| 문항번호 | | 12 | 난이도 | 중 |
| --- | --- | --- | --- | --- |
| 유형 | | 해시 | 예상 소요 시간 | 1시간 |
| 문제 | SNS에서 서로가 팔로우를 했다는 것을 '맞팔'이라고 표현합니다.  다음 팔로우 관계를 나타낸 배열을 통해 서로 맞팔 관계인 쌍의 수를 리턴하는 함수를 작성하세요.  이때 ["철수", "영희"] 라는 정보는 철수가 영희를 팔로우 했음을 나타냅니다.  입력  팔로우 관계가 표현된 String 2차원 배열 A.  출력  맞팔 관계인 쌍의 수 | | | |
| 매개변수  &  리턴타입 | 매개변수 : Stirng[][] A  리턴타입 : int | | | |
| 지원자  초기 코드 | * python   def solution(A):  answer = 0  return answer;   * javascript   function solution(A) {  var answer = 0;  return answer;  } | | | |
| 모범  답안 | * python   class FriendShip():  def \_\_init\_\_(self):  self.friendMap = dict()  self.fff = 0  def setFriendShip(self, friend):  try:  self.friendMap[friend[0]].add(friend[1])  except KeyError:  self.friendMap[friend[0]] = set([friend[1]])  if friend[1] not in self.friendMap or friend[0] not in self.friendMap[friend[1]]:  return  else:  self.fff += 1  return  def solution(friends):  fs = FriendShip()  for friend in friends:  fs.setFriendShip(friend)  return fs.fff   * javascript   function solution(friends)  {  var map = new Map();  var result = 0;  var target = [];  for (var i = 0; i < friends.length; i++)  {  target = map.get(friends[i][1]);  if (target == undefined || target.indexOf(friends[i][0]) == -1)  {  target = target == undefined ? [friends[i][1]] : target.push(friends[i][1]);  map.set(friends[i][0], target);  }  else  {  result ++;  }  }  return result;  } | | | |
| 테스트  케이스 | 입력 : [["철수", "영희"], ["영희", "진수"], ["진수", "동수"], ["진수", "진호"]]  출력 : 0  입력 : [["철수", "영희"], ["영희", "진수"], ["진수", "영희"], ["진수", "진호"]]  출력 : 1  입력 : [["철수", "영희"], ["영희", "철수"], ["진수", "진호"], ["진호", "진수"]]  출력 : 2  입력 : [["철수", "영희"]]  출력 : 0  입력 : [["철수", "영희"], ["철수", "진수"], ["철수", "동수"]]  출력 : 0 | | | |