

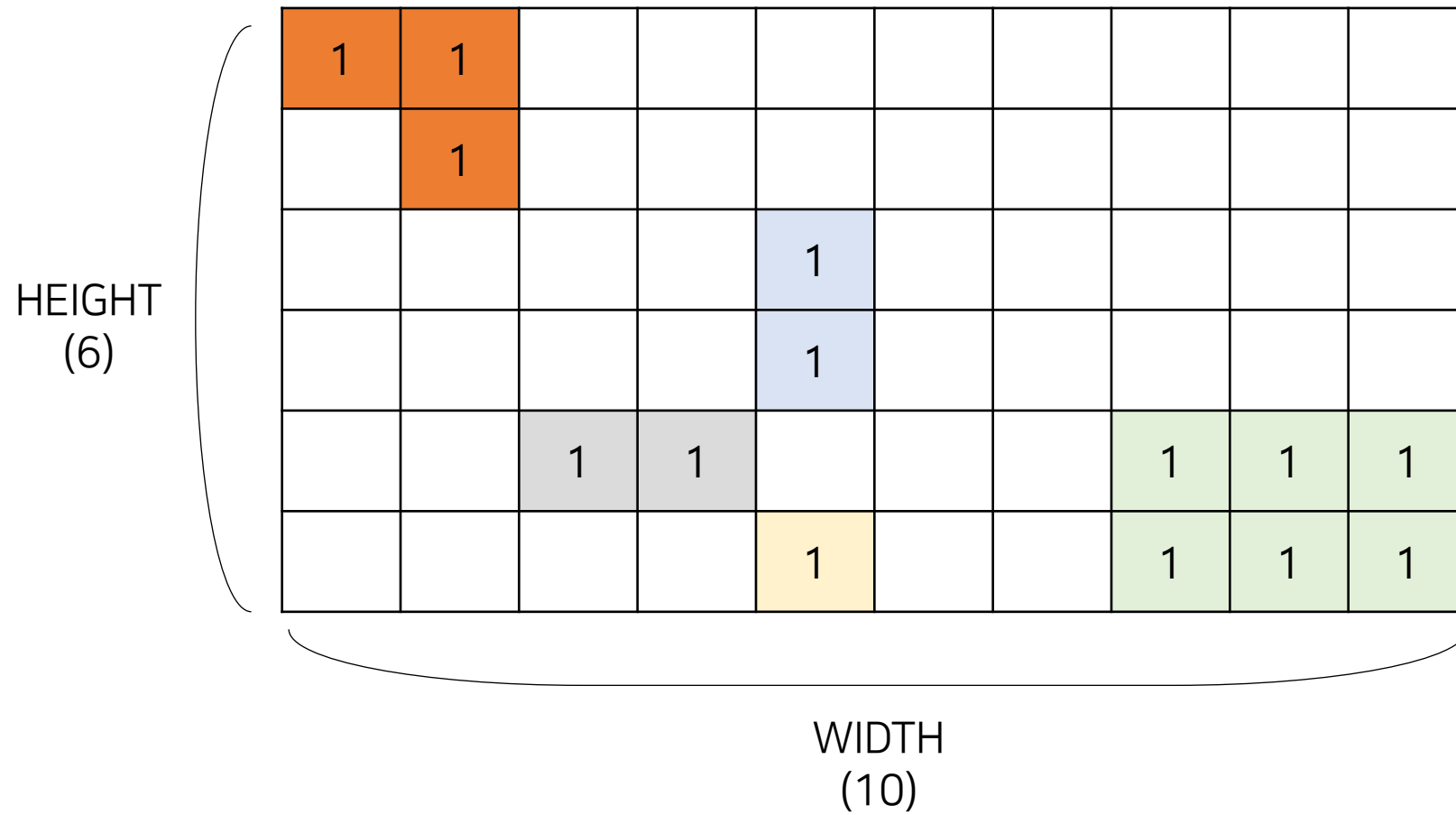
BFS(너비 우선 탐색)

1012번 문제풀이

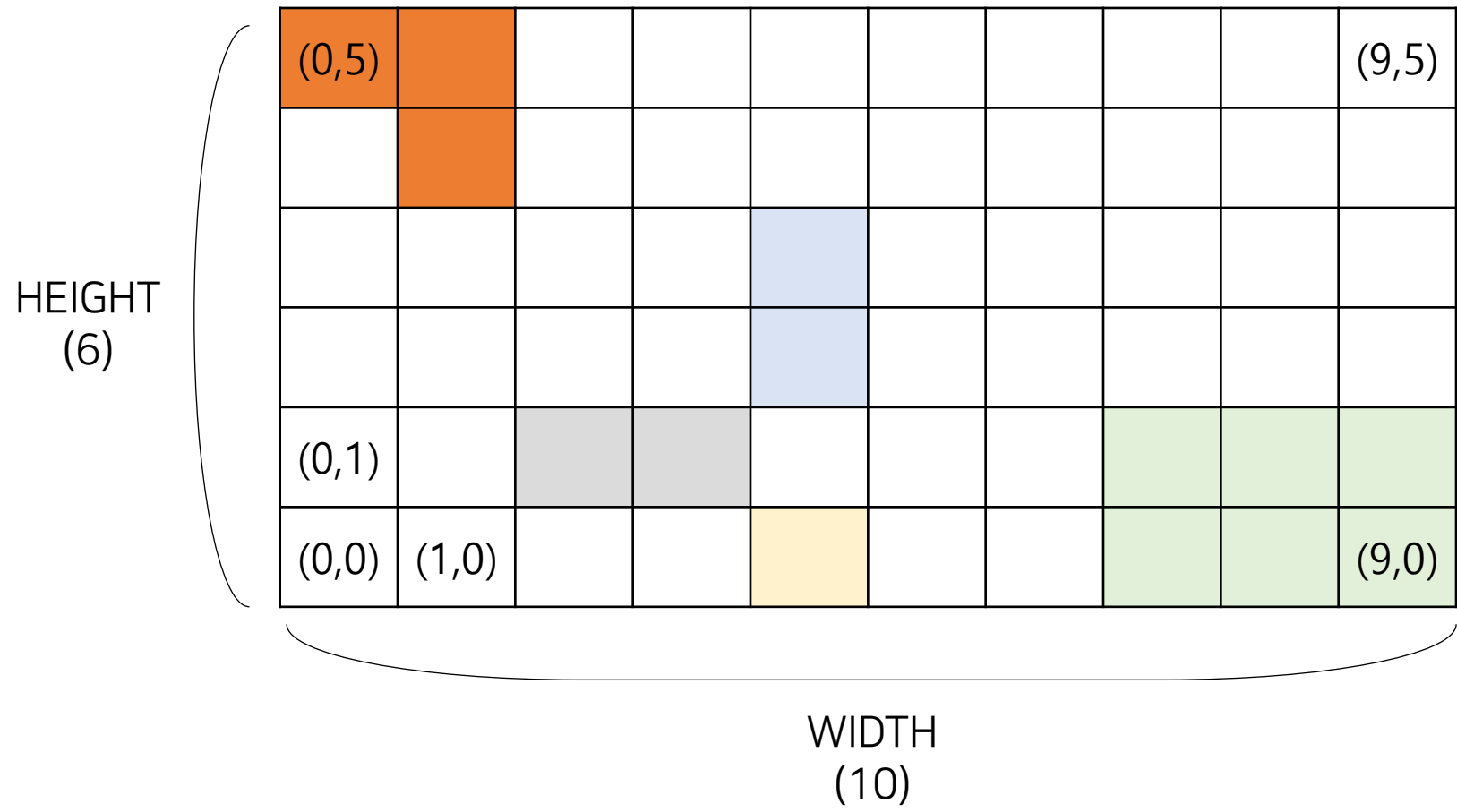
<https://www.acmicpc.net/problem/1012>

김은채
2021-09-25

문제 해석



| 좌표

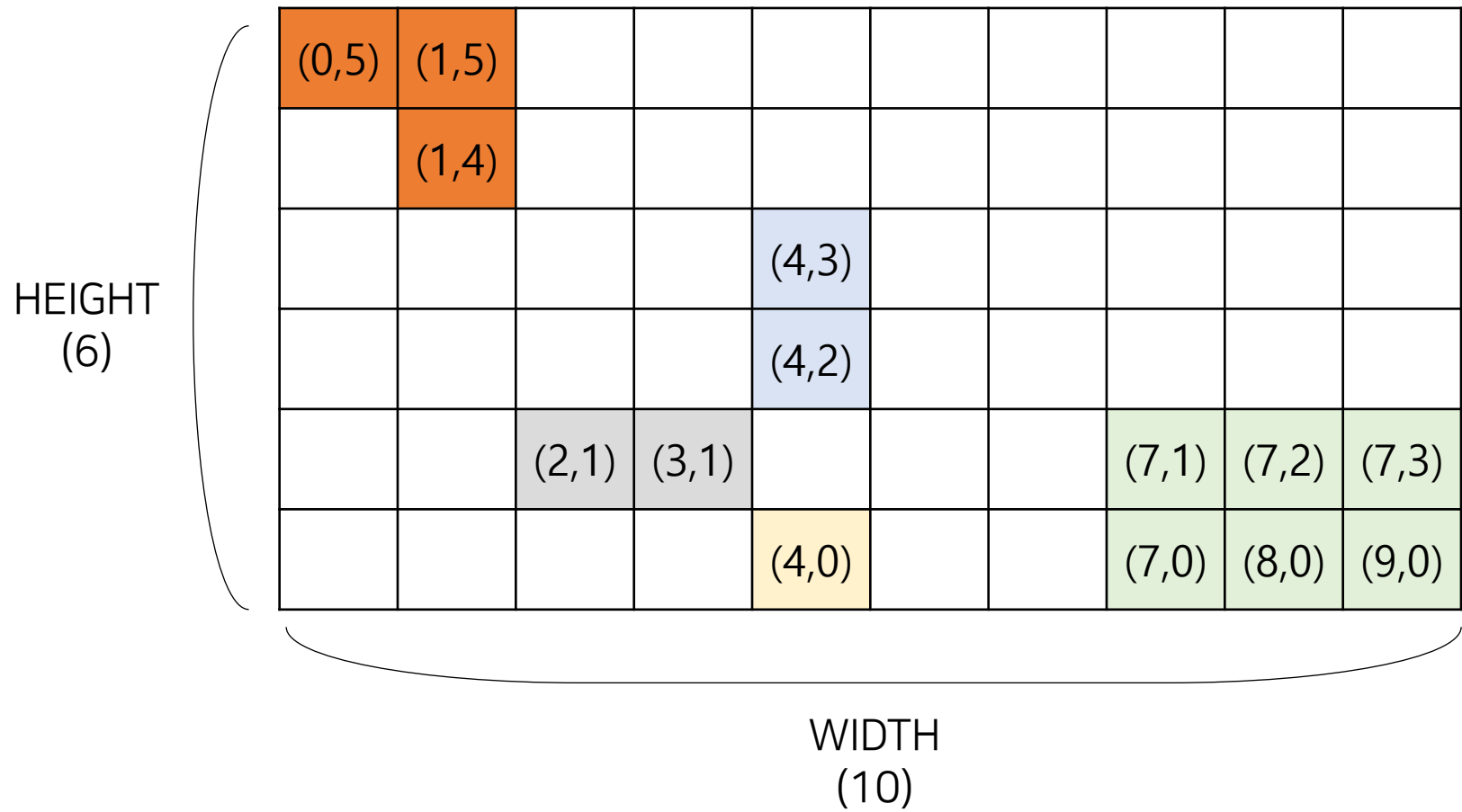


I 입력값

14개 배추의
좌표

```
1 테스트 케이스 횟수
10 6 14 WIDTH HEIGHT 심어진배추수
0 5
1 4
1 5
2 1
3 1
4 0
4 2
4 3
7 0
7 1
8 0
8 1
9 0
9 1
```

| 좌표



결과값

구해야 하는 값(Output): 필요한 지령이의 수

사고 과정

BFS를 써야하는 문제임을
이미 알고 있음.
어디에 적용하지?

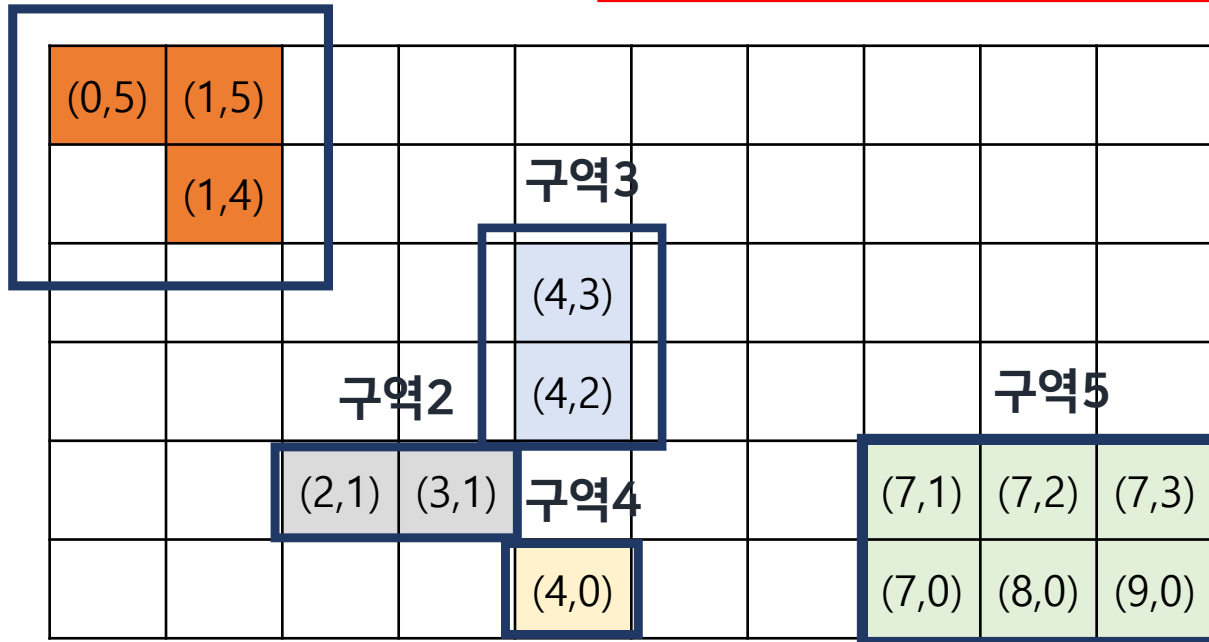


인접한 각각의 구역을 그래프
로 생각하고 BFS를 함.
한번 해당 그래프를 탐색하면
지령이수 += 1



각각의 구역에 BFS 하려면 모든
배추가 있는 모든 땅에 BFS를
하고, 이미 지나간 땅이면 pass

구역1



모든 배추가 있는 땅에 BFS를 하려면, 모든
배추가 있는 땅의 인접좌표 정보가 있어야 함

```
ex) adjacent_data = {  
    (0,5) : set([(1,5)]),  
    (1,4) : set([(1,5)]),  
    (1,5) : set([(0,5), (1,4)]),  
    ...  
}
```

┃ 코드 해석 - 테스트 케이스 INPUT 입력받기

```
TEST_CASE_COUNT = int(sys.stdin.readline()) # 테스트 케이스 횟수
```


코드 해석 - 가로/세로/배추개수 INPUT 입력받기

```
for _ in range(TEST_CASE_COUNT):  
    """  
    테스트 케이스 횟수 만큼 for문 반복  
    """  
  
    # 배추밭의 (가로), (세로), (심어진 배추의 개수)  
    WIDTH, HEIGHT, CABBAGE_COUNT = list(map(int, sys.stdin.readline().strip().split()))
```

코드 해석 - 심어진 배추들의 좌표 데이터

```
(0, 5): set()  
(1, 4): set()  
(1, 5): set()  
(2, 1): set()  
(3, 1): set()  
(4, 0): set()  
(4, 2): set()  
(4, 3): set()  
(7, 0): set()  
(7, 1): set()  
(8, 0): set()  
(8, 1): set()  
(9, 0): set()  
(9, 1): set()
```

set() 자료구조로 초기화한
이유는 마지막에 설명

```
for _ in range(TEST_CASE_COUNT):
```

(생략).....

```
    adjacent_data = dict()
```

```
    for _ in range(CABBAGE_COUNT):
```

```
        """
```

배추의 개수만큼 좌표IN

```
        """
```

```
        cabbage_coordinate =
```

```
        adjacent_data[cabbage_coordinate]
```

```
        cabbage_coordinate, ... ]), ... }
```

data에 (i,j):set() 형태로 추가

```
        data[(i,j):set()] = input().readline().strip().split()))
```

코드 해석 - 심어진 배추

```
for _ in range(TEST)
```

(생략).....

```
for coord in ad
```

```
"""
```

인접한 배추

```
"""
```

```
i, j = coord
```

```
if (i-1 >=
```

```
    adjacent
```

```
if (i+1 < W
```

```
    adjacent
```

```
if (j-1 >=
```

```
    adjacent
```

```
if (j < HEI
```

```
    adjacent_data[(1, j)].add((1, j+1))
```

(0, 5): {(1, 5)}

(1, 4): {(1, 5)}

(1, 5): {(1, 4), (0, 5)}

(2, 1): {(3, 1)}

(3, 1): {(2, 1)}

(4, 0): set()

(4, 2): {(4, 3)}

(4, 3): {(4, 2)}

(7, 0): {(7, 1), (8, 0)}

(7, 1): {(7, 0), (8, 1)}

(8, 0): {(9, 0), (7, 0), (8, 1)}

(8, 1): {(9, 1), (7, 1), (8, 0)}

(9, 0): {(9, 1), (8, 0)}

(9, 1): {(9, 0), (8, 1)}

왼쪽 좌표를 인접좌표로 추가

(1, 5): {(1, 4), (0, 5)}

(1,5)

(1,4)

and 심어져있는 배추이면

코드 해석 - 모든 배추가 있는 땅에 BFS를 하기 위해, for문 돌리는 코드

```
for _ in range(TEST_CASE_COUNT):  
    (생략).....  
    visited = set()  
    earth_worm = 0  
    for coord in range(10):  
        """  
        if 좌표가 배추인 경우  
        else 해치지 않음  
        """  
        queue = []  
        # 방문한 좌표 기록  
        # adjacents = []  
        if coord == 0:  
            continue
```

(0,5)	(1,5)								
	(1,4)								
				(4,3)					
				(4,2)					
		(2,1)	(3,1)				(7,1)	(7,2)	(7,3)
				(4,0)			(7,0)	(8,0)	(9,0)

-> 지렁이수 + 1

코드 해석 - 모든 배추가 있는 땅에 BFS를 하기 위해, for문 돌리는 코드

BFS -> 필요한 지렁이 수 += 1

```
for _ in range(TEST_CASE_COUNT):  
    (생략).....  
    visited = set()  
    earth_worm_count = 0  
    for coord in range(10):  
        """  
        if 좌표가 방문한 곳이면 continue  
        else 해킹 가능  
        """  
        queue = deque()  
        # 방문한 곳이면 continue  
        # adjacents = [(1,0), (-1,0), (0,1), (0,-1)]  
        if coord == (0,0):  
            queue.append(coord)  
            visited.add(coord)  
            earth_worm_count += 1  
            while queue:  
                cur = queue.popleft()  
                for adj in adjacents:  
                    next = (cur[0] + adj[0], cur[1] + adj[1])  
                    if next[0] < 0 or next[0] > 9 or next[1] < 0 or next[1] > 9:  
                        continue  
                    if next in visited:  
                        continue  
                    queue.append(next)  
                    visited.add(next)  
                    earth_worm_count += 1  
    print(earth_worm_count)
```

(0,5)	(1,5)								
	(1,4)								
				(4,3)					
				(4,2)					
		(2,1)	(3,1)				(7,1)	(7,2)	(7,3)
				(4,0)			(7,0)	(8,0)	(9,0)

-> 지렁이수 + 1

continue

코드 해석 - 모든 배추가 있는 땅에 BFS를 하기 위해, for문 돌리는 코드

이미 방문한 좌표이므로 continue



```
for _ in range(TEST_CASE_COUNT):
```

(생략).....

```
visited = set()
```

```
earth_worm = 0
```

```
for coord in range(10):
```

```
    """
```

```
    if 좌표(
```

```
    else 해
```

```
    """
```

```
    queue =
```

```
    # 방문한
```

```
    # adjac
```

```
    if coor
```

```
        continue
```

(0,5)	(1,5)								
	(1,4)								
				(4,3)					
				(4,2)					
		(2,1)	(3,1)				(7,1)	(7,2)	(7,3)
				(4,0)			(7,0)	(8,0)	(9,0)

-> 지렁이수 + 1

코드 해석 - 모든 배추가 있는 땅에 BFS를 하기 위해, for문 돌리는 코드

```
for _ in range(TEST_CASE_COUNT):
    (생략).....

    visited = set()
    earth_worm = [(0,5)]
    for coord in earth_worm:
        """
        if 좌표가 방문한 좌표이면
        else 해
        """
        queue = deque()
        queue.append(coord)
        # 방문한 좌표
        # adjac
        if coord in visited:
            continue
```

(0,5)	(1,5)								
	(1,4)								
				(4,3)					
				(4,2)					
		(2,1)	(3,1)				(7,1)	(7,2)	(7,3)
				(4,0)			(7,0)	(8,0)	(9,0)

-> 지렁이수 + 1

코드 해석 - 모든 배추가 있는 땅에 BFS를 하기 위해, for문 돌리는 코드

```
for _ in range(TEST_CASE_COUNT):  
    (생략).....  
    visited = set()  
    earth_worm = 0  
    for coord in range(10):  
        """  
        if 좌표 (coord, 0)에 배추가 있으면  
        else 해  
        """  
        queue = []  
        # 방문한 좌표 기록  
        # adjac  
        if coord == 0:  
            continue
```

(0,5)	(1,5)								
	(1,4)								
				(4,3)					
				(4,2)					
		(2,1)	(3,1)				(7,1)	(7,2)	(7,3)
				(4,0)			(7,0)	(8,0)	(9,0)

BFS / 필요한 지렁이 수 += 1

-> 지렁이수 + 1

코드 해석 - 모든 배추가 있는 땅에 BFS를 하기 위해, for문 돌리는 코드

```
for _ in range(TEST_CASE_COUNT):  
    (생략).....  
    visited = set()  
    earth_worm = 0  
    for coord in range(10):  
        """  
        if 좌표 (coord, 0) == (0, 5):  
            earth_worm += 1  
        else 해...  
        """  
        queue = deque()  
        # 방문한 좌표 기록  
        # adjac...  
        if coord == 0:  
            continue
```

(0,5)	(1,5)								
	(1,4)								
				(4,3)					
				(4,2)					
		(2,1)	(3,1)				(7,1)	(7,2)	(7,3)
				(4,0)			(7,0)	(8,0)	(9,0)

이미 방문한 좌표이므로 continue

-> 지렁이수 + 1

┃ 코드 해석 - 모든 배추가 있는 땅에 BFS를 하기 위해, for문 돌리는 코드

```
for _ in range(TEST_CASE_COUNT):
```

(생략).....

```
visited = set()          # 방문한 좌표  
earth_worm_count = 0     # 필요한 지렁이의 수
```

```
for coord in adjacent_data.keys():
```

```
    """
```

if 좌표(coord)가 이미 방문한 좌표이면 -> continue(다음 배추의 좌표로 이동)

else 해당 배추의 좌표(coord)대해 BFS(너비우선탐색) 시행 -> 방문한 좌표 추가 -> 지렁이수 + 1

```
    """
```

```
    queue = deque([coord])
```

방문한 좌표라면 지렁이 수를 카운트 하면 안되므로, 밑의 코드를 무시하고

adjacent_data의 다음 key값으로 이동하여 for문 진행

```
if coord in visited:
```

```
    continue
```

코드 해석 - BFS 코드

```
for _ in range(TEST CASE COUNT):
```

```
(생략)...
```

```
for coord in range(10):
```

```
(생략)
```

```
while True:
```

(0,5)	(1,5)								
	(1,4)								
				(4,3)					
				(4,2)					
		(2,1)	(3,1)			(7,1)	(7,2)	(7,3)	
				(4,0)		(7,0)	(8,0)	(9,0)	

```
# 현재
```

```
earth_worm_count += 1
```

```
print(earth_worm_count)
```

(1,5)의
인접 좌표

방문한 좌표

(1,4)

(0,5)

좌표) 의 차집합을 추가

adjacent_data의
자료구조로 사용한 이유
)는 차집합을 의미

길이 수 + 1

| 코드 해석 - BFS 코드

```
for _ in range(TEST_CASE_COUNT):  
    (생략).....  
    for coord in adjacent_data.keys():  
        (생략).....  
        while queue:  
            """  
            -BFS 코드  
            if queue에서 꺼낸노드(n)가 방문한 좌표가 아니면 ->방문한 좌표에 추가  
                queue에 (n의 인접 좌표) 와 (방문한 좌표) 의 차집합을 추가  
            """  
            n = queue.popleft()  
            if n not in visited:  
                visited.add(n)  
                queue += adjacent_data[n] - visited  
  
            # 하나의 연결된 그래프(인접해있는 구역)만큼 방문한 좌표에 추가되면 -> 지렁이 수 + 1  
            earth_worm_count += 1  
  
print(earth_worm_count)
```

끝