구현 (1)

23.12.11 송이

문제 57 - level3

문제 57

문제 이해하기

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"]

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

제한사항

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

문제 57
접근방법

ICN -> JFK

HND ->IAD

1

공항들간의 연결을
그래프로 만들자

2 그래프를 대상으로 DFS를 수행하자.

0 다만 가능한 경로가 2가지 이상일 경우 알파벳
순서가 앞서는 경로를 반환해야 하므로
먼저 정렬을 한 후 그래프를 만들자.

문제 57

그래프를 정의하고 데이터를 넣기

```
from collections import defaultdict
def solution(tickets):
    graph = defaultdict(list)

    for a,b in tickets : graph[a].append(b)
    for key in graph.keys():graph[key].sort()

    print(graph)
```

```
defaultdict(<class 'list'>, {'ICN': ['SFO', 'ATL'], 'SFO': ['ATL'], 'ATL':
['ICN', 'SFO']})
```

```
defaultdict(<class 'list'>, {'ICN': ['ATL', 'SFO'], 'SFO': ['ATL'], 'ATL':
['ICN', 'SFO']})
```

defaultdict vs dict

차이점은?

프로그래머스 > test.py > ...

```
1   from collections import defaultdict
2
3   int_dict = defaultdict(int)
4   print(int_dict)
```

```
defaultdict(<class 'int'>, {})
```

```
defaultdict(<class 'int'>, {})
defaultdict(<class 'int'>, {'key1': 0})
```

```
from collections import defaultdict

list_dict = defaultdict(list)
print(list_dict)

list_dict['key1']
list_dict['key2'] = 'test'

print(list_dict)
```

```
defaultdict(<class 'list'>, {})
defaultdict(<class 'list'>, {'key1': [], 'key2': 'test'}
)
```

문제 57

탐색 함수 만들기

```
def dfs(graph,path,visit):
    if path:
        to = path[-1]
        if graph[to]: path.append(graph[to].pop(0))
        else: visit.append(path.pop())
        dfs(graph,path,visit)
    return visit[::-1]
```

방문한 공항은 아예 pop 해서 제거하기

```
['SFO', 'ATL', 'SFO', 'ICN', 'ATL', 'ICN']
```

```
return dfs(graph,['ICN'],[])
```

우리가 초기에 세팅한 그래프 자체

이동경로

방문한 공항들 배열

문제에서 ICN에서 시작한다고 명시함

```
to ICN
path :  ['ICN', 'ATL']
to ATL
path :  ['ICN', 'ATL', 'ICN']
to ICN
path :  ['ICN', 'ATL', 'ICN', 'SFO']
to SFO
path :  ['ICN', 'ATL', 'ICN', 'SFO', 'ATL']
to ATL
path :  ['ICN', 'ATL', 'ICN', 'SFO', 'ATL', 'SFO']
to SFO
visit :  ['SFO']
to ATL
visit :  ['SFO', 'ATL']
to SFO
visit :  ['SFO', 'ATL', 'SFO']
```

```
defaultdict(<class 'list'>, {'ICN': ['ATL', 'SFO'], 'SFO': ['ATL'], 'ATL':
['ICN', 'SFO']})
```

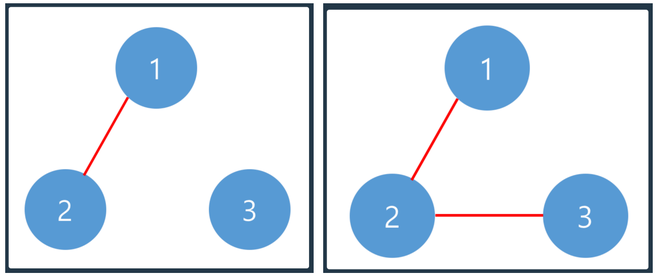
문제 58 - level3

문제 58

문제 이해하기

네트워크란 컴퓨터 상호 간에 정보를 교환할 수 있도록 연결된 형태를 의미합니다. 예를 들어, 컴퓨터 A와 컴퓨터 B가 직접적으로 연결되어있고, 컴퓨터 B와 컴퓨터 C가 직접적으로 연결되어 있을 때 컴퓨터 A와 컴퓨터 C도 간접적으로 연결되어 정보를 교환할 수 있습니다. 따라서 컴퓨터 A, B, C는 모두 같은 네트워크 상에 있다고 할 수 있습니다.

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



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

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

문제 58

접근방법

그래프에서 자주 등장하는 네트워크 문제

유니온 파인드를 이용해서 각 노드의 대표노드를 찾아 대표 노드가 같다면 같은 네트워크 안에 소속되어 있다고 판단할 수 있음

하지만 이번 문제에서는 이미 어떤 노드끼리 연결되어 있는지 친절하게 알려줬기 때문에 DFS를 사용해서 간단한 방법으로 풀 예정

보통의 DFS에서 방문처리를 하는 것은 이미 방문한 노드를 다시 방문하지 않으려는 목적이 크지만
이번 문제에서는 방문처리를 해주어 함수를 실행하고도 모든 노드를 방문하지 않았으면 그 노드는 하나의 네트워크로 구성된 컴퓨터가 아니라는 뜻이 된다.

탐색을 수행 할 준비를 하기

DFS 함수를 구현하기

문제 58

탐색 준비

```
def solution(n, computers):
    visited = [0] * n
    answer = 0

    for i in range(n):
        if visited[i]==0:
            #DFS 함수 실행
            answer +=1

    return answer
```

방문했는지 알 수 있는 배열 , 0으로 초기화

정답의 개수 , 처음엔 0으로 초기화

visited 배열을 순회하면서
dfs 함수 실행

조건에 맞다면 정답을 +1 해주기

answer 리턴

문제 58

DFS 함수 작성

```
dfs(i,computers,visited)
```

```
def dfs(k,graph,visited):
    visited[k] = 1
    for i in range(len(graph[k])):
        if visited[i] == 0 and graph[k][i] == 1:
            dfs(i,graph,visited)
```

k번째 노드는 방문했다고 처리

연결되어 있다면

i로 다시 탐색을 시작한다.

dfs(1,graph,visited)

computers
[[1, 1, 0], [1, 1, 0], [0, 0, 1]]

인덱스 0 인덱스 1 인덱스 2

0번 컴퓨터 0번 컴퓨터 1

1번 컴퓨터 1

2번 컴퓨터 0

문제 58

DFS 함수 작성

```
def solution(n, computers):
    visited = [0] * n
    answer = 0

    for i in range(n):
        if visited[i]==0:
            dfs(i,computers,visited)
            answer +=1

    return answer
```

dfs 함수를 수행하고도
모든 노드가 방문처리 되어 있지 않다면
그 노드는 하나의 네트워크로 구성된 컴퓨터가 아니라는 뜻이 됩니다.

즉 아직 방문하지 않은 0인 노드가 있다면 정답값을 1 증가시켜 주면 됩니다.