# Algorithm 4월 1주차

BFS

# Table of content

파괴되지 않은 건물

# [프로그래머스] 부대복귀

- 전체 노드 개수 n
- 출발지 번호 배열 sources
- 목적지 번호 destination
- 연결 간선 표시 배열 roads
- 각 source 에서 destination까지 도착하는 데 걸리는 시간을 반환

**문제 설명**

강철부대의 각 부대원이 여러 지역에 뿔뿔이 흩어져 특수 임무를 수행 중입니다. 지도에서 강철부대가 위치한 지역을 포함한 각 지역은 유일한 번호로 구분되며, 두 지역 간의 길을 통과하는 데 걸리는 시간은 모두 1로 동일합니다. 임무를 수행한 각 부대원은 지도 정보를 이용하여 최단시간에 부대로 복귀하고자 합니다. 다만 적군의 방해로 인해, 임무의 시작 때와 다르게 되돌아오는 경로가 없어져 복귀가 불가능한 부대원도 있을 수 있습니다.

강철부대가 위치한 지역을 포함한 총지역의 수 `n` , 두 지역을 왕복할 수 있는 길 정보를 담은 2차원 정수 배열 `roads` , 각 부대원이 위치한 서로 다른 지역들을 나타내는 정수 배열 `sources` , 강철부대의 지역 `destination` 이 주어졌을 때, 주어진 `sources` 의 원소 순서대로 강철부대로 복귀할 수 있는 최단시간을 담은 배열을 return하는 solution 함수를 완성해주세요. 복귀가 불가능한 경우 해당 부대원의 최단시간은 -1입니다.

| n | roads | sources | destination | result |
|---|---|---|---|---|
| 3 | [[1, 2], [2, 3]] | [2, 3] | 1 | [1, 2] |
| 5 | [[1, 2], [1, 4], [2, 4], [2, 5], [4, 5]] | [1, 3, 5] | 5 | [2, -1, 0] |

# [프로그래머스] 풀이법 - BFS

- 일단 최단 거리 소요 시간을 묻는 문제이므로 다익스트라 아니면 BFS로 푼다.
- 가중치가 없으므로 BFS로 풀면 됨 -> queue를 사용한다.
- Destination부터 출발해서 각 노드까지 도착하는 시간을 기록한다. (distance 배열에 기록)
- Source 부터 출발하여 기록하면 너무 많은 루트가 반복됨

```python
from collections import defaultdict, deque
def solution(n, roads, sources, destination):
    answer = []
    distance = [-1] * (n + 1)
    links = defaultdict(list)
    for i in range(len(roads)):
        links[roads[i][0]].append(roads[i][1])
        links[roads[i][1]].append(roads[i][0])
    q = deque()
    distance[destination] = 0
    q.append(destination)
    while q:
        now = q.popleft()
        for node in links[now]:
            if distance[node] == -1:
                distance[node] = distance[now] + 1
                q.append(node)
    for s in sources:
        answer.append(distance[s])
    return answer
```