50dayscodingchallenge](https://www.linkedin.com/feed/hashtag/?keywords=50dayscodingchallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633):  
[#leetcode](https://www.linkedin.com/feed/hashtag/?keywords=leetcode&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcodechallenge](https://www.linkedin.com/feed/hashtag/?keywords=leetcodechallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcodestreak](https://www.linkedin.com/feed/hashtag/?keywords=leetcodestreak&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcode2024](https://www.linkedin.com/feed/hashtag/?keywords=leetcode2024&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#leetcode50day](https://www.linkedin.com/feed/hashtag/?keywords=leetcode50day&highlightedUpdateUrns=urn:li:activity:7166316239483461633)  
   
Just kicked off my coding journey with a fascinating problem - "Successfully solved LeetCode Problem “2709. Greatest Common Divisor Traversal” !”  
   
✨ Task: Given a list of integers `nums`, you can traverse between indices `i` and `j` if `gcd(nums[i], nums[j]) > 1`. Task is to determine if for every pair of indices `i` and `j` where `i < j`, there exists a sequence of traversals from `i` to `j`.

Example 1:

- Input: nums = [2,3,6]

- Output: true

- Explanation: Successfully traversed all possible pairs of indices.

Example 2:

- Input: nums = [3,9,5]

- Output: false

- Explanation: Unable to traverse from index 0 to index 2. Returned false.

Example 3:

- Input: nums = [4,3,12,8]

- Output: true

- Explanation: Valid sequence of traversals for each pair, hence returned true.

Let's Connect:

If you find this problem intriguing or have insights to share, let's connect! I'm passionate about problem-solving, algorithmic thinking, and collaborative learning. Feel free to comment or reach out for engaging discussions and knowledge exchange.Unravel the mystery using your coding skills!

[#CodingChallenge](https://www.linkedin.com/feed/hashtag/?keywords=codingchallenge&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#Algorithm](https://www.linkedin.com/feed/hashtag/?keywords=algorithm&highlightedUpdateUrns=urn:li:activity:7166316239483461633) [#LinkedInPost](https://www.linkedin.com/feed/hashtag/?keywords=linkedinpost&highlightedUpdateUrns=urn:li:activity:7166316239483461633) #Algorithm #Optimization #DataStructures #CodingChallenge  
  
Excited about the progress and challenges ahead!  
   
Make Sure You Follow My GitHub For Solutions: [https://lnkd.in/d7EApJ2m](https://lnkd.in/d7EApJ2m" \t "https://www.linkedin.com/feed/_self)  
  
  
Happy coding!

**Solution:-**

class UnionFind {

  public UnionFind(int n) {

    id = new int[n];

    sz = new int[n];

    for (int i = 0; i < n; ++i)

      id[i] = i;

    for (int i = 0; i < n; ++i)

      sz[i] = 1;

  }

  public void unionBySize(int u, int v) {

    final int i = find(u);

    final int j = find(v);

    if (i == j)

      return;

    if (sz[i] < sz[j]) {

      sz[j] += sz[i];

      id[i] = j;

    } else {

      sz[i] += sz[j];

      id[j] = i;

    }

  }

  public int getSize(int i) {

    return sz[i];

  }

  private int[] id;

  private int[] sz;

  private int find(int u) {

    return id[u] == u ? u : (id[u] = find(id[u]));

  }

}

class Solution {

  public boolean canTraverseAllPairs(int[] nums) {

    final int n = nums.length;

    final int maxNum = Arrays.stream(nums).max().getAsInt();

    final int[] minPrimeFactors = sieveEratosthenes(maxNum + 1);

    Map<Integer, Integer> primeToFirstIndex = new HashMap<>();

    UnionFind uf = new UnionFind(n);

    for (int i = 0; i < n; ++i)

      for (final int primeFactor : getPrimeFactors(nums[i], minPrimeFactors))

        // `primeFactor` already appeared in the previous indices.

        if (primeToFirstIndex.containsKey(primeFactor))

          uf.unionBySize(primeToFirstIndex.get(primeFactor), i);

        else

          primeToFirstIndex.put(primeFactor, i);

    for (int i = 0; i < n; ++i)

      if (uf.getSize(i) == n)

        return true;

    return false;

  }

  // Gets the minimum prime factor of i, where 1 < i <= n.

  private int[] sieveEratosthenes(int n) {

    int[] minPrimeFactors = new int[n + 1];

    for (int i = 2; i <= n; ++i)

      minPrimeFactors[i] = i;

    for (int i = 2; i \* i < n; ++i)

      if (minPrimeFactors[i] == i) // `i` is prime.

        for (int j = i \* i; j < n; j += i)

          minPrimeFactors[j] = Math.min(minPrimeFactors[j], i);

    return minPrimeFactors;

  }

  private List<Integer> getPrimeFactors(int num, int[] minPrimeFactors) {

    List<Integer> primeFactors = new ArrayList<>();

    while (num > 1) {

      final int divisor = minPrimeFactors[num];

      primeFactors.add(divisor);

      while (num % divisor == 0)

        num /= divisor;

    }

    return primeFactors;

  }

}

