In a project, you have a list of required skills req\_skills, and a list of people. The ith person people[i] contains a list of skills that the person has.

Consider a sufficient team: a set of people such that for every required skill in req\_skills, there is at least one person in the team who has that skill. We can represent these teams by the index of each person.

* For example, team = [0, 1, 3] represents the people with skills people[0], people[1], and people[3].

Return *any sufficient team of the smallest possible size, represented by the index of each person*. You may return the answer in **any order**.

It is **guaranteed** an answer exists.

**Example 1:**

Input: req\_skills = ["java","nodejs","reactjs"], people = [["java"],["nodejs"],["nodejs","reactjs"]]  
Output: [0,2]

**Example 2:**

Input: req\_skills = ["algorithms","math","java","reactjs","csharp","aws"], people = [["algorithms","math","java"],["algorithms","math","reactjs"],["java","csharp","aws"],["reactjs","csharp"],["csharp","math"],["aws","java"]]  
Output: [1,2]

**Constraints:**

* 1 <= req\_skills.length <= 16
* 1 <= req\_skills[i].length <= 16
* req\_skills[i] consists of lowercase English letters.
* All the strings of req\_skills are **unique**.
* 1 <= people.length <= 60
* 0 <= people[i].length <= 16
* 1 <= people[i][j].length <= 16
* people[i][j] consists of lowercase English letters.
* All the strings of people[i] are **unique**.
* Every skill in people[i] is a skill in req\_skills.
* It is guaranteed a sufficient team exists.