Chapter. 09

그래프 기본 탐색 알고리즘

핵심 유형 문제풀이

FAST CAMPUS
ONLINE
유형별 문제풀이

강사. 나동빈

Chapter. 09

그래프 기본 탐색 알고리즘(핵심 유형 문제풀이)

FAST CAMPUS
ONLINE
나동빈 강사.

바이러스: https://www.acmicpc.net/problem/2606

혼자 힘으로 풀어 보기

문제 제목: 바이러스

문제 난이도: 하(Easy)

문제 유형: DFS, BFS

추천 풀이 시간: 30분

바이러스: https://www.acmicpc.net/problem/2606

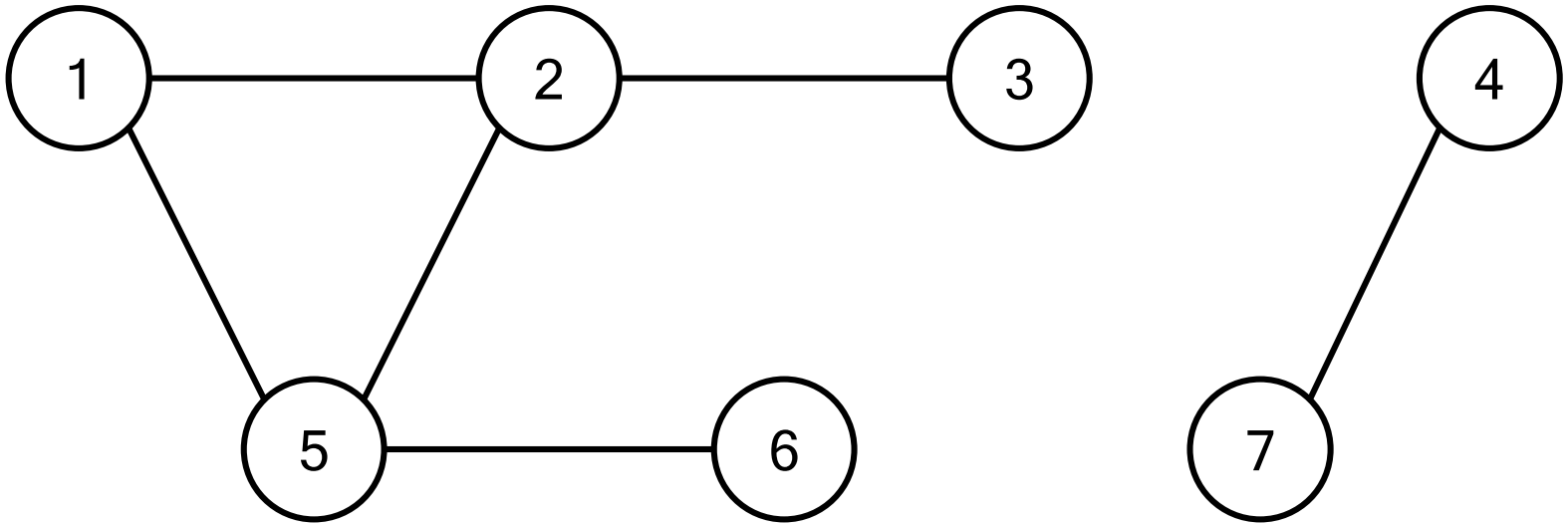
문제 풀이 핵심 아이디어

- 단순히 시작 정점에서부터 도달할 수 있는 노드의 수를 계산하는 문제입니다.
- 따라서 DFS 혹은 BFS를 이용하여 방문하게 되는 노드의 개수를 계산하면 됩니다.
- 컴퓨터의 수가 적으므로, DFS를 이용해 빠르게 문제를 푸는 것이 유리합니다.

바이러스: https://www.acmicpc.net/problem/2606

문제 풀이 핵심 아이디어

- 예시) 정점 개수 = 7, 간선 개수 = 6, 시작 정점 번호 = 1



- DFS: 1 → 2 → 3 → 5 → 6

바이러스: https://www.acmicpc.net/problem/2606

소스코드

```
n = int(input())
m = int(input())
adj = [[] for _ in range(n + 1)]
visited = [False] * (n + 1)
count = 0

for _ in range(m):
    x, y = map(int, input().split())
    adj[x].append(y)
    adj[y].append(x)

def dfs(now_pos):
    global count
    count += 1
    visited[now_pos] = True
    for next_pos in adj[now_pos]:
        if not visited[next_pos]:
            dfs(next_pos)

dfs(1)
print(count - 1)
```

혼자 힘으로 풀어 보기

유기농 배추: https://www.acmicpc.net/problem/1012

문제 제목: 유기농 배추

문제 난이도: 하(Easy)

문제 유형: DFS, BFS

추천 풀이 시간: 30분

유기농 배추: https://www.acmicpc.net/problem/1012

문제 풀이 핵심 아이디어

- 연결 요소의 개수를 세는 문제입니다.
- 모든 정점에 대하여 DFS 및 BFS를 수행하고, 한 번 방문한 정점은 다시 확인하지 않습니다.
- 전체적으로 **DFS 및 BFS를 수행한 총 횟수를 계산**합니다.
- DFS 및 BFS 응용 문제 중에서 출제 비중이 매우 높은 유형 중 하나입니다.
- DFS로 문제를 푸는 경우, sys 라이브러리의 setrecursionlimit() 함수 설정을 해줄 필요가 있습니다.

유기농 배추: https://www.acmicpc.net/problem/1012

문제 풀이 핵심 아이디어

- 예시를 확인해 봅시다.

1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1	1	1
0	0	0	0	1	0	0	1	1	1

유기농 배추: https://www.acmicpc.net/problem/1012

문제 풀이 핵심 아이디어

- 예시를 확인해 봅시다.

1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1	1	1
0	0	0	0	1	0	0	1	1	1

유기농 배추: https://www.acmicpc.net/problem/1012

문제 풀이 핵심 아이디어

- 예시를 확인해 봅시다.

1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1	1	1
0	0	0	0	1	0	0	1	1	1

유기농 배추: https://www.acmicpc.net/problem/1012

문제 풀이 핵심 아이디어

- 예시를 확인해 봅시다.

1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1	1	1
0	0	0	0	1	0	0	1	1	1

유기농 배추: https://www.acmicpc.net/problem/1012

문제 풀이 핵심 아이디어

- 예시를 확인해 봅시다.

1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1	1	1
0	0	0	0	1	0	0	1	1	1

유기농 배추: https://www.acmicpc.net/problem/1012

문제 풀이 핵심 아이디어

- 예시를 확인해 봅시다.

1	1	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0
0	0	1	1	0	0	0	1	1	1
0	0	0	0	1	0	0	1	1	1

  - 연결 요소의 개수: 5

유기농 배추: https://www.acmicpc.net/problem/1012

소스코드

```python
import sys
sys.setrecursionlimit(100000)

def dfs(x, y):
    visited[x][y] = True
    directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]
    for dx, dy in directions:
        nx, ny = x + dx, y + dy
        if nx < 0 or nx >= n or ny < 0 or ny >= m:
            continue
        if array[nx][ny] and not visited[nx][ny]:
            dfs(nx, ny)
```

```python
for _ in range(int(input())):
    m, n, k = map(int, input().split())
    array = [[0] * m for _ in range(n)]
    visited = [[False] * m for _ in range(n)]
    for _ in range(k):
        y, x = map(int, input().split())
        array[x][y] = 1
    result = 0
    for i in range(n):
        for j in range(m):
            if array[i][j] and not visited[i][j]:
                dfs(i, j)
                result += 1
    print(result)
```

혼자 힘으로 풀어 보기

효율적인 해킹: https://www.acmicpc.net/problem/1325

문제 제목: 효율적인 해킹

문제 난이도: 하(Easy)

문제 유형: DFS, BFS

추천 풀이 시간: 30분

효율적인 해킹: https://www.acmicpc.net/problem/1325

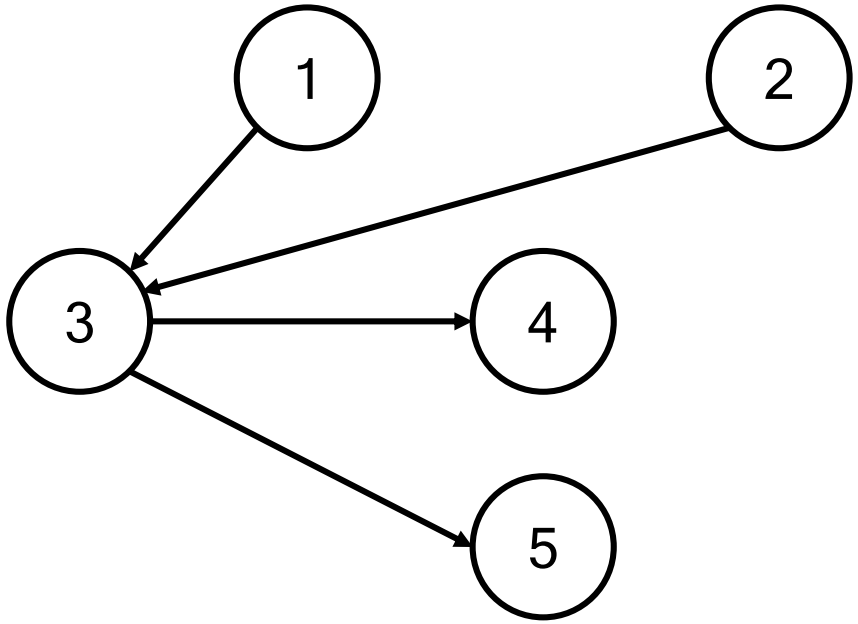
문제 풀이 핵심 아이디어

- 모든 정점에 대하여 DFS 및 BFS를 수행합니다.
- **DFS 혹은 BFS를 수행할 때마다 방문하게 되는 노드의 개수를 계산**하면 됩니다.
- 가장 노드의 개수가 크게 되는 시작 정점을 출력합니다.

효율적인 해킹: https://www.acmicpc.net/problem/1325

문제 풀이 핵심 아이디어

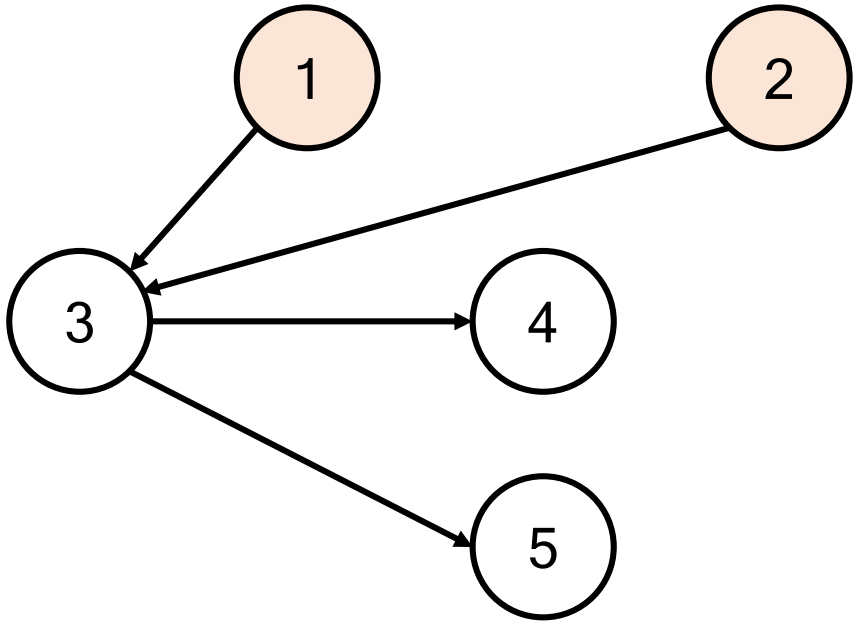
- 예시) 정점 개수 = 5, 간선 개수 = 4



효율적인 해킹: https://www.acmicpc.net/problem/1325

I 문제 풀이 핵심 아이디어

- 예시) 정점 개수 = 5, 간선 개수 = 4



- DFS: 1 → 3 → 4 → 5
- DFS: 2 → 3 → 4 → 5

효율적인 해킹: https://www.acmicpc.net/problem/1325

소스코드

```
from collections import deque

n, m = map(int, input().split())
adj = [[] for _ in range(n + 1)]

for _ in range(m):
    x, y = map(int, input().split())
    adj[y].append(x)

def bfs(v):
    q = deque([v])
    visited = [False] * (n + 1)
    visited[v] = True
    count = 1
    while q:
        v = q.popleft()
        for e in adj[v]:
            if not visited[e]:
                q.append(e)
                visited[e] = True
                count += 1
    return count

result = []
max_value = -1

for i in range(1, n + 1):
    c = bfs(i)
    if c > max_value:
        result = [i]
        max_value = c
    elif c == max_value:
        result.append(i)
        max_value = c

for e in result:
    print(e, end=" ")
```