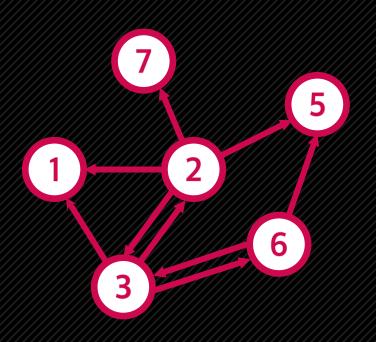


알고리즘 특강 그래프 탐색

코테 단골손님 중 하나인 그래프입니다. DFS/BFS는 정말 능숙하게 사용할 줄 알아야 합니다!

그래프 탐색





Breadth First Search

- 너비 우선 탐색이라고도 하며, 자신의 자식들부터 순차적으로 탐색.
- 순차 탐색 이후 다른 자식 노드의 자식을 확인하기 위해 큐를 사용.

Depth First Search

- 깊이 우선 탐색이라고도 하며, 최대한 깊게 탐색 후 빠져 나옴.
- 백트래킹의 일종이며, 재귀를 활용함.

유형을 하나씩 알아보자!





/<> Silver 3 - 바이러스 (#2606)

요약

- 한 컴퓨터가 웜 바이러스에 걸리면 네트워크에 연결된 모든 컴퓨터가 바이러스에 걸린다.
- 컴퓨터 수와 네트워크 상에 서로 연결된 정보가 주어질 때, 1번을 통해 감염된 컴퓨터의 수를 구하는 프로그램을 작성하시오.

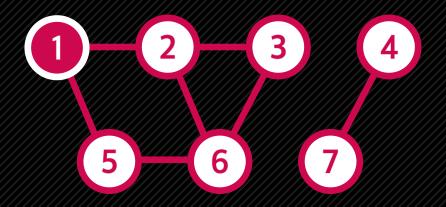
제약조건

컴퓨터의 수는 100 이하이다.





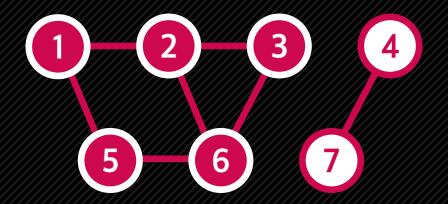
/◇ Silver 3 - 바이러스 (#2606)







/ Silver 3 - 바이러스 (#2606)







요약

- 집이 있는 곳을 1, 집이 없는 곳을 0으로 하는 그래프가 주어진다.
- 상하좌우에 다른 집이 있는 경우, 그 집은 붙어있다고 한다.
- 이때, 붙어있는 단지의 수와 각 단지 마다 속한 집의 수를 출력하는 프로그램을 작성하시오.

제약조건

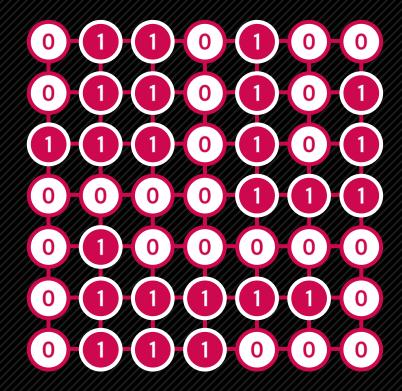
• 그래프의 가로/세로 크기는 25 이하이다.





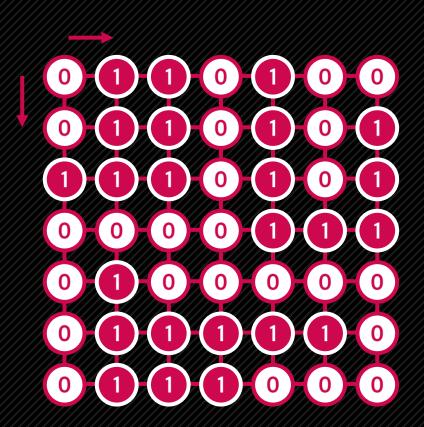






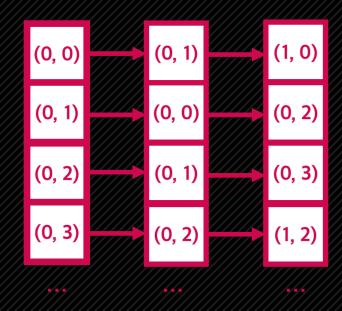


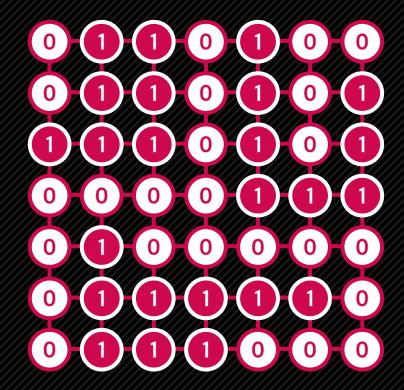










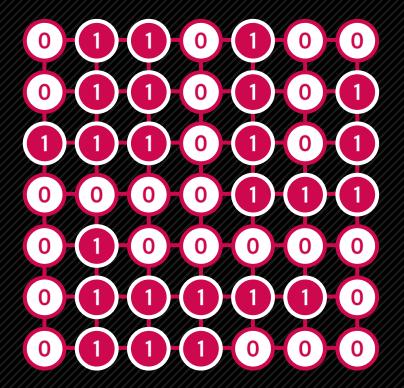






```
dx = [1, -1, 0, 0]
dy = [0, 0, 1, -1]

def dfs(x, y):
    visited[x][y] = 1
    for i in range(4):
        X = x + dx[i], Y = y + dy[i]
        if X < 0 or X == N or Y < 0 or Y == M:
             continue
        if not(visited[x][y]) and graph[x][y]:
             dfs(X, Y)</pre>
```











- 집이 있는 곳을 1, 집이 없는 곳을 0으로 하는 그래프가 주어진다.
- 상하좌우에 다른 집이 있는 경우, 그 집은 붙어있다고 한다.
- 이때, 붙어있는 단지의 수와 각 단지 마다 속한 집의 수를 출력하는 프로그램을 작성하시오.

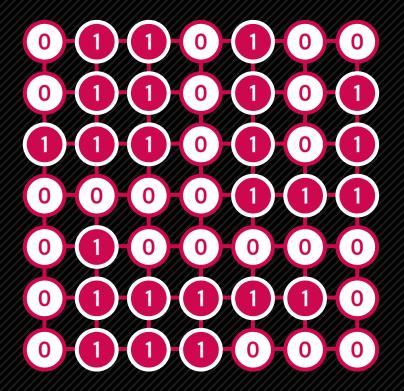
제약조건

• 그래프의 가로/세로 크기는 25 이하이다.



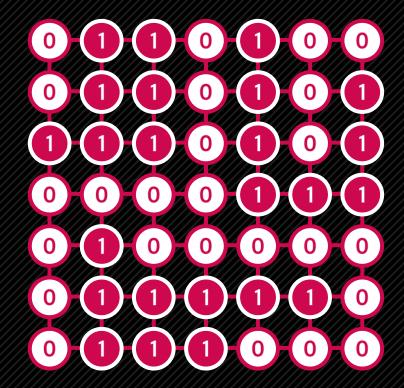


그래프의 진입점?













/<> Silver 1 - 토마토 (#7576)

요약

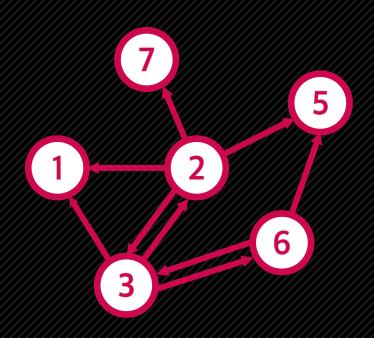
- 토마토를 N×M 크기의 상자에 넣는다.
- 만약 익지 않은 토마토 주변 (상, 하, 좌, 우) 에 익은 토마토가 있다면, 다음 날 그 토마토는 익는다.
- 상자 안에 있는 토마토가 며칠안에 모두 익는지 구하는 프로그램을 작성하시오.

제약조건

• M과 N의 범위는 2 <= N, M <= 1,000 이다.







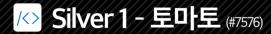
Breadth First Search

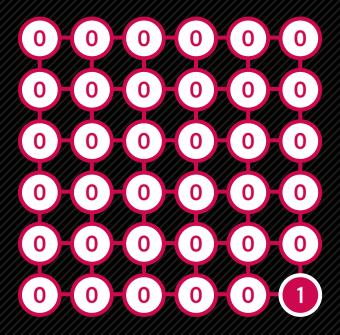
- 너비 우선 탐색이라고도 하며, 자신의 자식들부터 순차적으로 탐색.
- 순차 탐색 이후 다른 자식 노드의 자식을 확인하기 위해 규를 사용.

Depth First Search

- 깊이 우선 탐색이라고도 하며, 최대한 깊게 탐색 후 빠져 나옴.
- 백트래킹의 일종이며, 재귀를 활용함.



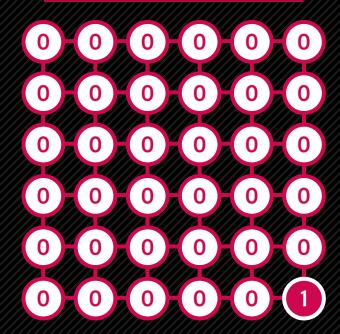




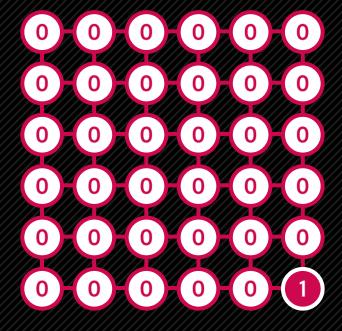




Breadth First Search



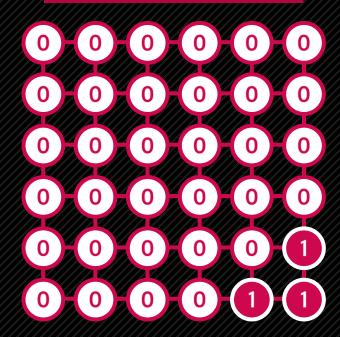
Depth First Search



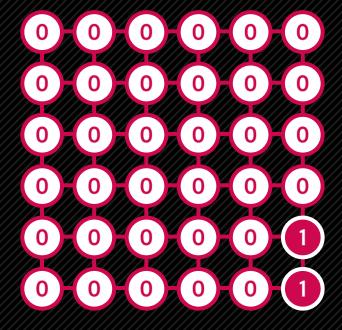




Breadth First Search

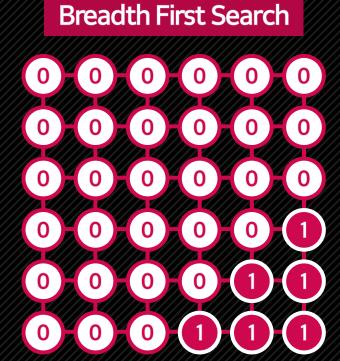


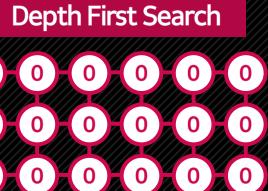
Depth First Search

















/<> Silver 1 - 토마토 (#7569)

요약

- 토마토를 N×M×H 크기의 상자에 넣는다.
- 만약 익지 않은 토마토 주변에 익은 토마토가 있다면, 다음 날 그 토마토는 익는다.
- 상자 안에 있는 토마토가 며칠안에 모두 익는지 구하는 프로그램을 작성하시오.

제약조건

• M과 N의 범위는 2 <= N, M, H <= 100 이다.





요약

- 수빈이와 동생은 숨바꼭질을 하고 있는데, <mark>현재 수빈이는 점 N에 있고, 동생은 M에 있다</mark>.
- 수빈이는 동생이 있는 곳 까지 가기 위해 <mark>걷거나</mark> (X 1 또는 X + 1로 이동) 순간이동 (2 × X로 이동)을 할 수 있다.
- 수빈이가 동생을 찾을 수 있는 가장 빠른 시간을 구하는 프로그램을 작성하시오.

제약조건

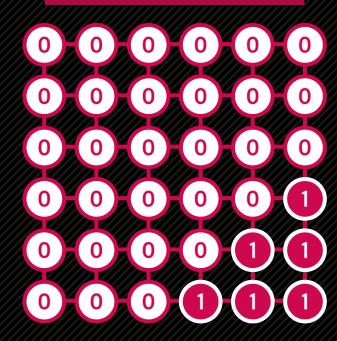
• N과 M의 범위는 0 <= N, M <= 100,000 이다.



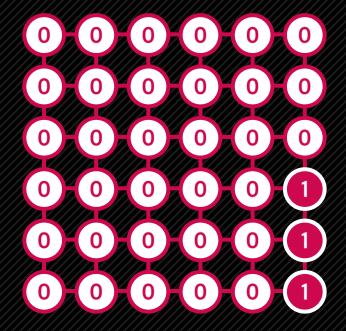




Breadth First Search



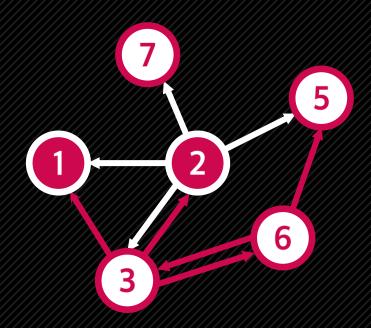
Depth First Search



BFS의 또다른 발견



Silver 1 - 숨바꼭질 (#1697)

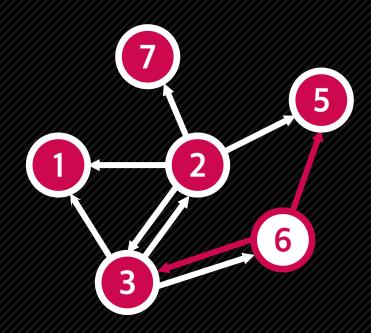


```
function bfs(pos) {
    set Q = Queue
    pos -> Q
    while Q is not empty
        set node = popped element of Q
        for children of pos
            if each child has not been visited
                 visit child
                       child -> Q
}
```

BFS의 또다른 발견



Silver 1 - 숨바꼭질 (#1697)



```
function bfs(pos) {
    set Q = Queue
    pos -> Q
    while Q is not empty
        set node = popped element of Q
        for children of pos
            if each child has not been visited
            visit child
            child -> Q
}
```







● N번 탐색 후 큐에 들어있는 내용을 모두 꺼내서 한 번 씩 탐색하면, 큐에는 N+1 번째로 방문한 값들이 들어있음이 보장됨!





요약

- 수빈이와 동생은 숨바꼭질을 하고 있는데, <mark>현재 수빈이는 점 N에 있고, 동생은 M에 있다</mark>.
- 수빈이는 동생이 있는 곳 까지 가기 위해 <mark>걷거나</mark> (X 1 또는 X + 1로 이동) 순간이동 (2 × X로 이동)을 할 수 있다.
- 수빈이가 동생을 찾을 수 있는 가장 빠른 시간을 구하는 프로그램을 작성하시오.

제약조건

• N과 M의 범위는 0 <= N, M <= 100,000 이다.







• 수빈이가 동생을 찾을 수 있는 가장 빠른 시간을 구하는 프로그램을 작성하시오.



● N번 탐색 후 큐에 들어있는 내용을 모두 꺼내서 한 번 씩 탐색하면, 큐에는 N + 1 번째로 방문한 값들이 들어있음이 보장됨!







● 첫 방문: 1회 이동하여 방문 가능한 모든 지점



● K번째 방문 중 M이 큐에 들어오게 되면 최단시간!





Cold 5 - 숨바꼭질 3 (#13549)

요약

- 수빈이와 동생은 숨바꼭질을 하고 있는데, <mark>현재 수빈이는 점 N에 있고, 동생은 M에 있다</mark>.
- 수빈이는 동생이 있는 곳 까지 가기 위해 겉거나 (X 1 또는 X + 1로 이동) 순간이동 (2 × X로 이동, 0초)을 할 수 있다.
- 수빈이가 동생을 찾을 수 있는 가장 빠른 시간을 구하는 프로그램을 작성하시오.

제약조건

• N과 M의 범위는 0 <= N, M <= 100,000 이다.





요약

- N×M 크기의 맵이 있는데, (1, 1)에서 (N, M)까지 이동하려고 한다.
- 만약 이동하는 중 벽을 부수고 이동하는 것이 더 빠르다면, 벽을 한 개 까지 부수고 이동해도 된다.
- 맵이 주어졌을 때, 최단 거리를 구하는 프로그램을 작성하시오.

제약조건

• N과 M의 범위는 0 <= N, M <= 1,000 이다.



Gold 4 - 벽 부수고 이동하기 (#2206)

일반적인 DFS, BFS를 한다면…

??: 너 떡 몇 개 华泉어? ??: 그러게&?



● 플래그를 넣어주자! (벽을 현재 부순 상태인지)

```
def bfs():
_q = deque()
_q.append((0, 0, 0)) # 세번째인자: flag
```



● 플래그만 넣으면 다야?





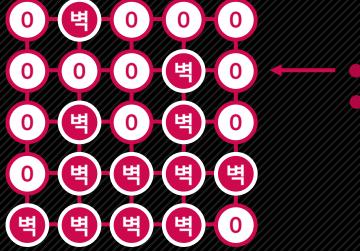
● 플래그만 넣으면 다야?





Gold 4 - 벽 부수고 이동하기 (#2206)

● 플래그만 넣으면 다야?



- 벽을 부순 경우, 해당 칸에 5호 만에, 아닌 경우 7호 만에 도달함.
- 벽을 부수지 않고 도달하면 이미 visit 상태라 방문하지 않고 넘어감.



Gold 4 - 벽 부수고 이동하기 (#2206)

```
visited[1001][1001][2];

def bfs():
    _q = deque()
    _q.append((0, 0, 0)) #서번째인자: flag
```

● 벽을 부순 상태와 부수지 않은 상태의 **visit 값 분리**!







Gold 5 - 연구소 (#14502)

요약

- N×M 크기의 연구소에 바이러스가 퍼졌다. 퍼지지 않은 다른 곳을 보호하기 위해 벽을 설치해야 한다.
- 벽은 정확히 3개를 세울 수 있다.
- 이때, 얻을 수 있는 안전 구역의 크기의 최댓값을 구하는 프로그램을 작성하시오.

제약조건

• N과 M의 범위는 3 <= N, M <= 8 이다.

완전탐색!



Gold 5 - 연구소 (#14502)

요약

- N×M 크기의 연구소에 바이러스가 퍼졌다. 퍼지지 않은 다른 곳을 보호하기 위해 벽을 설치해야 한다.
- 벽은 정확히 3개를 세울 수 있다.
- 이때, 얻을 수 있는 안전 구역의 크기의 최댓값을 구하는 프로그램을 작성하시오.

제약조건

• N과 M의 범위는 3 <= N, M <= 8 이다.

접근

- 벽 3개를 고르는 경우의 수는?
- → 64C3 = **41,664**
- 그러면 벽 3개 고르고 탐색해도 시간 남겠네?







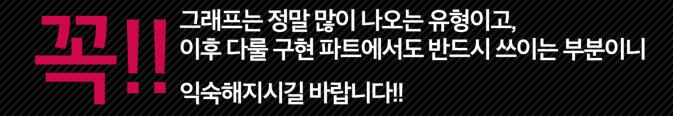
요약

- N×M 크기의 보물지도가 있다. 보물지도에는 육지와 바다가 표시되어 있다.
- 보물은 서로 간에 최단거리로 이동하는데 있어 가장 긴 시간이 걸리는 육지 두 곳에 묻혀 있다.
- 보물이 묻혀 있는 두 곳간의 최단 거리로 이동하는 시간을 구하는 프로그램을 작성하시오.

제약조건

• N과 M의 범위는 N, M <= 50 이다.





추가 추천 문제



- Silver 2 나이트의 이동 (#7562)
 - 나이트는 특이하게 움직입니다. 배운 것을 활용해서 나이트의 이동을 정의할 수 있을까요?
- (#16964)
 - DFS의 원리를 떠올려 보세요! 정 안되면 첫번째 자료를 참고해보세요. 힌트가 나와있습니다.
- Gold 4 빙산 (#2573)
 - 조금 생각을 해봐야 하는 문제입니다. 꼭 풀어보세요!
- **Level 3 단어 변환** (#43163)
 - 그래프가 아닌 것 같이 생겼습니다. 모델링이 중요한 문제입니다!

</>;

To be continue,,,