백준 16940 BFS 스페셜저지

202020988 조아영

001 문제 소개

002 아이디어

003 코드 설명

문제 소개

문제

BOJ에서 정답이 여러가지인 경우에는 스페셜 저지를 사용한다. 스페셜 저지는 유저가 출력한 답을 검증하는 코드를 통해서 정답 유무를 결정하는 방식이다. 오늘은 스페셜 저지 코드를 하나 만들어보려고 한다.

정점의 개수가 N이고, 정점에 1부터 N까지 번호가 매겨져있는 양방향 그래프가 있을 때, BFS 알고리즘은 다음과 같은 형태로 이루어져 있다.

1. 큐에 시작 정점을 넣는다. 이 문제에서 시작 정점은 1이다. 1을 방문했다고 처리한다.
2. 큐가 비어 있지 않은 동안 다음을 반복한다.
    1. 큐에 들어있는 첫 정점을 큐에서 꺼낸다. 이 정점을 x라고 하자.
    2. x와 연결되어 있으면, 아직 방문하지 않은 정점 y를 모두 큐에 넣는다. 모든 y를 방문했다고 처리한다.

2-2 단계에서 방문하지 않은 정점을 방문하는 순서는 중요하지 않다. 따라서, BFS의 결과는 여러가지가 나올 수 있다.

트리가 주어졌을 때, 올바른 BFS 방문 순서인지 구해보자.

입력

첫째 줄에 정점의 수 N($2 \le N \le 100,000$)이 주어진다. 둘째 줄부터 N-1개의 줄에는 트리의 간선 정보가 주어진다. 마지막 줄에는 BFS 방문 순서가 주어진다. BFS 방문 순서는 항상 N개의 정수로 이루어져 있으며, 1부터 N까지 자연수가 한 번씩 등장한다.

출력

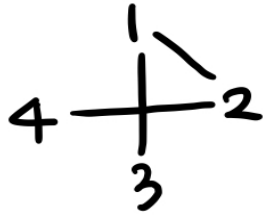
입력으로 주어진 BFS 방문 순서가 올바른 순서면 1, 아니면 0을 출력한다.

예시

70%에서 틀림

① 그래프 연결정보 저장 (graph)
② 입력된 방문순서에 따른 배열생성 (index) → ③ graph 정렬

입력 N=4
1 2
1 3
2 4
) 간선정보 →



가능한 정답
1 2 3 4
1 3 2 4 예시!
:

↳ graph = [[], [2,3], [1,4], [1], [2]]

① 3 2 4 ⇒ index 배열 = [0, 0, 2, 1, 3]
index 0 1 2 3

인덱스값이 작은것이 먼저 나와야함!

⇒ graph 정렬! [[], [3,2], [1,4], [1], [2]]

코드 설명

```python
import sys
from collections import deque

N = int(sys.stdin.readline())
graph = [[] for _ in range(N+1)]
visited = [False for _ in range(N)]

for _ in range(N-1):
    a, b = map(int, sys.stdin.readline().split())
    graph[a].append(b)
    graph[b].append(a)

order = list(map(int, sys.stdin.readline().split()))
index = [0 for _ in range(N+1)]

for i in range(len(order)):
    index[order[i]] = i

for i in range(1, len(graph)):
    graph[i].sort(key = lambda x:index[x])
```

정점의 개수 입력

양방향 그래프의 연결 정보 저장

BFS 순서 입력

입력 받은 방문 순서의 인덱스 값을 저장

Index 배열을 기준으로 그래프의 인접리스트 정렬

코드 설명

```
22 def bfs(v):
23     q = deque()
24     visited = [False] * (N+1)
25     result = [v]
26     q.append(v)
27     visited[v] = True
28     while q:
29         x = q.popleft()
30         for y in graph[x]:
31             if visited[y] == False:
32                 q.append(y)
33                 visited[y]=True
34                 result.append(y)
35     return result
36
37 bfs(order[0])
38 if order == bfs(order[0]):
39     print(1)
40 else:
41     print(0)
```

Queue를 이용한 BFS 탐색

방문순서를 result에 저장

입력받은 방문순서와 bfs함수로 방문한 순서가 같은지 비교

감사합니다