```
#djikstra
from queue import PriorityQueue

n, s, f = map(int, input().split())
graph = dict()
dist = dict()
prev = dict()
q = PriorityQueue()

for i in range(1, n + 1):
    line = list(map(int, input().split()))
    graph[i] = list()
    for j in range(1, n + 1):
        if line[j - 1] >= 0:
            graph[i].append((j, line[j - 1]))
            dist[j] = float('inf')

q.put((0, s))
dist[s] = 0
while not q.empty():
    elem = q.get()
    v = elem[1]
    cur_d = elem[0]
    if cur_d > dist[v]:
        continue
    for i in range(len(graph[v])):
        to = graph[v][i][0]
        L = graph[v][i][1]
        if dist[v] + L < dist[to]:
            dist[to] = dist[v] + L
            prev[to] = v
            q.put((dist[to], to))

if dist[f] != float('inf'):
    line = list()
    node = f
    while node != s:
        line.append(node)
        node = prev[node]
    line.append(s)
    print(*line[::-1])
else:
    print(-1)
```

```
# dfs
n, m = map(int, input().split())
graph = dict()
used = [False] * (n + 1)
for i in range(m):
    f, s = map(int, input().split())
    if f not in graph:
        graph[f] = list()
    if s not in graph:
        graph[s] = list()
    graph[f].append(s)
    graph[s].append(f)
```

```
def dfs(v, p):
    used[v] = True
    if v not in graph:
        return
    for u in graph[v]:
        if not used[u]:
            dfs(u, v)
        elif u != p:
            return 'NO'
    return 'YES'


if m == n - 1 and graph.keys():
    print(dfs(1, -1))
else:
    print('NO')





#bfs
from queue import Queue

n = int(input())
visited = list()
graph = dict()
q = Queue()
adj = [0] * n


def bfs(start, end):
    if start == end:
        return 0
    q.put(start)
    while not q.empty():
        v = q.get()
        for u in graph[v]:
            if u != end and u not in visited:
                adj[u - 1] = v
                visited.append(v)
                q.put(u)
            elif u in visited:
                continue
            else:
                adj[u - 1] = v
                return 1
    return -1


def counting(start, end):
    a = end
    visited.clear()
    while a != start:
        visited.append(a)
        a = adj[a - 1]
    visited.append(a)
```

```
for i in range(1, n + 1):
    graph[i] = list()
    line = input()
    for num, j in enumerate(line.split(), 1):
        if j == '1':
            graph[i].append(num)
start, end = list(map(int, input().split()))

count = bfs(start, end)
if count < 1:
    print(count)
else:
    counting(start, end)
    print(len(visited) - 1)
    print(*reversed(visited))
```