[Algorithm] 20강 : DFS & BFS 기초 문제 풀이 — 나무늘보의 개발 블로그

노트북: 첫 번째 노트북
만든 날짜: 2020-11-04 오전 8:58
URL: https://continuous-development.tistory.com/176

<문제> 음료수 얼려먹기

N x M 크기의 얼음 틀이 있습니다. 구멍이 뚫려있는 부분은 0 , 칸막이가 존재하는 부분은 1로 표시됩니다. 구멍이 뚫려있는 부분끼리 상, 하, 좌, 우로 붙어있는 경우 서로 연결되어 있는 것으로 간주합니다. 이때 얼음 틀의 모양이 주어졌을 때 생성되는 총아이스크림의 개수를 구하는 프로그램을 작성하세요. 다음의 4 x 5 얼음 틀 예시에서는 아이스크림이 총 3개 생성됩니다.

```
00110
00011
11111
00000
```

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

풀이 시간 30 분 | 시간제한 1초 | 메모리 제한 128MB
입력 조건 1. 첫 번째 줄에 얼음 틀의 세로 길이 N과 가로길이 M이 주어집니다(1 <= N, M <=1000)
2. 두 번째 줄부터 N+1번째 줄까지 얼음 틀의 형태가 주어집니다.
3. 이때 구멍이 뚫려있는 부분은 0, 그렇지 않은 부분은 1입니다.
출력 조건 : 한 번에 만들 수 있는 아이스크림의 개수를 출력합니다.

입력 예시

```
4 5
00110
00011
11111
00000
```

출력 예시

```
3
```

#문제풀이 방법

이 문제는 DFS 혹은 BFS로 해결할 수 있다.
DFS를 활용하는 알고리즘은 다음과 같다
1. 특정한 지점의 주변 상, 하, 좌, 우를 살펴본 뒤에 주변 지점중에서 값이 0 이면서 아직 방문하지 않은 지점이 있다면 해당 지점을 방문한다.
2. 방문한 지점에서 다시 상,하,좌,우를 살펴보면서 방문을 진행하는 과정을 반복하면, 연결된 모든 지점을 방문할 수 있습니다.
3. 모든 노드에 대하여 1~2번의 과정을 반복하며, 방문하지 않은 지점의 수를 카운트한다.

#구현

```
# DFS로 특정 노드를 방문하고 연결된 모든 노드들도 방문
def dfs(x,y):
    # 주어진 범위를 벗어나는경우에는 즉시 종료
    if x <= -1 or x>=n or y <= -1 or y >=m:
        return False
    #현재 노드를 아직 방문하지 않았다면
    if graph[x][y]==0:
        #해당 노드 방문처리
        graph[x][y] = 1
        # 상, 하, 좌, 우의 위치들도 모두 재귀적으로 호출
        dfs(x-1,y)
        dfs(x,y-1)
        dfs(x+1,y)
        dfs(x,y+1)
        retrun False

# N,M을 공백을 기준으로 구분하며 입력받기
n,m = map(int,input().split())

# 2차원 리스트의 맵 정보 입력받기
graph = []
for i in range(n):
    graph.append(list(map(int,input())))

# 모든 노드(위치)에 대하여 음료수 채우기
result = 0
for i in range(n):
    for j in range(m):
        # 현재 위치에서 DFS 수행
        if dfs(i,j) == True:
            result += 1

print(result) # 정답 출력
```

<문제> 미로 탈출

동빈이는 N x M 크기의 직사각형 형태의 미로에 갇혔습니다. 미로에는 여러 마리의 괴물이 있어 이를 피해 탈출해야 합니다.
동빈이의 위치는 (1,1)이며 미로의 출구 (N, M)의 위치에 존재하며 한 번에 한 칸씩 이동할 수 있습니다. 이때 괴물이 있는 부분은 0으로, 괴물이 없는 부분은 1로 표시되어 있습니다. 미로는 반드시 탈출할 수 있는 형태로 제시된다.
이때 동빈이가 탈출하기 위해 움직여야 하는 최소 칸의 개수를 구하세요. 칸을 셀 때는 시작 칸과 마지막 칸을 모두 포함해서 계산합니다.

풀이 시간 30분 | 시간제한 1초 | 메모리 제한 128MB

입력 조건 : 첫째 줄에 두 정수 N, M(4 <=N, M <=200)이 주어진다. 다음 N개의 줄에는 각각 M개의 정수(0 혹은 1)로 미로의 정보가 주어진다. 각각의 수들은 공백 없이 붙어서 입력으로 제시된다. 또한 시작 칸과 마지막 칸은 항상 1이다.

출력 조건 : 첫째 줄에 최소 이동 칸의 개수를 출력합니다.

#문제 풀이 방법

BFS는 시작 지점에서 가까운 노드부터 차례대로 그래프의 모든 노드를 탐색한다
상, 하, 좌, 우로 연결된 모든 노드로의 거리가 1로 동일하다. 따라서 (1,1) 지점부터 BFS를 수행하여 모든 노드의 최단 거리 값을 기록하면 해결할 수 있다.

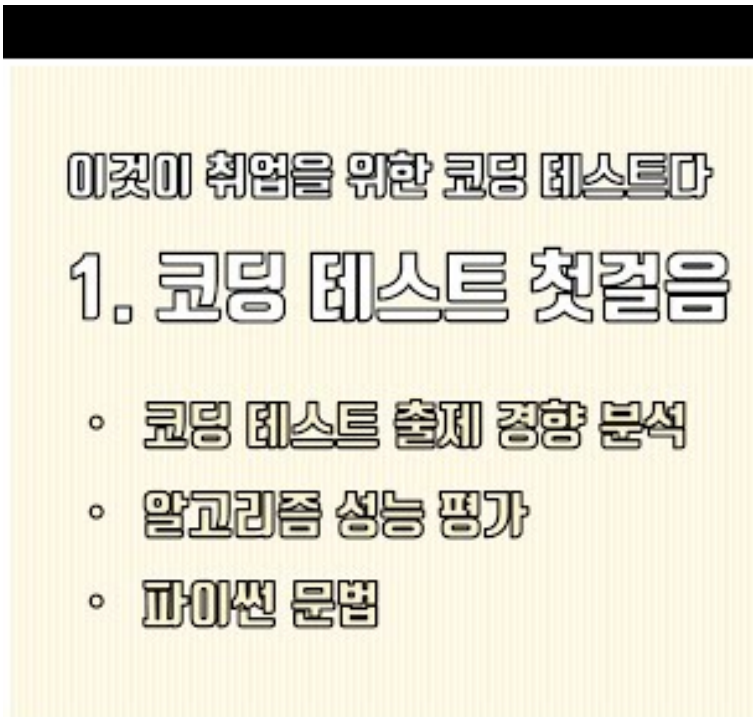
#구현

```
# BFS 소스코드 구현
def bfs(x,y):
    # 큐 구현을 위해 deque 라이브러리 사용
    queue = deque()
    queue.append((x,y))
    # 큐가 빌 때까지 반복하기
    while queue:
```

```
        x,y = queue.popleft()
        # 현재 위치에서 4가지 방향으로 위치 확인
        for i in range(4):
            nx = x + dx[i]
            ny = y + dy[i]
            # 미로 찾기 공간을 벗어난 경우 무시
            if nx < 0 or nx >= n or ny < 0 or ny >= m:
                continue
            # 벽인 경우 무시
            if graph[nx][ny] == 0:
                continue
            # 해당 노드를 처음 방문하는 경우에만 최단 거리 기록
            if graph[nx][ny] == 1:
                graph[nx][ny] = graph[x][y] + 1
        # 가장 오른쪽 아래까지 최단 거리 변환
        return graph[n-1][m-1]

from collections import deque

# N,M 을 공백을 기준으로 구분하여 입력 받기
n, m = map(int,input().split())

# 2차원 리스트의 맵 정보 입력받기
graph = []

for i in range(n):
    graph.append(list(map(int,input())))

# 이동할 네 가지 방향 정의 (상, 하, 좌, 우)
dx = [-1,1,0,0]
dy = [0,0,-1,1]

# BFS를 수행한 결과 출력
print(bfs(0,0))
```

---

www.youtube.com/watch?v=m-9pAwq1o3w&list=PLRx0vPvlEmdAghTr5mXQxGpHjWqSz0dgC



[redacted]

이 자료는 동빈 나 님의 이코 테 유튜브 영상을 보고 정리한 자료입니다.

'**Algorithm**' 카테고리의 다른 글□

[Algorithm] 20강 : DFS & BFS 기초 문제 풀이□

[Algorithm] 19강 : BFS(너비 우선 탐색) 알고리즘 정의와 예제□

[Algorithm] 18강 : DFS(깊이 우선 탐색) 알고리즘 정의와 예제□

[Algorithm] 17강 : 재귀함수의 정의와 예제□

[Algorithm] 16강 : 스택과 큐 자료구조□

[Algorithm] 15강 : 구현 유형 문제 풀이□