1. 완주하지 못한 선수

문제 설명

수많은 마라톤 선수들이 마라톤에 참여하였습니다. 단 한 명의 선수를 제외하고는 모든 선수가 마라톤을 완주하였습니다.

마라톤에 참여한 선수들의 이름이 담긴 배열 participant 와 완주한 선수들의 이름이 담긴 배열 completion 이 주어질 때, 완주하지 못한 선수의 이름을 return 하도록 solution 함수를 작성해주세요.

제한사항

- 마라톤 경기에 참여한 선수의 수는 1 명 이상 100,000 명 이하입니다.
- completion 의 길이는 participant 의 길이보다 1 작습니다.
- 참가자의 이름은 1 개 이상 20 개 이하의 알파벳 소문자로 이루어져 있습니다.
- 참가자 중에는 동명이인이 있을 수 있습니다.

입출력 예

participant	completion	return
[leo, kiki, eden]	[eden, kiki]	leo
[marina, josipa, nikola, vinko, filipa]	[josipa, filipa, marina, nikola]	vinko
[mislav, stanko, mislav, ana]	[stanko, ana, mislav]	mislav

입출력 예 설명

예제 #1
leo 는 참여자 명단에는 있지만, 완주자 명단에는 없기 때문에 완주하지 못했습니다.

예제 #2
vinko 는 참여자 명단에는 있지만, 완주자 명단에는 없기 때문에 완주하지 못했습니다.

예제 #3
mislav 는 참여자 명단에는 두 명이 있지만, 완주자 명단에는 한 명밖에 없기 때문에 한명은 완주하지 못했습니다.

풀이:

```
import java.util.Arrays;class Solution {
    public String solution(String[] participant, String[]
completion) {
```

```
        String answer = "";
        String temp = "";

        Arrays.sort(participant);
        Arrays.sort(completion);

        int i = 0;

        while(i < completion.length){
            if(!completion[i].equals(participant[i])){
                temp = participant[i];
                break;
            }else{
                i++;
            }
        }

        if(!temp.equals("")){
            answer = temp;
        }else{
            answer = participant[participant.length-1];
        }

        return answer;
    }
}
```

https://medium.com/@nsh235482/java-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%A8%B8%EC%8A%A4-hash-lv1-%EC%99%84%EC%A3%BC%ED%95%98%EC%A7%80-%EB%AA%BB%ED%95%9C-%EC%84%A0%EC%88%98-1ddf416516ad

2. 전화번호 목록

문제 설명

전화번호부에 적힌 전화번호 중, 한 번호가 다른 번호의 접두어인 경우가 있는지 확인하려 합니다.
전화번호가 다음과 같을 경우, 구조대 전화번호는 영석이의 전화번호의 접두사입니다.

- 구조대 : 119
- 박준영 : 97 674 223
- 지영석 : 11 9552 4421

전화번호부에 적힌 전화번호를 담은 배열 phone_book 이 solution 함수의 매개변수로 주어질 때, 어떤 번호가 다른 번호의 접두어인 경우가 있으면 false 를 그렇지 않으면 true 를 return 하도록 solution 함수를 작성해주세요.

제한 사항

- phone_book 의 길이는 1 이상 1,000,000 이하입니다.
- 각 전화번호의 길이는 1 이상 20 이하입니다.

입출력 예제

phone_book	return
[119, 97674223, 1195524421]	false
[123,456,789]	true
[12,123,1235,567,88]	false

입출력 예 설명

입출력 예 #1
앞에서 설명한 예와 같습니다.

입출력 예 #2
한 번호가 다른 번호의 접두사인 경우가 없으므로, 답은 true 입니다.

입출력 예 #3
첫 번째 전화번호, "12"가 두 번째 전화번호 "123"의 접두사입니다. 따라서 답은 false 입니다.

알림

2019 년 5 월 13 일, 테스트 케이스가 변경되었습니다. 이로 인해 이전에 통과하던 코드가 더 이상 통과하지 않을 수 있습니다

```
public boolean startsWith(String prefix, int toffset) {
       char ta[] = value;
       int to = toffset;
       char pa[] = prefix.value;
       int po = 0;
       int pc = prefix.value.length;
       // Note: toffset might be near -1>>>1.
       if ((toffset < 0) || (toffset > value.length - pc)) {
          return false;
       }
       while (--pc >= 0) {
          if (ta[to++] != pa[po++]) {
             return false;
          }
       }
       return true;
    }
```

```
class Solution {
   public boolean solution(String[] phoneBook) {
      for(int i=0; i<phoneBook.length-1; i++) {
          for(int j=i+1; j<phoneBook.length; j++) {
             if(phoneBook[i].startsWith(phoneBook[j])) {return
false;}
             if(phoneBook[j].startsWith(phoneBook[i])) {return
false;}
          }
       }
       return true;
    }
}
```

https://codevang.tistory.com/290

3. 위장

문제 설명

스파이들은 매일 다른 옷을 조합하여 입어 자신을 위장합니다.

예를 들어 스파이가 가진 옷이 아래와 같고 오늘 스파이가 동그란 안경, 긴 코트, 파란색 티셔츠를 입었다면 다음날은 청바지를 추가로 입거나 동그란 안경 대신 검정 선글라스를 착용하거나 해야 합니다.

종류	이름
얼굴	동그란 안경, 검정 선글라스
상의	파란색 티셔츠
하의	청바지
겉옷	긴 코트

스파이가 가진 의상들이 담긴 2 차원 배열 clothes 가 주어질 때 서로 다른 옷의 조합의 수를 return 하도록 solution 함수를 작성해주세요.

제한사항

- clothes 의 각 행은 [의상의 이름, 의상의 종류]로 이루어져 있습니다.
- 스파이가 가진 의상의 수는 1 개 이상 30 개 이하입니다.
- 같은 이름을 가진 의상은 존재하지 않습니다.
- clothes 의 모든 원소는 문자열로 이루어져 있습니다.
- 모든 문자열의 길이는 1 이상 20 이하인 자연수이고 알파벳 소문자 또는 '_' 로만 이루어져 있습니다.
- 스파이는 하루에 최소 한 개의 의상은 입습니다.

입출력 예

clothes	return
[[yellow_hat, headgear], [blue_sunglasses, eyewear], [green_turban, headgear]]	5
[[crow_mask, face], [blue_sunglasses, face], [smoky_makeup, face]]	3

입출력 예 설명

예제 #1
headgear 에 해당하는 의상이 yellow_hat, green_turban 이고 eyewear 에 해당하는 의상이 blue_sunglasses 이므로 아래와 같이 5 개의 조합이 가능합니다.

1. yellow_hat
2. blue_sunglasses
3. green_turban
4. yellow_hat + blue_sunglasses
5. green_turban + blue_sunglasses

예제 #2
face 에 해당하는 의상이 crow_mask, blue_sunglasses, smoky_makeup 이므로 아래와 같이 3 개의 조합이 가능합니다.

1. crow_mask
2. blue_sunglasses
3. smoky_makeup

```
import java.util.HashMap;

class Solution {
   public int solution(String[][] clothes) {
       int answer = 1;

       HashMap<String, Integer> map = new HashMap<>();

       for(int i = 0; i < clothes.length; i++) {
              if(map.get(clothes[i][1]) == null)
                     map.put(clothes[i][1], 1);
              else
                     map.put(clothes[i][1], map.get(clothes[i][1])
+ 1);
       }

       for(String keys: map.keySet()) {
              answer *= (map.get(keys) + 1);
       }

       answer -= 1;

       return answer;
   }
}

https://2ssue.github.io/algorithm/programmers_42578/
```

4. 베스트앨범

문제 설명

스트리밍 사이트에서 장르 별로 가장 많이 재생된 노래를 두 개씩 모아 베스트 앨범을 출시하려 합니다. 노래는 고유 번호로 구분하며, 노래를 수록하는 기준은 다음과 같습니다.

1. 속한 노래가 많이 재생된 장르를 먼저 수록합니다.
2. 장르 내에서 많이 재생된 노래를 먼저 수록합니다.
3. 장르 내에서 재생 횟수가 같은 노래 중에서는 고유 번호가 낮은 노래를 먼저 수록합니다.

노래의 장르를 나타내는 문자열 배열 genres 와 노래별 재생 횟수를 나타내는 정수 배열 plays 가 주어질 때, 베스트 앨범에 들어갈 노래의 고유 번호를 순서대로 return 하도록 solution 함수를 완성하세요.

제한사항

- genres[i]는 고유번호가 i 인 노래의 장르입니다.
- plays[i]는 고유번호가 i 인 노래가 재생된 횟수입니다.
- genres 와 plays 의 길이는 같으며, 이는 1 이상 10,000 이하입니다.
- 장르 종류는 100 개 미만입니다.
- 장르에 속한 곡이 하나라면, 하나의 곡만 선택합니다.
- 모든 장르는 재생된 횟수가 다릅니다.

입출력 예

genres	plays	return
[classic, pop, classic, classic, pop]	[500, 600, 150, 800, 2500]	[4, 1, 3, 0]

입출력 예 설명

classic 장르는 1,450 회 재생되었으며, classic 노래는 다음과 같습니다.

- 고유 번호 3: 800 회 재생
- 고유 번호 0: 500 회 재생
- 고유 번호 2: 150 회 재생

pop 장르는 3,100 회 재생되었으며, pop 노래는 다음과 같습니다.

- 고유 번호 4: 2,500 회 재생
- 고유 번호 1: 600 회 재생

따라서 pop 장르의 [4, 1]번 노래를 먼저, classic 장르의 [3, 0]번 노래를 그다음에 수록합니다.

※ 공지 - 2019 년 2 월 28 일 테스트케이스가 추가되었습니다.

.

```
import java.util.*;class Solution {
   public int[] solution(String[] genres, int[] plays) {

       //고유번호 = key 값 & 장르, 플레이횟수 = value 값
       HashMap<Integer, Integer> pMap = new HashMap<Integer,
Integer>();
       HashMap<Integer, String> gMap = new HashMap<Integer,
String>();
       for(int i = 0; i < genres.length; i++){
          pMap.put(i, plays[i]);
          gMap.put(i, genres[i]);
       }

       //HashSet 으로 장르 분류
       HashSet<String> gSet = new HashSet<String>();

       for(int i = 0; i < genres.length; i++){
          gSet.add(genres[i]);
       }

       //장르별 총 플레이 횟수
       HashMap<Integer, String> coPlay = new HashMap<Integer,
String>();
       for(String x : gSet){
          int count = 0;
          for(int i = 0; i < gMap.size(); i++){
             if(gMap.get(i).equals(x)){
                count+= pMap.get(i);
             }
          }
          coPlay.put(count, x);
       }

       //플레이 횟수 별로 정렬(키 값을 기준으로 정렬, TreeMap)
       TreeMap sort = new TreeMap(coPlay);
       String[] sortGenre = new String[gSet.size()];
```

```
        int index = 0;
        for(Object o : sort.keySet()){
            sortGenre[index] = sort.get(o).toString();
            index++;
        }

        //장르별 많이 플레이 된 노래의 고유번호 찾기
        ArrayList<Integer> fIndex = new ArrayList<Integer>();
        for(int i = sortGenre.length-1; i >= 0; i--){
            int count = 0;
            for(int p1 : gMap.keySet()){
                if(sortGenre[i].equals(gMap.get(p1))){
                    count++;
                }
            }

            int[] temp = new int[count];
            int k = 0;
            for(int p2 : gMap.keySet()){
                if(sortGenre[i].equals(gMap.get(p2))){
                    temp[k] = pMap.get(p2);
                    k++;
                }
            }

            if(temp.length != 1){
                Arrays.sort(temp);for(int j = temp.length - 1; j >=
temp.length - 2; j--){
                    for(int p : pMap.keySet()){
                        if(temp[j] == pMap.get(p)){
                            fIndex.add(p);
                            pMap.put(p, 0);
                            break;
                        }
                    }
                }
            }else{
                for(int p : pMap.keySet()){
                    if(temp[0] == pMap.get(p)){
                        fIndex.add(p);
                        pMap.put(p, 0);
                        break;
                    }
                }
            }}

        int[] answer = new int[fIndex.size()];

        for(int i = 0; i < fIndex.size(); i++){
            answer[i] = fIndex.get(i);
```

```
        }

        return answer;
    }
}
```

https://medium.com/@nsh235482/java-coding-programmers-hash-lv3-%EB%B2%A0%EC%8A%A4%ED%8A%B8-%EC%95%A8%EB%B2%94-278fa3ad4d9c