[LeetCode](https://leetcode.com/problems/find-all-people-with-secret/description/?envType=daily-question&envId=2024-02-24)

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

<https://leetcode.com/problems/find-all-people-with-secret/description/?envType=daily-question&envId=2024-02-24>

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

 Day 5 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 “2092. Find All People With Secret” !”  
   
✨ Task: Problem-Solving Showcase: Unraveling Secrets Through Meetings

Hello LinkedIn community! I recently encountered an intriguing problem that involves unraveling secrets through a series of meetings. Let's dive into it!

Problem Overview:

You are given a scenario with `n` people, each identified by a unique number from 0 to n-1. The interactions between individuals are defined by a 2D array `meetings`, where `meetings[i] = [xi, yi, timei]` signifies a meeting between person `xi` and person `yi` at time `timei`. The twist is that a secret is initially shared by person 0 and cascades through subsequent meetings. The objective is to identify all individuals who possess the secret after all the meetings.

Example Scenarios:

Example 1:

- Input: n = 6, meetings = [[1,2,5],[2,3,8],[1,5,10]], firstPerson = 1

- Output: [0,1,2,3,5]

- Explanation: The secret travels through meetings, and eventually, individuals 0, 1, 2, 3, and 5 are in the know.

Example 2:

- Input: n = 4, meetings = [[3,1,3],[1,2,2],[0,3,3]], firstPerson = 3

- Output: [0,1,3]

- Explanation: The secret-sharing unfolds differently, and only individuals 0, 1, and 3 are privy to the secret.

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) [#TownJudge](https://www.linkedin.com/feed/hashtag/?keywords=townjudge&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];

    rank = new int[n];

    for (int i = 0; i < n; ++i)

      id[i] = i;

  }

  public void unionByRank(int u, int v) {

    final int i = find(u);

    final int j = find(v);

    if (i == j)

      return;

    if (rank[i] < rank[j]) {

      id[i] = j;

    } else if (rank[i] > rank[j]) {

      id[j] = i;

    } else {

      id[i] = j;

      ++rank[j];

    }

  }

  public boolean connected(int u, int v) {

    return find(u) == find(v);

  }

  public void reset(int u) {

    id[u] = u;

  }

  private int[] id;

  private int[] rank;

  private int find(int u) {

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

  }

}

class Solution {

  public List<Integer> findAllPeople(int n, int[][] meetings, int firstPerson) {

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

    UnionFind uf = new UnionFind(n);

    TreeMap<Integer, List<Pair<Integer, Integer>>> timeToPairs = new TreeMap<>();

    uf.unionByRank(0, firstPerson);

    for (int[] m : meetings) {

      timeToPairs.putIfAbsent(m[2], new ArrayList<>());

      timeToPairs.get(m[2]).add(new Pair<>(m[0], m[1]));

    }

    for (List<Pair<Integer, Integer>> pairs : timeToPairs.values()) {

      Set<Integer> peopleUnioned = new HashSet<>();

      for (Pair<Integer, Integer> pair : pairs) {

        final int x = pair.getKey();

        final int y = pair.getValue();

        uf.unionByRank(x, y);

        peopleUnioned.add(x);

        peopleUnioned.add(y);

      }

      for (final int person : peopleUnioned)

        if (!uf.connected(person, 0))

          uf.reset(person);

    }

    for (int i = 0; i < n; ++i)

      if (uf.connected(i, 0))

        ans.add(i);

    return ans;

  }

}

