# 미로 찾기 프로그램(그래프 탐색)  
# 입력: 미로 정보 g, 출발점 start, 도착점 end  
# 출력: 미로를 나가기 위한 이동 경로는 문자열, 나갈 수 없는 미로면 물음표("?")  
  
def solve\_maze(g, start, end):  
 qu = [] # 기억 장소 1: 앞으로 처리해야 할 이동 경로를 큐에 저장  
 done = set() # 기억 장소 2: 이미 큐에 추가한 꼭짓점들을 집합에 기록(중복 방지)  
  
 qu.append(start) # 출발점을 큐에 넣고 시작  
 done.add(start) # 집합에도 추가  
  
 while qu: # 큐에 처리할 경로가 남아 있으면  
 p = qu.pop(0) # 큐에서 처리 대상을 꺼냄  
 v = p[-1] # 큐에 저장된 이동 경로의 마지막 문자가 현재 처리해야 할 꼭짓점  
 if v == end: # 처리해야 할 꼭짓점이 도착점이면(목적지 도착!)  
 return p # 지금까지의 전체 이동 경로를 돌려주고 종료  
 for x in g[v]: # 대상 꼭짓점에 연결된 꼭짓점들 중에  
 if x not in done: # 아직 큐에 추가된 적이 없는 꼭짓점을  
 qu.append(p + x) # 이동 경로에 새 꼭짓점으로 추가하여 큐에 저장하고  
 done.add(x) # 집합에도 추가함  
  
 # 탐색을 마칠 때까지 도착점이 나오지 않으면 나갈 수 없는 미로임  
 return "?"  
  
  
# 미로 정보  
# 미로의 각 위치에 알파벳으로 이름을 지정  
# 각 위치에서 한 번에 이동할 수 있는 모든 위치를 선으로 연결하여 그래프로 표현  
maze = {  
 'a': ['e'],  
 'b': ['c', 'f'],  
 'c': ['b', 'd'],  
 'd': ['c'],  
 'e': ['a', 'i'],  
 'f': ['b', 'g', 'j'],  
 'g': ['f', 'h'],  
 'h': ['g', 'l'],  
 'i': ['e', 'm'],  
 'j': ['f', 'k', 'n'],  
 'k': ['j', 'o'],  
 'l': ['h', 'p'],  
 'm': ['i', 'n'],  
 'n': ['m', 'j'],  
 'o': ['k'],  
 'p': ['l']  
}  
  
found\_miro = solve\_maze(maze, 'a', 'p')  
found\_miro = '->'.join(found\_miro)  
  
print('찾은 미로 경로는', found\_miro)