LandCount 문제를 통해 BFS 방법을 이해하는 날입니다.

주어진 문제 설명을 이해하고 전체 코드를 구현하는 문제 입니다.

[1] 문제 제목 : LandCount_BFS

파일명: LandCountBFS.py

- 먼저 스스로 구현하고 학습한 후 솔루션 파일을 참고하세요. 솔루션 파일명은 다음과 같습니다.

[1] 문제 제목 :: LandCount_BFS

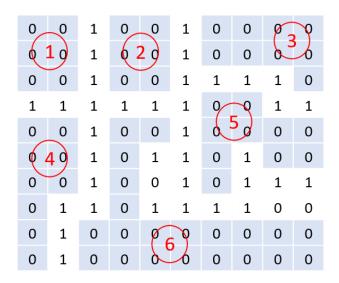
파일명: LandCountBFS_solution.py

[1] LandCount

[문제설명]

지구는 일부의 바다와 육지로 구성되어 있다. 다음 그림은 지구 표면의 일부를 나타낸 것이다. 바다는 1 로 표기되어 있고 육지는 0 으로 표기되어 있다. 다음 그림의 예에서는 1 로 표기된 바다를 제외하고 육지만으로 구성된 부분을 서로 다른 색으로 표시 해 보았다. 총 6 개의 육지가보이고 있다.

주어진 샘플에서 보여지는 육지는 정사각형 하나에서 가로 방향, 세로 방향 즉, 상하좌우 방향으로 움직일 수 있으면 같은 육지로 판단한다. 총 몇 개의 육지로 구성되어 있는지를 세는 프로그램을 작성한다.



육지를 찾기 위한 구현으로는 DFS 방법을 이용하여 구현합니다. DFS 구현 방법은 재귀 호출 또는 스택을 이용할 수 있습니다. 방법은 자유롭게 선택하여 구현하세요.

[입력값 설명]

2차원 배열로 정보가 주어지며 바다를 1으로 육지를 0로 표기된 값을 담은 data가 solution 매개 변수에 전달됩니다.

[출력값 설명]

바다를 두고 분리하며 상하좌우로 움직일 수 있는 길은 하나의 육지로 판단했을 때 주어진 표면에 총 몇 개의 육지가 있는지를 리턴 한다.

[입출력 예]

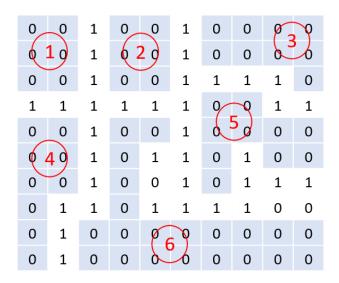
data	result
[[0, 0, 1, 0, 0, 1, 0, 0, 0, 0],	1
[0, 0, 1, 0, 0, 1, 0, 0, 0, 0],	
[0, 0, 1, 0, 0, 1, 1, 1, 1, 0],	
[0, 1, 1, 1, 1, 0, 0, 0, 1, 0],	
[1, 1, 1, 0, 1, 0, 0, 0, 1, 0],	
[0, 0, 0, 0, 1, 1, 0, 1, 1, 1],	
[0, 0, 0, 0, 0, 1, 0, 1, 0, 0],	
[0, 0, 0, 0, 0, 1, 0, 1, 0, 0],	
[0, 0, 0, 0, 0, 1, 1, 1, 0, 0],	
[0, 0, 0, 0, 0, 0, 1, 0, 0, 0]]	
[[1, 0, 1, 0, 1],	8
[0, 0, 0, 0, 0],	
[1, 0, 1, 0, 1],	
[0, 0, 0, 0, 0],	
[1, 0, 1, 1, 1]]	
기타 코드에 제공된 자료 참고	

[1] LandCount

[문제설명]

지구는 일부의 바다와 육지로 구성되어 있다. 다음 그림은 지구 표면의 일부를 나타낸 것이다. 바다는 1 로 표기되어 있고 육지는 0 으로 표기되어 있다. 다음 그림의 예에서는 1 로 표기된 바다를 제외하고 육지만으로 구성된 부분을 서로 다른 색으로 표시 해 보았다. 총 6 개의 육지가보이고 있다.

주어진 샘플에서 보여지는 육지는 정사각형 하나에서 가로 방향, 세로 방향 즉, 상하좌우 방향으로 움직일 수 있으면 같은 육지로 판단한다. 총 몇 개의 육지로 구성되어 있는지를 세는 프로그램을 작성한다.



육지를 찾기 위한 구현으로는 BFS 방법을 이용하여 구현합니다. BFS 구현 방법은 큐 자료구조의 특징을 이용하여 순회하는 방법입니다..

[입력값 설명]

2차원 배열로 정보가 주어지며 바다를 1으로 육지를 0로 표기된 값을 담은 data 가 solution 매개 변수에 전달됩니다.

[출력값 설명]

바다를 두고 분리하며 상하좌우로 움직일 수 있는 길은 하나의 육지로 판단했을 때 주어진 표면에 총 몇 개의 육지가 있는지를 리턴 한다.

[입출력 예]

data	result
[[0, 0, 1, 0, 0, 1, 0, 0, 0, 0],	6
[0, 0, 1, 0, 0, 1, 0, 0, 0, 0],	
[0, 0, 1, 0, 0, 1, 1, 1, 1, 0],	
[1, 1, 1, 1, 1, 1, 0, 0, 1, 1],	
[0, 0, 1, 0, 0, 1, 0, 0, 0, 0],	
[0, 0, 1, 0, 1, 1, 0, 1, 0, 0],	
[0, 0, 1, 0, 0, 1, 0, 1, 1, 1],	
[0, 1, 1, 0, 1, 1, 1, 1, 0, 0],	
[0, 1, 0, 0, 0, 0, 0, 0, 0],	
[0, 1, 0, 0, 0, 0, 0, 0, 0]]	
[[1, 0, 1, 0, 1],	1
[0, 0, 0, 0, 0],	
[1, 0, 1, 0, 1],	
[0, 0, 0, 0, 0],	
[1, 0, 1, 1, 1]]	
기타 코드에 제공된 자료 참고	