TCS NQT 2025

15th march shift 2

Ques 5:

Problem Statement: Minimum Team Selection for Required Skills

You are given a set of required skills and a number of candidates, where each candidate possesses certain skills. Your task is to form a team with the minimum number of people that collectively covers all the required skills.

Input Format:

A single line containing:

1. A space-separated list of required skills.
2. A comma (,) followed by the number of candidates.
3. A comma-separated list where each part contains space-separated skills of a candidate.

Output Format:

A space-separated list of indices representing the minimum number of candidates needed to cover all required skills.

Example:

Input:

```
a b c d , 4 , a b , c , c d , c
```

Output:

```
0 2
```

Explanation:

- The required skills are {a, b, c, d}.
- There are 4 candidates:
  - Candidate 0: {a, b}
  - Candidate 1: {c}
  - Candidate 2: {c, d}
  - Candidate 3: {c}
- The minimum team is [0, 2] because:
  - Candidate 0 covers {a, b}.
  - Candidate 2 covers {c, d},Together, they cover {a, b, c, d} with the fewest people.

```cpp
#include <iostream>
#include <vector>
#include <unordered_map>
#include <sstream>

using namespace std;

int target_mask;
vector<int> result;
unordered_map<string, int> dp;

void solve(vector<int>& people_skill, int idx, vector<int>& temp, int mask) {
   if (idx == people_skill.size()) {
      if (mask == target_mask && (result.empty() || temp.size() < result.size()))
         result = temp;
      return;
   }
   string key = to_string(idx) + "_" + to_string(mask);
   if ((dp.count(key) && dp[key] <= temp.size()) || (!result.empty() && temp.size() >=
result.size())) return;

   solve(people_skill, idx + 1, temp, mask);
   if ((mask | people_skill[idx]) != mask) {
      temp.push_back(idx);
      solve(people_skill, idx + 1, temp, mask | people_skill[idx]);
      temp.pop_back();
      dp[key] = temp.empty() ? -1 : temp.size();
   }
}

vector<int> smallestSufficientTeam(vector<string>& req_skills,
vector<vector<string>>& people) {
   unordered_map<string, int> skills;
   vector<int> people_skill;
   int n = req_skills.size();
   for (int i = 0; i < n; ++i) skills[req_skills[i]] = i;
   for (auto& v : people) {
      int skill_bit = 0;
      for (string& skill : v) skill_bit |= 1 << skills[skill];
      people_skill.push_back(skill_bit);
   }
   target_mask = (1 << n) - 1;
   solve(people_skill, 0, {}, 0);
   return result;
}
```

```cpp
void parseInput(string input, vector<string>& skills, vector<vector<string>>& people)
{
 stringstream ss(input); string part;
 vector<string> parts;
 while (getline(ss, part, ',')) parts.push_back(part);

 stringstream ss1(parts[0]); string skill;
 while (ss1 >> skill) skills.push_back(skill);

 for (size_t i = 2; i < parts.size(); i++) {
 vector<string> personSkills;
 stringstream ss2(parts[i]);
 while (ss2 >> skill) personSkills.push_back(skill);
 people.push_back(personSkills);
 }
}

int main() {
 string input;
 cout << "Enter input string: ";
 getline(cin, input);

 vector<string> skills;
 vector<vector<string>> people;
 parseInput(input, skills, people);

 vector<int> team = smallestSufficientTeam(skills, people);
 if (team.empty()) cout << "-1\n";
 else for (size_t i = 0; i < team.size(); i++) cout << team[i] << (i == team.size() - 1 ? "" : " ");
 cout << endl;
 return 0;
}
```

PYTHON

```
import sys

def solve(people_skills, idx, temp, mask):
    global result, target_mask, dp
    if idx == len(people_skills):
        if mask == target_mask and (not result or len(temp) < len(result)):
            result = temp[:]
        return
    key = (idx, mask)
    if key in dp and dp[key] <= len(temp): return
    if result and len(temp) >= len(result): return

    solve(people_skills, idx + 1, temp, mask)
    new_mask = mask | people_skills[idx]
    if new_mask != mask:
        temp.append(idx)
        solve(people_skills, idx + 1, temp, new_mask)
        temp.pop()
        dp[key] = len(temp) if temp else -1

def smallest_sufficient_team(req_skills, people):
    global target_mask, result, dp
    skill_map = {skill: i for i, skill in enumerate(req_skills)}
    people_skills = [sum(1 << skill_map[s] for s in p if s in skill_map) for p in people]

    target_mask, result, dp = (1 << len(req_skills)) - 1, [], {}
    solve(people_skills, 0, [], 0)
    return result

def parse_input(input_str):
    parts = [p.strip() for p in input_str.split(",")]
    skills = parts[0].split()
    people = [p.split() for p in parts[2:]]
    return skills, people

if __name__ == "__main__":
    input_str = sys.stdin.read().strip()
    skills, people = parse_input(input_str)
    team = smallest_sufficient_team(skills, people)
    print("-1" if not team else " ".join(map(str, team)))
```

JAVA

```
import java.util.*;

public class Main {
  private static int targetMask;
  private static List<Integer> result = new ArrayList<>();
  private static Map<String, Integer> dp = new HashMap<>();

  private static void solve(List<Integer> peopleSkills, int idx, List<Integer> temp, int mask) {
    if (idx == peopleSkills.size()) {
      if (mask == targetMask && (result.isEmpty() || temp.size() < result.size())) result = new
ArrayList<>(temp);
      return;
    }
    String key = idx + "_" + mask;
    if (dp.containsKey(key) && dp.get(key) <= temp.size()) return;
    if (!result.isEmpty() && temp.size() >= result.size()) return;

    solve(peopleSkills, idx + 1, temp, mask);
    int newMask = mask | peopleSkills.get(idx);
    if (newMask != mask) {
      temp.add(idx);
      solve(peopleSkills, idx + 1, temp, newMask);
      temp.remove(temp.size() - 1);
      dp.put(key, temp.isEmpty() ? -1 : temp.size());
    }
  }

  private static List<Integer> smallestSufficientTeam(List<String> reqSkills, List<List<String>>
people) {
    int n = reqSkills.size();
    Map<String, Integer> skillMap = new HashMap<>();
    for (int i = 0; i < n; i++) skillMap.put(reqSkills.get(i), i);

    List<Integer> peopleSkills = new ArrayList<>();
    for (List<String> person : people) {
      int skillBit = 0;
      for (String skill : person) if (skillMap.containsKey(skill)) skillBit |= 1 << skillMap.get(skill);
      peopleSkills.add(skillBit);
    }
    targetMask = (1 << n) - 1;
    solve(peopleSkills, 0, new ArrayList<>(), 0);
    return result;
    }

    private static void parseInput(String input, List<String> skills, List<List<String>> people) {
    String[] parts = input.split(",");
    skills.addAll(Arrays.asList(parts[0].trim().split("\\s+")));
    for (int i = 2; i < parts.length; i++) people.add(Arrays.asList(parts[i].trim().split("\\s+")));
    }

    public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.println("Enter input string:");
    String input = sc.nextLine();
    sc.close();

    List<String> skills = new ArrayList<>();
    List<List<String>> people = new ArrayList<>();
    parseInput(input, skills, people);

    List<Integer> team = smallestSufficientTeam(skills, people);
    System.out.println(team.isEmpty() ? "-1" : team.toString().replaceAll("[\\[\\],]", ""));
    }
}
```

Ques 2:

Question:

You are given a list of integers of length N. Every element in the list appears exactly two times, except for one unique element, which appears exactly once. Your task is to find and print this unique element.

Input Format:

- **The first line contains an integer N, the length of the list.**
- **The second line contains N space-separated integers representing the elements of the list.**

Output Format:

- **Print the unique element that appears exactly once.**

Constraints: $1 \leq N \leq 10^5$

All elements except one appear exactly twice.

Example 1:

📌 **Input:**

9

1 1 2 2 5 6 6 7 7

📌 **Output:**

5

Example 2:

📌 **Input:**

7

3 3 4 4 9 2 2

📌 **Output:**

9

JAVA

```
import java.util.Scanner;

public class UniqueElement {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        int N = sc.nextInt();
        int unique_element = 0;

        for (int i = 0; i < N; i++) {
            int num = sc.nextInt();
            unique_element ^= num;
        }

        System.out.println(unique_element);
        sc.close();
    }
}
```

```cpp
#include <iostream>
#include <vector>

using namespace std;

int main() {
    int N;
    cin >> N;
    vector<int> arr(N);

    for (int i = 0; i < N; i++) {
        cin >> arr[i];
    }

    int unique_element = 0;
    for (int num : arr) {
        unique_element ^= num;
    }

    cout << unique_element << endl;
    return 0;
}
```

PYTHON

```
N = int(input())
arr = list(map(int, input().split()))

unique_element = 0
for num in arr:
    unique_element ^= num

print(unique_element)
```