Save Your Privacy!

JAG Domestic Practice 2012: Problem A

原案: 三廻部
作成: 酒井・田山

概要

個人情報を漏洩させた犯人を特定しよう！

Sample 1

```
3
2 2 3   // 2も3も知ってるので犯人かも
1 1     // 2を知らないので犯人じゃない
1 1     // 2を知らないので犯人じゃない
2 2 3
=> 1 が犯人
```

Sample 2

```
3
2 2 3   // 2を知っているので犯人かも
1 3     // 犯人じゃない
1 2     // 犯人かも
1 2
=> -1 (犯人らしい人が2人いて特定不能)
```

Sample 3

(省略)

Sample 4

```
3
2 2 1    // 3を知らないので犯人じゃない
1 1      // 3を知らないので犯人じゃない
1 2      // 3を知らないので犯人じゃない
3 3 2 1
=> -1 (犯人らしい人がいない)
```

判定

ある人が疑わしい（犯人である可能性がある）というのは
「全ての漏れた情報をその人が知っている」ということに等しい

こんな関数を作りましょう

```
/**
 * ある人が疑わしいかどうかを判定する。
 * int know[]: その人が知っている情報の配列
 * int know_len: know の要素数
 * int leaked[]: 漏洩した情報の配列
 * int leaked_len: leaked の要素数
 * returns: 疑わしいなら 1, そうでなければ 0
 */
int isSuspicous(int know[], int know_len,
                int leaked[], int leaked_len);
```

実装

isSuspicous は単純な2重ループで実装できる

- 計算量 $O(n^2)$ (isSuspicous を n 回実行する必要があるので全体で $O(n^3)$)

```
int isSuspicious(int know[], int know_len,
                 int leaked[], int leaked_len) {
  for (int i = 0; i < leaked_len; ++i) {
    int ok = 0;
    for (int j = 0; j < know_len; ++j) {
      if (leaked[i] == know[j]) {
        ok = 1; // leaked[i] を知ってる！
      }
    }
    if (ok != 1) {
      // leaked[i] をこの人は知らないので無罪
      return 0;
    }
  }
  return 1;
}
```

中級編

- 国内予選はスピードレースになることが多い
  - 3 - 4 問解けたチームのうち、ペナルティの少ないチームまでが通過
- 突破を目指すなら、簡単な問題に時間を割いてはいられない！

- 速く解くには？
  - プログラムを短く書こう！
- 短く書くには？
  - ライブラリを活用しよう！

中級編

- isSuspicous の内側のループは「この配列の中にこの要素がありますか？」という判定
  - よくある処理なので、ライブラリ化されている
  - 以下、C++でのお話
- find: 配列の中から特定の要素を検索する
- count: 配列の中に特定の要素がいくつあるかを返す
- 計算量: $O(n^2)$ (全体で $O(n^3)$)

```
int isSuspicous(int know[], int know_len,
                int leaked[], int leaked_len) {
  for(int i = 0; i < leaked_len; ++i) {
    // if (count(know, know + know_len, leaked[i]) == 1)  でもよい
    if (find(know, know + know_len, leaked[i]) != know + know_len){
      // leadked[i] を知らないので無罪
      return 0;
    }
    return 1;
  }
}
```

上級編

マイナーな関数を使ってみよう！

- 集合の包含判定をしてくれる includes という関数があり、これを使えば一発
- 計算量 O(n log(n)) (全体で $O(n^2 \log(n))$)

```
int isSuspicious(int know[], int know_len,
                 int leaked[], leaked_len) {
  sort(know, know + know_len);
  sort(leaked, leaked + leaked_len);
  return includes(know, know + know_len,
                  leaked, leaked + leaked_len) ? 1 : 0;
}
```

- ライブラリには意外と高機能な関数もあり、活用するとコーディング/デバッグ時間を大きく節約できる
- 一方、使い方を熟知していないとバグるので注意！
  - includes はあらかじめ配列をソートしておかないと動かない！
    - この問題の入力もさりげなくソートされていない (e.g. sample 4)
  - うろ覚えのまま使ってバグるよりも、いっそ使わないでおく勇気も必要

Summary

First Accept: negainoid (1491 secs)

* (サーバトラブルのため参考記録)

Total Accept: 104