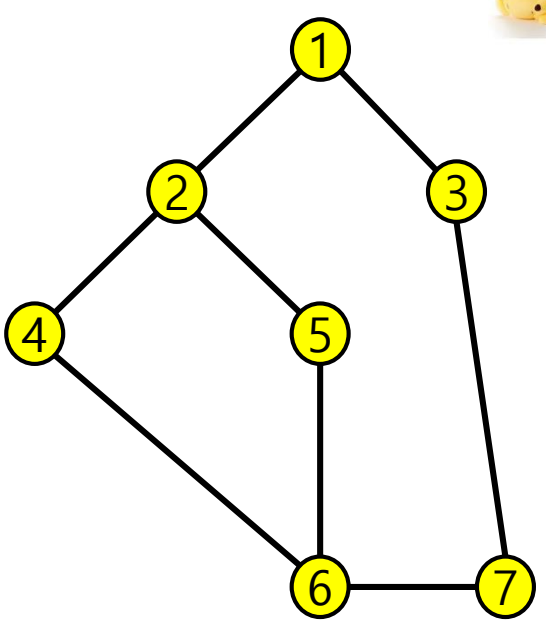
비선형자료구조의 완전탐색 : DFS, BFS

AD 보충수업 1일차

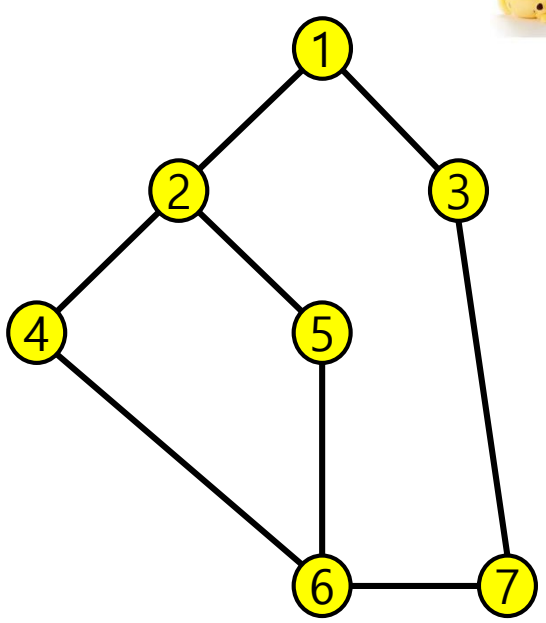
DFS




```
G = [[], [2, 3], [1, 4, 5], [1, 7], [2, 6], [2, 6], [4, 5, 7], [3, 6]]
visited = [0] * 8

dfs(1)
```

DFS

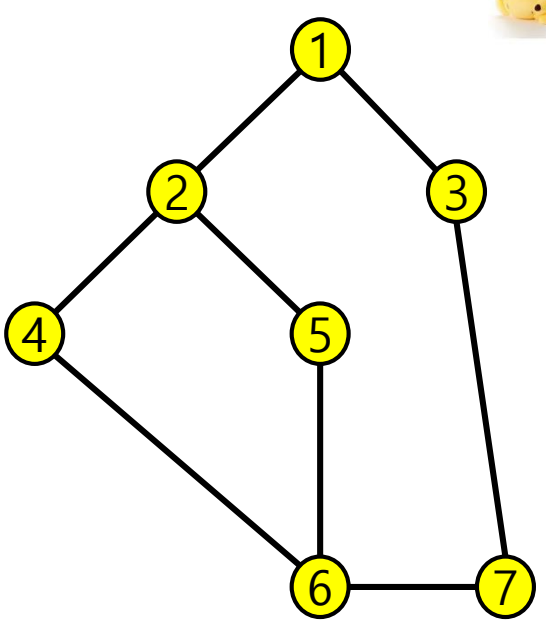
```
def dfs(v):
    s = []
    s.append(v)
    while s:
        v = s.pop(-1)
        if not visited[v]:
            visited[v] = 1
            print(v, end=' ')
            for w in G[v]:
                if not visited[w]:
                    s.append(w)
```




```
G = [[], [2, 3], [1, 4, 5], [1, 7], [2, 6], [2, 6], [4, 5, 7], [3, 6]]
visited = [0] * 8

dfs(1)
```

DFS

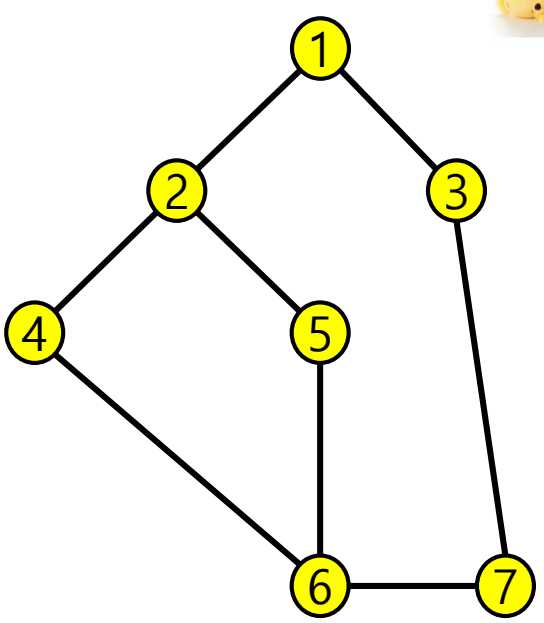
```
def dfsr(v):
    visited[v] = 1
    print(v, end =' ')
    for w in G[v]:
        if not visited[w]:
            dfsr(w)
```




```
G = [[], [2, 3], [1, 4, 5], [1, 7], [2, 6], [2, 6], [4, 5, 7], [3, 6]]
visited = [0] * 8

dfs(1)
```

BFS

```
def bfs(v):
    q = []
    q.append(v)
    while q:
        v = q.pop(0)
        if not visited[v]:
            visited[v] = 1
            print(v, end=' ')
            for w in G[v]:
                if not visited[w]:
                    q.append(w)
```




```
G = [[], [2, 3], [1, 4, 5], [1, 7], [2, 6], [2, 6], [4, 5, 7], [3, 6]]
visited = [0] * 8

dfs(1)
```

BFS 와 DFS

```python
def bfs(v):
    q = []
    q.append(v)
    while q:
        v = q.pop(0)
        if not visited[v]:
            visited[v] = 1
            print(v, end=' ')
            for w in G[v]:
                if not visited[w]:
                    q.append(w)
```

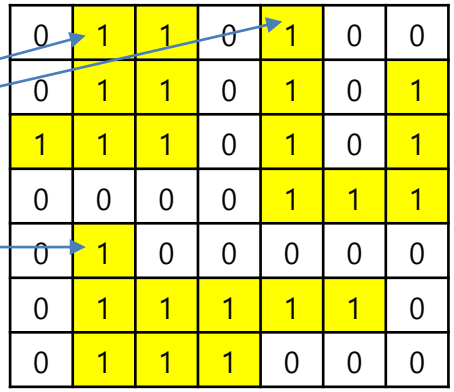
```python
def dfs(v):
    s = []
    s.append(v)
    while s:
        v = s.pop(-1)
        if not visited[v]:
            visited[v] = 1
            print(v, end=' ')
            for w in G[v]:
                if not visited[w]:
                    s.append(w)
```

단지번호붙이기

https://www.acmicpc.net/problem/2667

단지번호붙이기

```
N = int(input())
mat = [list(map(int, input())) for _ in range(N)]

res = []

for i in range(N):
    for j in range(N):
        if mat[i][j]:
            res.append(dfs(i, j))

res.sort()
print(len(res))
for i in res:
    print(i)
```



7	8	9

7	8	9

단지번호붙이기

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

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

0,1에서 시작

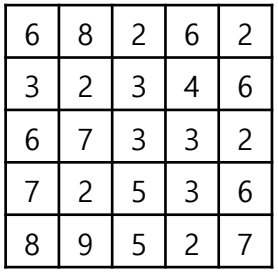
```
def dfs(x, y):
    mat[x][y] = 0

    ret = 0
    for (dx, dy) in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
        xx, yy = x + dx, y + dy
        if not (0 <= xx < N and 0 <= yy < N): continue
        if mat[xx][yy]:
            ret += dfs(xx, yy)

    return ret + 1
```

0	0	0
1	1	0
1	0	0

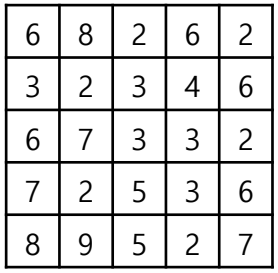
이웃한 노드의 개수와 자기 노드의 합 3을 반환

안전 영역

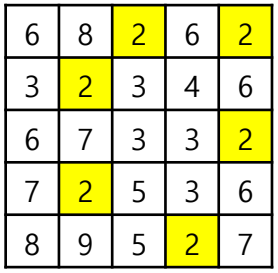
https://www.acmicpc.net/problem/2468



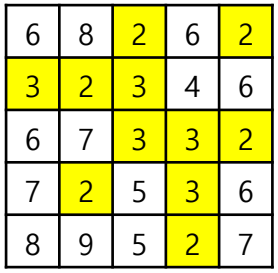
K=0



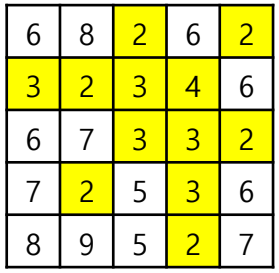
K=1



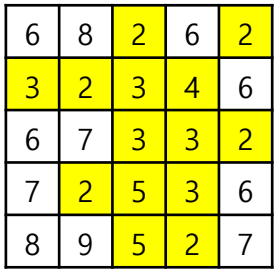
K=2



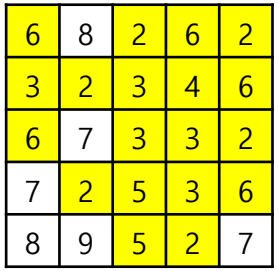
K=3



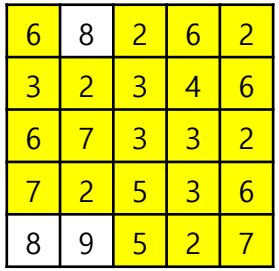
K=4



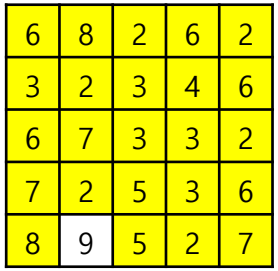
K=5



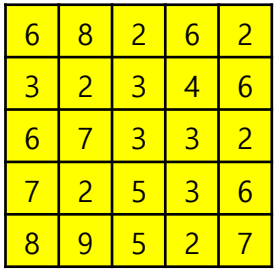
K=6



K=7



K=8



K=9

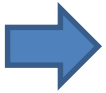
안전 영역

```
N = int(input())
mat = [list(map(int, input().split())) for _ in range(N)]

ans = 1
for k in range(1, max(sum(mat, []))):
    visited = [[0] * N for _ in range(N)]
    cnt = 0
    for i in range(N):
        for j in range(N):
            if not visited[i][j] and mat[i][j] > k:
                BFS(i, j, k)
                cnt += 1
    ans = max(ans, cnt)

print(ans)
```

6	8	2	6	2
3	2	3	4	6
6	7	3	3	2
7	2	5	3	6
8	9	5	2	7

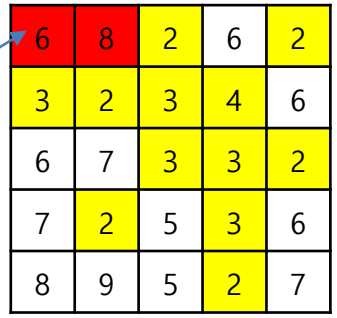
K=4

0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

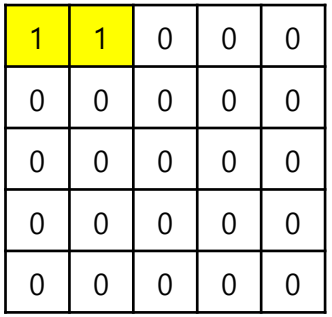


1	1	0	1	0
0	0	0	0	1
1	1	0	0	0
1	0	1	0	1
1	1	1	0	1

안전 영역



K=4



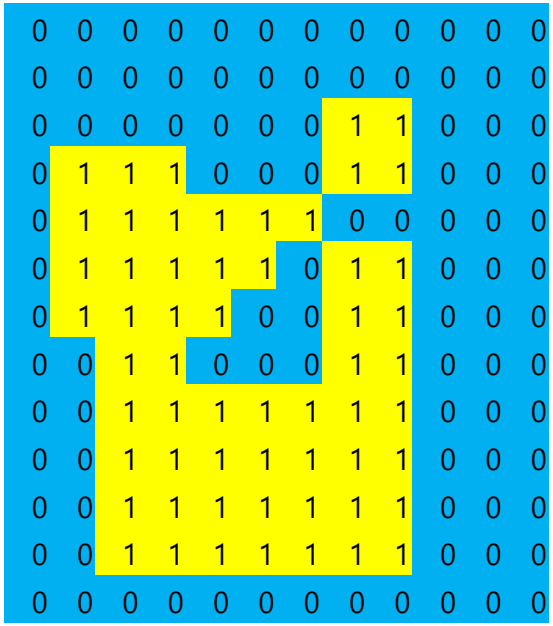

```
def BFS(x, y, k):
    q = []
    q.append((x, y))
    visited[x][y] = 1

    while q:
        x, y = q.pop(0)
        for dx, dy in (-1, 0), (1, 0), (0, -1), (0, 1):
            xx = x + dx
            yy = y + dy
            if not (0 <= xx < N and 0 <= yy < N): continue
            if not visited[xx][yy] and mat[xx][yy] > k:
                q.append((xx, yy))
                visited[xx][yy] = 1
```

치즈

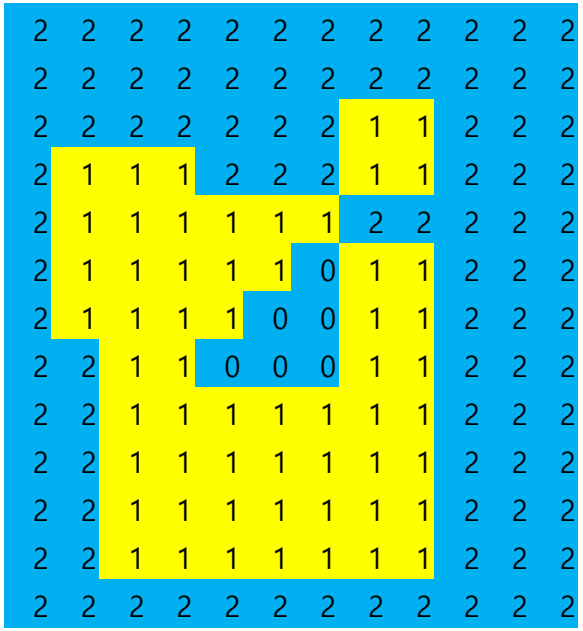
https://www.acmicpc.net/problem/2636

치즈

```
# step1 : 치즈 외부 공기 2로 표시하기
# step2 : 치즈의 공기와 접촉된 면을 표시하기 3, 치즈 내부 4
# step3 : 치즈 녹이기, 자료 재 정리, 치즈 개수 반환

N, M = map(int, input().split())
mat = [list(map(int, input().split())) for _ in range(N)]

cnt = 0
while True:
    cnt += 1
    step1(0, 0)
    for i in range(N):
        for j in range(M):
            if mat[i][j] == 1:
                step2(i, j)
    last_cheeze = step3()
    if not sum(sum(mat, [])) : break


print(cnt)
print(last_cheeze)
```

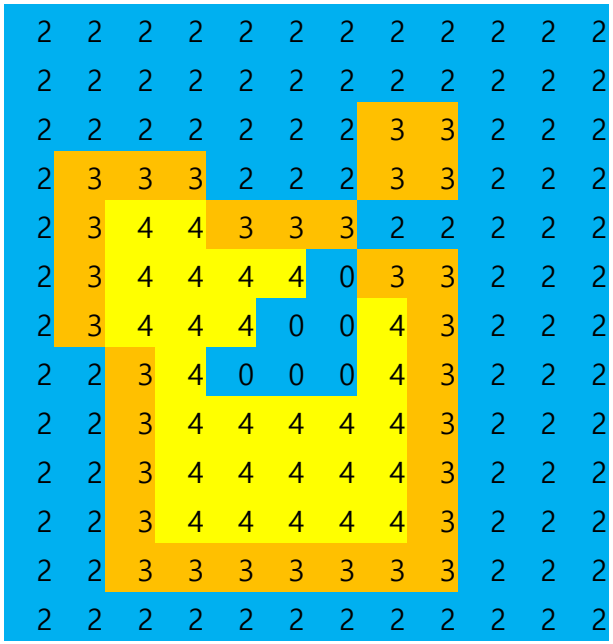


치즈



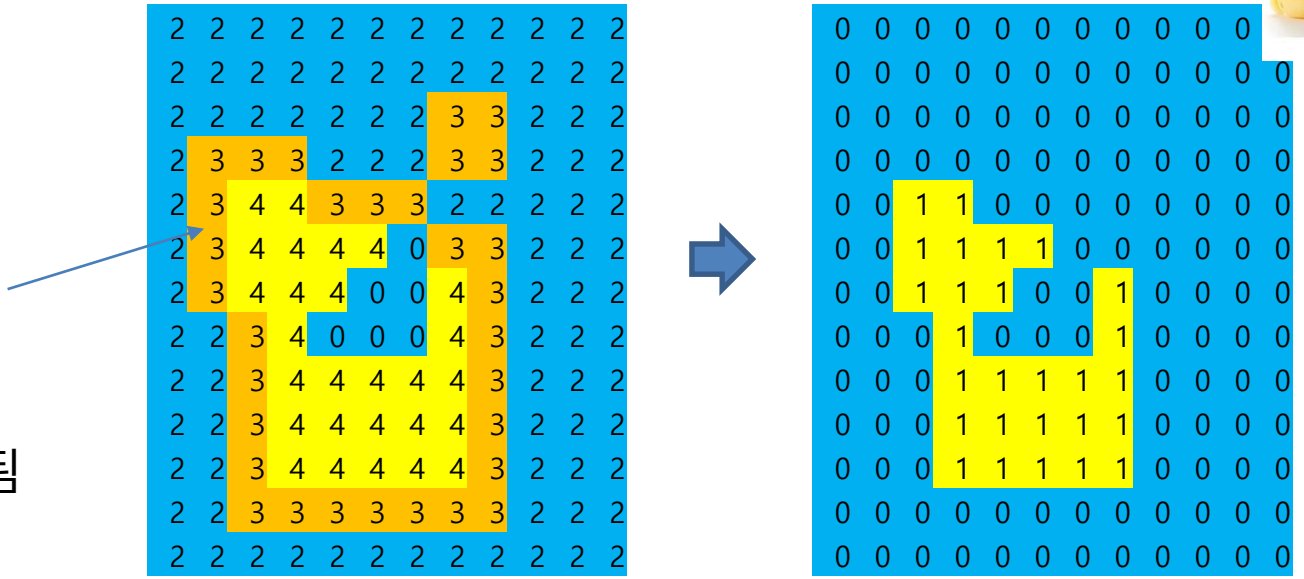

```
# 치즈 외부 공기 2로 표시하기
def step1(x, y):
    q = []
    q.append((x, y))
    mat[x][y] = 2

    while q:
        x, y = q.pop(0)

        for (dx, dy) in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
            xx, yy = x + dx, y + dy

            if not (0 <= xx < N and 0 <= yy < M): continue
            if mat[xx][yy] == 0 :
                mat[xx][yy] = 2
                q.append((xx, yy))
```

치즈




```
# 치즈의 공기와 접촉된 면을 표시하기 3, 치즈 내부 4
def step2(x, y):
    q = []

    mat[x][y] = 3

    q.append((x, y))
    while q:
        x, y = q.pop(0)

        for (dx, dy) in [(1, 0), (-1, 0), (0, 1), (0, -1)]:
            xx, yy = x + dx, y + dy

            if mat[xx][yy] == 1 :
                mat[xx][yy] = 4
                if mat[xx + 1][yy] == 2 or mat[xx - 1][yy] == 2 \
                        or mat[xx][yy + 1] == 2 or mat[xx][yy - 1] == 2:
                    mat[xx][yy] = 3
                q.append((xx, yy))
```

치즈



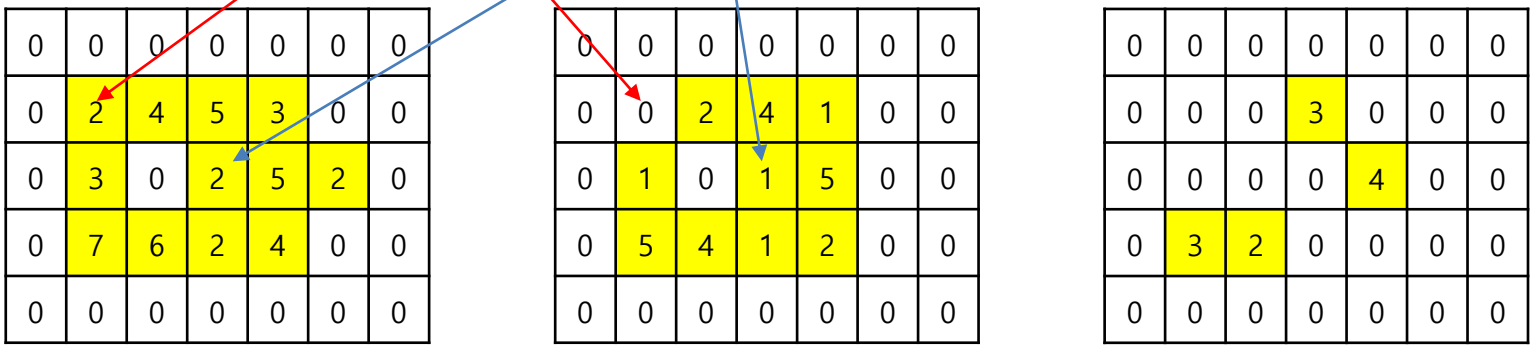
녹인 치즈의 개수
cnt가 반환
last_cheeze값이 됨

```
# 치즈 녹이기, 자료 재 정리, 녹인 치즈 개수 반환
def step3():
    cnt = 0
    for i in range(N):
        for j in range(M):
            if mat[i][j] == 2: mat[i][j] = 0
            elif mat[i][j] == 3: mat[i][j] = 0; cnt += 1
            elif mat[i][j] == 4: mat[i][j] = 1

    return cnt
```

빙산

https://www.acmicpc.net/problem/2573

이웃한 물의 개수 만큼 녹음



1년 후

2년 후
빙산이 분리됨

빙산

```
N, M = map(int, input().split())
mat = [list(map(int, input().split())) for _ in range(N)]

ans = 1

while True:
    x, y, cnt = step1()    #녹인다.
    if cnt == 0:
        ans = 0
        break
    cnt1 = BFS(x, y)
    if cnt != cnt1:
        break
    ans += 1

print(ans)
```

0	0	0	0	0	0	0
0	0	2	4	1	0	0
0	1	0	1	5	0	0
0	5	4	1	2	0	0
0	0	0	0	0	0	0

1년 후

0	0	0	0	0	0	0
0	0	0	3	0	0	0
0	0	0	0	4	0	0
0	3	2	0	0	0	0
0	0	0	0	0	0	0

2년 후
빙산이 분리됨

cnt1 ➔ 빙산 내의 얼음 개수 2 반환

x, y, cnt ➔ 마지막 얼음의 좌표와 남아있는 얼음개수 3, 2, 4 반환

빙산

```
def step1():
    # 이웃한 물의 개수
    mat2 = [[0] * M for i in range(N)]

    for x in range(N):
        for y in range(M):
            if mat[x][y] == 0: continue
            for dx, dy in (-1, 0), (1, 0), (0, -1), (0, 1):
                xx, yy = x + dx, y + dy
                if mat[xx][yy] == 0:
                    mat2[x][y] += 1

    cnt = x = y = 0
    for i in range(N):
        for j in range(M):
            mat[i][j] -= mat2[i][j]
            if mat[i][j] < 0 : mat[i][j] = 0
            if mat[i][j] :
                cnt += 1
                x = i
                y = j
    return x, y, cnt
```

마지막 얼음 위치, 얼음의 개수 반환

mat

0	0	0	0	0	0	0
0	0	2	4	1	0	0
0	1	0	1	5	0	0
0	5	4	1	2	0	0
0	0	0	0	0	0	0

mat2

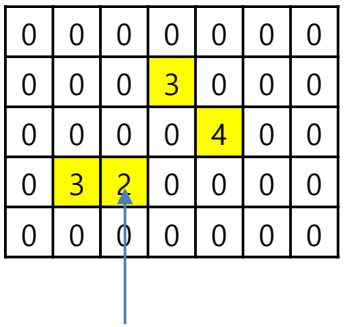
0	0	0	0	0	0	0
0	0	3	1	2	0	0
0	3	0	1	1	0	0
0	2	2	1	2	0	0
0	0	0	0	0	0	0

mat – mat2

mat'

0	0	0	0	0	0	0
0	0	0	3	0	0	0
0	0	0	0	4	0	0
0	3	2	0	0	0	0
0	0	0	0	0	0	0

빙산



빙산 내의 얼음 개수 2개

```
def BFS(x, y):
    visited = [[0] * M for _ in range(N)]
    q = []
    q.append((x, y))
    visited[x][y] = 1
    cnt = 1

    while q:
        x, y = q.pop(0)
        for dx, dy in (-1, 0), (1, 0), (0, -1), (0, 1):
            xx, yy = x + dx, y + dy
            if not visited[xx][yy] and mat[xx][yy] > 0:
                q.append((xx, yy))
                visited[xx][yy] = 1
                cnt += 1
    return cnt
```

보물섬

https://www.acmicpc.net/problem/2589

W	L	L	W	W	W	L
L	L	L	W	L	L	L
L	W	L	W	L	W	W
L	W	L	W	L	L	L
W	L	L	W	L	W	W

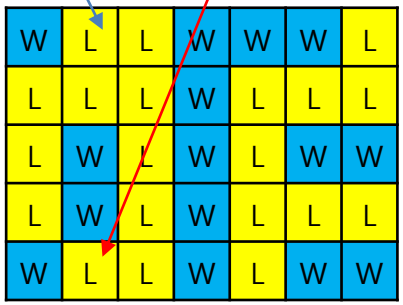
W	L	L	W	W	W	L
L	L	L	W	L	L	L
L	W	L	W	L	W	W
L	W	L	W	L	L	L
W	L	L	W	L	W	W

※ 최단 거리를 구할 때는 DFS 보다는 BFS

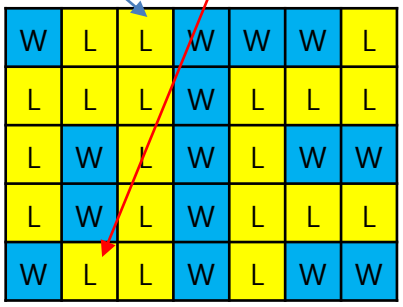
보물섬

```
N, M = map(int, input().split())
G = [input() for i in range(N)]

ans = 0
for i in range(N):
    for j in range(M):
        if G[i][j] == 'L':
            ans = max(ans, BFS(i, j))
print(ans)
```

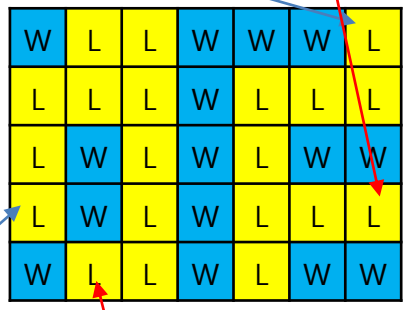
처음 시작 ➔ 6 반환



다음 시작 ➔ 5 반환

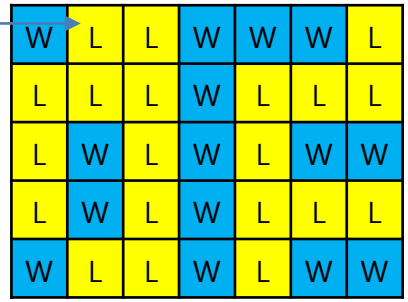


그 다음 시작 ➔ 7 반환



답은 이곳 시작 ➔ 8 반환

보물섬




```
def BFS(x, y):
    q = []
    dist = [[0] * M for i in range(N)]

    q.append((x, y))
    dist[x][y] = 1

    while q:
        x, y = q.pop(0)

        for dx, dy in ((0, 1), (0, -1), (1, 0), (-1, 0)):
            xx, yy = x + dx, y + dy

            if not (0 <= xx < N and 0 <= yy < M): continue
            if G[xx][yy] == 'L' and dist[xx][yy] == 0:
                q.append((xx, yy))
                dist[xx][yy] = dist[x][y] + 1

    return max(sum(dist, [])) - 1
```

dist

0	1	2	0	0	0	0
3	2	3	0	0	0	0
4	0	4	0	0	0	0
5	0	5	0	0	0	0
0	7	6	0	0	0	0