**< 알고리즘 문제 출제 보고서 >**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **난이도 : A단계**  **문제 제목: 준영이의 하수구 뚫기**  **문제 지문**  준영이가 오랜만에 집에서 때를 밀었더니 하수구가 막혀 물이 역류하기 시작하였다. 아뿔싸, 그녀가 도착하기 전 얼른 하수구를 뚫어 이를 해결해야 한다! 준영이는 빠르게 하수구를 뚫기 위해 아래 그림과 같은 모양의 하수구 도면을 얻었다. 해당 도면을 바탕으로 준영이는 하수구를 한 칸씩 이동하여 막힌 하수구를 끝까지 뚫어야 한다. 과연, 준영이는 하수구를 완전히 뚫기 위해 최소 몇 번을 시도해야 하는지 알아보자.   |  |  |  |  | | --- | --- | --- | --- | | **(출발)**  0 | 7 | 7 | 2 | | 5 | 8 | 7 | 5 | | **-1** | 1 | 7 | 4 | | 9 | 0 | 4 | 0  **(도착)** |   하수구의 시작점은 **(0, 0)**으로 도착점은 **(N-1, N-1)**이다. 해당 도면을 보면 하수구 각 구간의 막힌 정도가 표시되어 있다. 위의 그림에서 **(1, 0)** 지점에서는 **'5'**만큼 막혀있으므로 해당 칸을 뚫기 위해서는 5번의 시도 횟수가 발생한다. 보조 아이템인 락스가 존재하는 지점 **(2,1)**에 도착하게 된다면 1씩 힘의 크기가 증가한다. 즉, 해당 칸에 도착하게 된다면 이후의 칸들에서는 힘의 크기가 **‘2’**가 되는 것이다. 증가한 힘에 따라, **(3, 3)**을 지나기 위해서는 ‘4번’이 아닌 ‘2번’의 시도만 하면 된다. 이런 방식으로 탐색을 하면, 준영이는 위 도면의 하수구를 뚫기 위해 총 ‘8번’만 시도하면 된다.  **[제한사항]**   1. 시작점은 (0, 0)으로 도착점은 (N-1, N-1)로 고정한다. ( 0 < N ≤ 100 ) 2. 한번에 뚫을 수 있는 양은 1이다. (단, 락스 아이템 장착 후에는 해당 양만큼 추가된 양으로 작업이 가능하다.) 3. 락스 아이템은 ‘-1’로 표현된다. 해당 칸에 도착하면 힘의 양이 ‘+1’만큼 증가한다. 4. 이동 방향은 상, 하, 좌. 우이다. 단, 지나온 길을 다시 되돌아갈 수 없다.   **[입력]**  첫째 줄에 테스트 케이스 개수인 t가 주어진다. 다음 줄에는 행렬의 크기인 N이 주어진다. 지도의 크기는 최대 100 \* 100 이다. 다음 줄부터는 지도의 크기만큼 2차원 배열 형태의 지도 정보가 주어진다.  **[출력]**  각 테스트 케이스의 답을 순서대로 출력하며, 각 케이스마다 줄의 시작에 #C 를 출력하여야 한다. 이때, C는 케이스의 번호이다. 같은 줄에 빈 칸을 하나 두고, 주어진 입력에서 출발지에서 도착지까지 가는 경로 중에 복구 작업에 드는 가장 적은 시도 횟수를 출력하시오.   |  |  | | --- | --- | | **Input** | **Output** | | 5  4  0 7 7 -1  5 8 7 5  2 1 7 4  9 0 4 0  5  0 2 6 6 8  8 3 1 2 9  9 9 4 3 6  6 6 8 2 2  2 -1 0 7 0  5  0 2 2 5 7  8 5 3 0 2  1 0 3 7 3  2 2 1 1 4  9 9 6 9 0  5  0 9 7 3 4  4 1 2 -1 0  2 6 8 7 9  0 3 1 5 9  0 6 -1 0 0  4  0 4 8 2  6 2 5 6  3 9 3 2  -1 1 -1 0 | 5  4  0 7 7 -1  5 8 7 5  2 1 7 4  9 0 4 0  5  0 2 6 6 8  8 3 1 2 9  9 9 4 3 6  6 6 8 2 2  2 -1 0 7 0  5  0 2 2 5 7  8 5 3 0 2  1 0 3 7 3  2 2 1 1 4  9 9 6 9 0  5  0 9 7 3 4  4 1 2 -1 0  2 6 8 7 9  0 3 1 5 9  0 6 -1 0 0  4  0 4 8 2  6 2 5 6  3 9 3 2  -1 1 -1 0 | |