1. 타겟 넘버

문제 설명

n 개의 음이 아닌 정수가 있습니다. 이 수를 적절히 더하거나 빼서 타겟 넘버를 만들려고 합니다. 예를 들어 [1, 1, 1, 1, 1]로 숫자 3 을 만들려면 다음 다섯 방법을 쓸 수 있습니다.

```
-1+1+1+1+1 = 3
+1-1+1+1+1 = 3
+1+1-1+1+1 = 3
+1+1+1-1+1 = 3
+1+1+1+1-1 = 3
```

사용할 수 있는 숫자가 담긴 배열 numbers, 타겟 넘버 target 이 매개변수로 주어질 때 숫자를 적절히 더하고 빼서 타겟 넘버를 만드는 방법의 수를 return 하도록 solution 함수를 작성해주세요.

제한사항

- 주어지는 숫자의 개수는 2 개 이상 20 개 이하입니다.
- 각 숫자는 1 이상 50 이하인 자연수입니다.
- 타겟 넘버는 1 이상 1000 이하인 자연수입니다.

입출력 예

numbers	target	return
[1, 1, 1, 1, 1]	3	5

입출력 예 설명

문제에 나온 예와 같습니다.

```
class Solution {
   public int solution(int[] numbers, int target) {
       int answer = 0;

       answer = bfs(numbers, target, numbers[0], 1) + bfs(numbers,
target, -numbers[0], 1);

       return answer;
   }

   public int bfs(int[] numbers, int target, int sum, int i) {

       if(i == numbers.length) {
          if(sum == target) {
```

```
            return 1;
         } else {
            return 0;
         }
      }

      int result = 0;
      result += bfs(numbers, target, sum+numbers[i], i+1);
      result += bfs(numbers, target, sum-numbers[i], i+1);
      return result;
   }
}
```

https://velog.io/@jaesika/%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%A8%B8%EC%8A%A4-DFS-%ED%83%80%EA%B2%9F%EB%84%98%EB%B2%84

2. 네트워크

문제 설명

네트워크란 컴퓨터 상호 간에 정보를 교환할 수 있도록 연결된 형태를 의미합니다. 예를 들어, 컴퓨터 A 와 컴퓨터 B 가 직접적으로 연결되어있고, 컴퓨터 B 와 컴퓨터 C 가 직접적으로 연결되어 있을 때 컴퓨터 A 와 컴퓨터 C 도 간접적으로 연결되어 정보를 교환할 수 있습니다. 따라서 컴퓨터 A, B, C 는 모두 같은 네트워크 상에 있다고 할 수 있습니다.

컴퓨터의 개수 n, 연결에 대한 정보가 담긴 2 차원 배열 computers 가 매개변수로 주어질 때, 네트워크의 개수를 return 하도록 solution 함수를 작성하시오.

제한사항

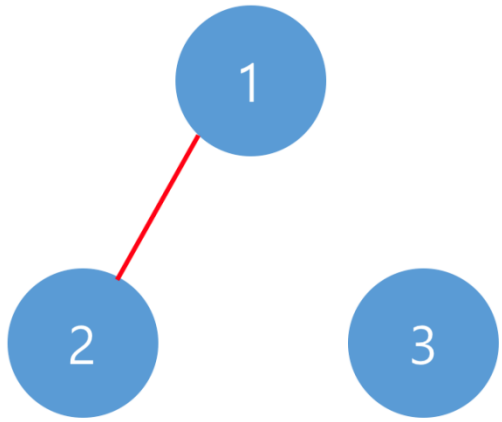
- 컴퓨터의 개수 n 은 1 이상 200 이하인 자연수입니다.
- 각 컴퓨터는 0 부터 n-1 인 정수로 표현합니다.
- i 번 컴퓨터와 j 번 컴퓨터가 연결되어 있으면 computers[i][j]를 1 로 표현합니다.
- computer[i][i]는 항상 1 입니다.

입출력 예

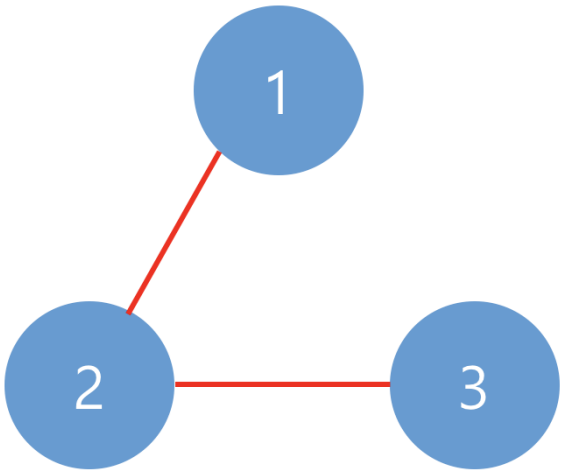
n	computers	return
3	[[1, 1, 0], [1, 1, 0], [0, 0, 1]]	2
3	[[1, 1, 0], [1, 1, 1], [0, 1, 1]]	1

입출력 예 설명

예제 #1
아래와 같이 2 개의 네트워크가 있습니다.



예제 #2
아래와 같이 1 개의 네트워크가 있습니다.




```
public class Solution {
  public int solution(int n, int[][] computers) {
    int answer = 0;
    boolean[] check = new boolean[n]; // n 갯수만큼 boolean 배열을
만들고 모든 요소를 false 로 초기화

    for (int i = 0; i < n; i++) {
      if (!check[i]) {
        dfs(computers, i, check);
        answer++;
      }
    }

    return answer;
  }

  boolean[] dfs(int[][] computers, int i, boolean[] check) {
    check[i] = true;

    for (int j = 0; j < computers.length; j++) {
      if (i != j && computers[i][j] == 1 && check[j] == false) {
        check = dfs(computers, j, check);
      }
    }
    return check;
  }

  @Test
  public void 정답() {
```

```
   Assert.assertEquals(2, solution(3, new int[][]{{1, 1, 0}, {1,
1, 0}, {0, 0, 1}}));
   Assert.assertEquals(1, solution(3, new int[][]{{1, 1, 0}, {1,
1, 1}, {0, 1, 1}}));
  }
}
```

https://velog.io/@ajufresh/%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%A8%B8%EC%8A%A4-%EB%84%A4%ED%8A%B8%EC%9B%8C%ED%81%AC-%EB%AC%B8%EC%A0%9C%ED%92%80%EC%9D%B4-Java

3. 단어 변환

문제 설명

두 개의 단어 begin, target 과 단어의 집합 words 가 있습니다. 아래와 같은 규칙을 이용하여 begin 에서 target 으로 변환하는 가장 짧은 변환 과정을 찾으려고 합니다.

1. 한 번에 한 개의 알파벳만 바꿀 수 있습니다.
2. words 에 있는 단어로만 변환할 수 있습니다.

예를 들어 begin 이 hit, target 가 cog, words 가 [hot,dot,dog,lot,log,cog]라면 hit -> hot -> dot -> dog -> cog 와 같이 4 단계를 거쳐 변환할 수 있습니다.

두 개의 단어 begin, target 과 단어의 집합 words 가 매개변수로 주어질 때, 최소 몇 단계의 과정을 거쳐 begin 을 target 으로 변환할 수 있는지 return 하도록 solution 함수를 작성해주세요.

제한사항

- 각 단어는 알파벳 소문자로만 이루어져 있습니다.
- 각 단어의 길이는 3 이상 10 이하이며 모든 단어의 길이는 같습니다.
- words 에는 3 개 이상 50 개 이하의 단어가 있으며 중복되는 단어는 없습니다.
- begin 과 target 은 같지 않습니다.
- 변환할 수 없는 경우에는 0 를 return 합니다.

입출력 예

begin	target	words	return
hit	cog	[hot, dot, dog, lot, log, cog]	4
hit	cog	[hot, dot, dog, lot, log]	0

입출력 예 설명

예제 #1
문제에 나온 예와 같습니다.

예제 #2
target 인 cog 는 words 안에 없기 때문에 변환할 수 없습니다.

```
// https://programmers.co.kr/learn/courses/30/lessons/43163

import java.util.Arrays;

class Solution {
   public int solution(String begin, String target, String[]
words) {
```

```
        int minimum = words.length + 1;

        minimum = dfs(begin, target, words, new
boolean[words.length], 0, words.length + 1, words.length);

        return minimum == words.length + 1? 0 : minimum;
    }

    int dfs(String word, String target, String[] words, boolean[]
visited, int n, int minimum, int maximum) {
        for (int i = 0; i < maximum; i++) {
            if ( !visited[i] && conversion(word, words[i]) ) {
                //System.out.println(n + " " + minimum + " " + word
+ " " + words[i]);

                if ( words[i].equals(target) ) {
                    return Math.min(minimum, n + 1);
                }

                visited[i] = true;
                int num = dfs(words[i], target, words, visited, n +
1, minimum, maximum);
                if ( num < minimum ) minimum = num;
                visited[i] = true;
            }
        }

        return minimum;
    }

    boolean conversion(String w1, String w2) {
        int tmp = 0;

        for (int i = 0; i < w1.length(); i++) {
            if ( w1.charAt(i) != w2.charAt(i) ) {
                tmp++;
                if ( tmp > 1 ) return false;
            }
        }

        return true;
    }
}
```

https://mapled.tistory.com/entry/%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%A8%B8%EC%8A%A43%EB%8B%A8%EA%B3%84%EC%9E%90%EB%B0%94-%EB%8B%A8%EC%96%B4-%EB%B3%80%ED%99%98

4. 여행경로

문제 설명

주어진 항공권을 모두 이용하여 여행경로를 짜려고 합니다. 항상 ICN 공항에서 출발합니다.

항공권 정보가 담긴 2 차원 배열 tickets 가 매개변수로 주어질 때, 방문하는 공항 경로를 배열에 담아 return 하도록 solution 함수를 작성해주세요.

제한사항

- 모든 공항은 알파벳 대문자 3 글자로 이루어집니다.
- 주어진 공항 수는 3 개 이상 10,000 개 이하입니다.
- tickets 의 각 행 [a, b]는 a 공항에서 b 공항으로 가는 항공권이 있다는 의미입니다.
- 주어진 항공권은 모두 사용해야 합니다.
- 만일 가능한 경로가 2 개 이상일 경우 알파벳 순서가 앞서는 경로를 return 합니다.
- 모든 도시를 방문할 수 없는 경우는 주어지지 않습니다.

입출력 예

tickets	return
[[ICN, JFK], [HND, IAD], [JFK, HND]]	[ICN, JFK, HND, IAD]
[[ICN, SFO], [ICN, ATL], [SFO, ATL], [ATL, ICN], [ATL,SFO]]	[ICN, ATL, ICN, SFO, ATL, SFO]

입출력 예 설명

예제 #1

[ICN, JFK, HND, IAD] 순으로 방문할 수 있습니다.

예제 #2

[ICN, SFO, ATL, ICN, ATL, SFO] 순으로 방문할 수도 있지만
[ICN, ATL, ICN, SFO, ATL, SFO] 가 알파벳 순으로 앞섭니다.

```
import java.util.ArrayList;
import java.util.Collections;
import java.util.List;
class Solution {
   List<String> list = new ArrayList<>();
       static String route = "";
       static boolean[] visit;

       private void dfs(String[][] tickets, String end, int cnt)
{
              route += end + ",";

              if(cnt == tickets.length) {
                     list.add(route); return;
              }

              for(int i = 0; i < tickets.length; i++) {
                     String s = tickets[i][0], e = tickets[i][1];
                     if(s.equals(end) && !visit[i]) {
                            visit[i] = true;
                            dfs(tickets, e, cnt + 1);
                            visit[i] = false; route =
route.substring(0, route.length()-4);
                     }
              }
       }

       public String[] solution(String[][] tickets) {
              for(int i = 0; i < tickets.length; i++) {
                     visit = new boolean[tickets.length];
                     String start = tickets[i][0], end =
tickets[i][1];

                     if(start.equals("ICN")) {
                            route = start + ","; visit[i] = true;
                            dfs(tickets, end, 1);
                     }
              }

              Collections.sort(list);
              String[] answer = list.get(0).split(",");

              return answer;
       }
}
```

https://geehye.github.io/programmers-dfs-bfs-04/#