

Assignment #1

Recursive Maze Implementation

과제 설명

미로의 정보가 입력으로 주어질 때, 출발지에서 목적지까지 미로를 빠져나갈 수 있는 경로를 찾는 프로그램을 재귀함수를 사용하여 구현하시오.

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

maze배열 예시

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

미로 경로 예시

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

path 배열 예시

1. 출발점은 항상 왼쪽 상단의 1행1열(maze[0][0])의 칸이어야 한다.
2. 도착점은 항상 우측 하단의 N행 N열(maze[N-1][N-1])의 칸이다.
3. maze배열의 각 칸은 지도에서 대응되는 해당 칸의 통행가능 여부를 나타낸다.
 - A. 0은 이동 가능(passable)하다는 뜻이며 1은 이동할 수 없는 칸임을 의미한다.

주어진 maze배열을 사용하여 미로의 경로를 탐색한 후, 이 경로를 path배열에 저장해야 한다. path배열은 자신이 미로 탈출을 위해 사용해야 할 경로 정보를 0과 1로 표시한다. 1은 경로를 나타내며 0은 경로가 아닌 칸을 의미한다.

최종적으로 save_result(path, path exist, N);를 통해 출력된 경로를 제출한 답안으로 인정한다.

과제 목표

아래 두 함수를 완성하여 재귀적인 방법으로 미로의 경로를 찾는 프로그램을 완성하시오.

1. `bool isSafe(Map maze, int r, int c, Map path, int N);`
A. maze지도의 r행 c열의 칸이 이동가능한 칸 인지 판단하는 함수
2. `bool solveMazeUtil(Map maze, int r, int c, Map path, int N, int length);`
A. 현재(r행, c열)칸에서 도착지까지 도달하는 경로가 있는지 판단하는 재귀 함수.
B. 내부에서 각 칸의 방문 가능 여부를 조사할 때에는 반드시 `isSafe()` 함수를 사용하여야 함.

과제 유의사항

아래의 사항을 꼭 유의하며 과제를 진행하세요.

1. `Map` 타입은 `int**` 타입과 같습니다. (헤더파일 내에 정의되어 있음.)
2. `Source.h` 헤더파일의 경우 ‘`DATA_FOLDER`’ 상수 이외에는 절대로 수정하지 마세요. 해당 상수 이외의 헤더파일을 수정한 정답은 인정하지 않습니다.
A. ‘`DATA_FOLDER`’ 상수를 수정하여 각 폴더의 데이터를 실행해볼 수 있습니다.
B. 기본적으로 `data_sample`, `data_test_01`, `data_test_02`, `test_data_03`, `test_data_04` 네 폴더가 제공됩니다.
C. 데이터 폴더를 직접 만들어도 무관합니다. 스스로 많은 데이터를 만들어 충분히 테스트 해보기를 권장합니다.
3. 소스코드 중간에 존재하는 `time_check();` 함수를 지우거나 이동하지 마세요. 실행 시간이 10초가 초과하는 경우 감점될 수 있습니다.
A. 실제 점수 채점은 학생들에게 주어지지 않은 별도의 데이터로 진행됩니다.
B. 이에 대한 이의제기는 받지 않습니다. 어떤 입력이 주어져도 실행되는 프로그램을 작성해야 합니다.
4. 미로의 경로를 찾으면 됩니다. 항상 최적의 경로를 찾을 필요는 없습니다.
A. 최적의 경로만을 찾아 출력하는 경우 가산점이 주어질 수 있습니다.
B. 불가능한 경로를 출력하는 경우 오답으로 판단합니다.
5. `isSafe()` 함수와 `solveMazeUtil()` 함수 내부를 자유롭게 수정하여도 무관하지만, 함수의 반환 타입이나 파라미터를 수정하지 마세요.
A. 주어진 함수의 정의에 따라 작성해야 합니다.
B. 각 함수의 의미와 역할은 하단의 설명과 코드의 주석을 참고하세요.
6. 과제 카피의 경우 이유를 불문하고 0점 처리 함. (서로 다른 반 포함)

입/출력 예시

작성한 코드가 정상적으로 수행되었을 시, 아래와 같은 출력(오른쪽)이 `output.txt`에 나타납니다.

1	10	1	PATH EXIST
2	0 0 0 1 0 0 0 0 0 0	2	PATH LENGTH : 25
3	0 1 0 0 0 0 0 1 0 0	3	
4	0 0 0 1 1 1 0 1 1 1	4	0..#.....
5	1 0 1 0 0 0 0 1 0 0	5	0#.....#..
6	0 0 1 1 0 1 0 0 0 0	6	00.###.###
7	0 1 0 0 0 1 0 0 1 0	7	#0#....#..
8	0 0 1 1 0 1 1 1 1 1	8	00##.#....
9	1 0 0 1 0 1 0 0 0 0	9	0#...#...#.
10	0 1 0 1 0 0 0 1 0 0	10	00##.#####
11	0 0 0 0 0 1 0 0 1 0	11	#00#.000.
12		12	.#0#000#00
13		13	..000#..#0

- 1. ‘0’는 출발지부터 목적지까지 이용해야 할 칸을 의미합니다.
- 2. ‘#’은 벽을 의미합니다.
- 3. ‘.’은 방문하지 않은 칸을 의미합니다.
- 4. ‘x’는 방문할 수 없으나, 출력 답안에는 방문한 것으로 처리 된 칸을 의미합니다.

불가능한 경로를 출력한 경우 `output.txt` 파일 최하단에 에러 메시지가 출력됩니다.

함수 설명

Source.c 파일

함수의 반환 타입이나 파라미터 구조를 변경하면 안됩니다. 함수 내부만을 수정하여 과제를 수행해주세요.

필요한 경우 함수와 전역 변수는 자유롭게 선언할 수 있습니다.

main()함수의 경우 전체적인 틀은 유지하되, 일부 수정하는 것은 가능합니다. (새 변수 선언 및 사용 등)

1. isSafe() 함수

함수	<code>bool isSafe(Map maze, int r, int c, Map path, int N);</code>	
정의	미로의 r 행 c 열 칸을 현재 방문할 수 있는지 검사하는 함수입니다.	
TODO	효율적인 탐색이 가능하도록 방문 여부를 정할 수 있게 수정되어야 합니다.	
파라미터	Map maze	미로의 정보를 저장하는 2 차원 정수 배열입니다.
	int r	검사하고자 하는 행의 번호를 나타내는 정수입니다.
	int c	검사하고자 하는 열의 번호를 나타내는 정수입니다.
	Map path	현재까지 방문해온 경로정보를 저장하는 2 차원 정수 배열입니다.
	int N	미로의 행과 열의 수를 나타내는 정수입니다.
반환	bool	해당 칸을 현재 방문할 수 있다면 true, 그렇지 않다면 false

2. solveMazeUtil() 함수

함수	<code>bool solveMazeUtil(Map maze, int r, int c, Map path, int N, int length)</code>	
정의	현재 칸(maze[r][c])에서 도착지로 가는 경로가 존재하는지 검사/탐색하는 재귀 함수입니다.	
TODO	경로가 존재하는 미로에 대하여 올바르게 경로를 찾을 수 있도록 수정되어야 합니다. 재귀 함수 방식으로 구현되어야만 정답으로 인정합니다. + 최적의 경로를 찾는 방식을 구현하면 가산점이 주어질 수 있습니다.	
파라미터	Map maze	미로의 정보를 저장하는 2 차원 정수 배열입니다.
	int r	현재 탐색을 시작하려는 칸의 행의 번호를 나타내는 정수입니다.
	int c	현재 탐색을 시작하려는 칸의 열의 번호를 나타내는 정수입니다.
	Map path	현재까지 방문해온 경로정보를 저장하는 2 차원 정수 배열입니다.
	int N	미로의 행과 열의 수를 나타내는 정수입니다.
	int length	현재까지 방문한 칸의 수를 나타냅니다.
반환	bool	현재 칸(maze[r][c])부터 도착지까지 경로가 존재한다면 true 를 반환합니다. 아무 경로도 존재하지 않는다면 false 를 반환합니다.

Source.h 파일

이 파일에 있는 함수는 절대로 수정하지 마세요. 처음 과제 첨부파일에 주어진 원본 헤더파일을 사용하여 채점을 진행하므로, 이 헤더 파일을 수정해서 발생하는 불이익은 모두 학생에게 책임이 있습니다.

물론 본인의 소스코드 내에서 자유롭게 사용하는 것은 상관없습니다.

함수	<code>void time_check();</code>
정의	현재 프로그램의 수행 시간을 검사합니다. 시간 제한(10 초)을 초과했다면 에러와 함께 종료합니다.

함수	<code>void init_map(Map* map, int N);</code>
정의	2 차원 배열(Map)의 포인터를 받아 새로운 2 차원 배열을 동적으로 할당합니다. 2 차원 배열의 크기는 N 행 N 열입니다.

함수	<code>void copy_map(Map src, Map dest, int N);</code>
정의	2 차원 배열 src 의 모든 원소를 똑같이 dest 배열에 복제합니다. (덮어씁니다.) 배열의 크기는 모두 N 행 N 열로 동일해야 합니다.

함수	<code>void copy_map(Map src, Map dest, int N);</code>
정의	2 차원 배열 src 의 모든 원소를 똑같이 dest 배열에 복제합니다. (덮어씁니다.) 배열의 크기는 모두 N 행 N 열로 동일해야 합니다.

함수	<code>void init_and_input_maze(Map* maze, Map* path, int* N);</code>
정의	데이터 폴더에서 input.txt 를 읽어와 두 2 차원 배열 maze 와 path 를 초기화합니다. 가장 첫 줄에서 미로의 크기를 읽어 *N 에 저장합니다. 이후 maze 와 path 에 N 행 N 열 2 차원 배열을 동적할당한 후 모두 0 으로 초기화합니다.

함수	<code>void release_map(Map map, int N);</code>
정의	N 행 N 열의 2 차원 배열 map 을 할당해제합니다.

함수	<code>void save_result(Map path, bool path_exist, int N);</code>	
정의	N 행 N 열의 2 차원 배열 map 을 할당 해제합니다.	
파라미터	Map path	출발점부터 도착지까지의 경로를 0 과 1 로 표현한 2 차원 배열입니다. 방문한 칸은 1 로, 방문하지 않은 칸은 0 으로 표시되어야 합니다.
	bool path_exist	출발지에서 목적지로 가는 경로가 존재하는지 여부를 true/false 로 전달합니다.
	int N	미로의 행과 열의 수를 나타내는 정수입니다