**DFS**

```
# Реализация
def dfs(used, vertex, v):
    global count
    count += 1
    used[v] = True
    for elem in vertex[v]:
        if not used[elem]:
            dfs(used, vertex, elem)

# dfs с восстановлением пути
def dfs(graph, u, visited, previous):
    visited[u] = True
    for v in graph[u]:
        if not visited[v]:
            previous[v] = u
            dfs(graph, v, visited, previous)
    return previous

def get_path(previous, u):
    path = []
    while previous[u] is not None:
        path.append(u)
        u = previous[u]
    path.reverse()
    return path

visited = [False] * len(graph)
previous = [None] * len(graph)
prev = dfs(graph, 0, visited, previous)
print(get_path(prev, 4))
print(get_path(prev, 2))


# Количество компонент связности
n, m = map(int, input().split())
used = [False] * n
```

```
vertex = [[] for i in range(n)]
count = 0
for i in range(n):
    v = list(map(int, input().split()))
    for j in range(n):
        if v[j]:
            vertex[i].append(j)
dfs(used, vertex, m - 1)
print(count)

# Проверка на наличие циклов (неориентированный случай)
def dfs(graph, u, visited, prev=None):
    visited[u] = True
    for v in graph[u]:
        if v != prev and visited[v]:
            return True
        else:
            if dfs(graph, v, visited, u):
                return True
    return False

visited = [False] * len(graph)

# Проверка на наличие циклов (ориентированный случай)
WHITE, GRAY, BLACK = 0, 1, 2
def dfs(graph, u, state):
    state[u] = GRAY
    is_cyclic = False
    for v in graph[u]:
        if state[v] == GRAY:
            is_cyclic = True
            break
        elif state[v] == WHITE:
            dfs(graph, v, state)
    state[u] = BLACK
    return is_cyclic
```

```
state = [WHITE] * len(graph)


def dfs(graph, u, visited, colors, color):
    visited[u] = True
    colors[u] = color
    for v in graph[u]:
        if not visited[v]:
            dfs(graph, v, visited, colors, color ^ 1)
        else:
            if colors[v] == color:
                return False
    return True

n = int(input())
visited = [False] * n
colors = [-1] * n
dfs(graph, v, visited, colors, 0)
```