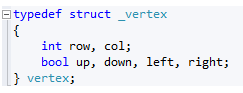
프로젝트 PRJ-2 미로(MAZE) 3주차 결과보고서

전공 : 컴퓨터공학 학년 : 2 학번 : 20141579 이름 : 정연준

1. **실습 및 숙제로 작성한 프로그램의 알고리즘과 자료구조를 요약하여 기술한다. 완성한 알고리즘의 시간 및 공간 복잡도를 보이고 실험 전에 생각한 방법과 어떻게 다른지 아울러 기술한다.**

***자료구조***

이번 과제를 해결하기 위해 사용한 자료구조는 다음과 같다. *row, col*은 각각 해당 칸의 x와 y좌표를 나타내며, 4개의 bool형 flag는 각각 위, 아래, 왼쪽, 오른쪽에 벽이 있는지의 여부를 나타낸다. 완성 전의코드와 달라진 점은 위쪽과 왼쪽의 벽의 존재 여부를 나타내주는 *up, left*변수를 추가해 주었다는 것 이다. 기존의 오른쪽과 아래쪽의 벽의 여부를 알려주는 두 변수만 사용하여 코드를 작성하면, 왼쪽과 위 쪽의 벽의 여부를 알기 위해 왼쪽 방의 오른쪽 벽의 여부, 위쪽 방의 아래쪽 벽의 여부를 알아보아야 하였기 때문에, 코드 작성에 있어 번거러움을 줄이기 위하여 변수를 추가하게 되었다. 

***알고리즘***

알고리즘은 예비 보고서에서 작성한 바와 같다. DFS, BFS의 경우 모두 시간복잡도는 다음과 같다.

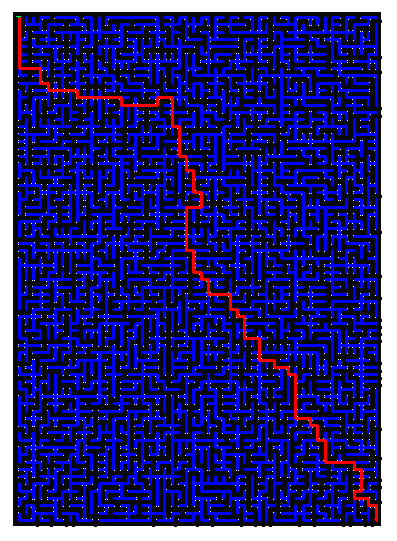
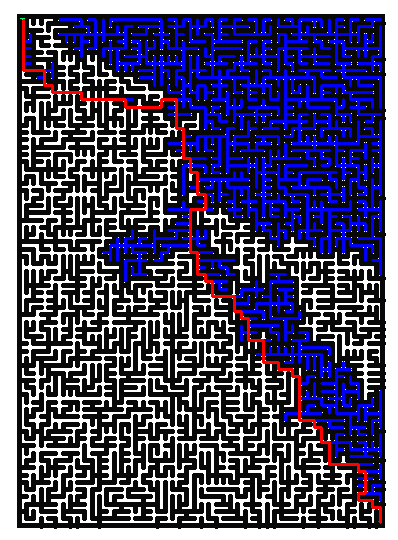
O(1+b+b^2+...+b^d) = O(b^d)

**2. 자신이 설계한 프로그램을 실행하여 보고 DFS, BFS 알고리즘을 서로 비교한다. 각각의 알고리즘은 어떤 장단점을 가지고 있는지, 자신의 자료구조에는 어떤 알고리즘이 더 적합한지 등에 대해 관찰하고 설명한다.**

미로를 탐색하기 위하여 DFS와 BFS 알고리즘을 사용하였다. 각각의 알고리즘에 대한 설명은 예비보고서의 내용으로 대체한다.

DFS와 BFS의 장, 단점은 다음과 같다. DFS의 경우, 자신이 정한 우선순위에 따라 한 쪽 방향을 택하면, 그 방향을 택하였을때 갈 수 있는 모든 방을 방문한 후, 그 다음 우선순위를 가진 방향으로 탐색을 하기 때문에, 목적지의 위치에 따라 적은 탐색으로 길을 찾을 수도, 최악의 경우 모든 방을 한 번씩 방문한 후에야 원하는 길을 찾을 수도 있다. 반면 BFS의 경우, 매 탐색 시 같은 깊이에 있는 모든 노드들을 방문한 후에, 그 다음 깊이로 넘어가기 때문에 우선순위에 상관없이 균일한 시간에 원하는 길을 찾을 수 있다. 이러한 두 알고리즘의 탐색 방법의 차이는, 순환하는 길이 존재하는 불완전한 미로에서 다음과 같은 차이를 불러온다. 완전한 미로와 달리 불완전한 미로에서는, 목적지에 도달하는 길이 한 가지 이상일 수 있기 때문에, 목적지에 도달할 수 있는 최단 경로를 찾는 것이 중요하다. 이 때, DFS의 경우 사용자가 지정한 우선순위에 따라 찾은 길이 다를 수 있기 때문에, 찾은 길이 최적의(최단의) 길인지의 여부를 알기 위해서는, 존재하는 모든 길을 찾은 후에 비교를 통해서야 비로소 알 수 있다. 하지만, BFS의 경우, 탐색 중인 깊이에 존재하는 모든 노드를 방문하고서야 다음 깊이로 넘어가기 때문에, BFS를 통해 길을 찾았다면, 그 길은 최적의 길 임이 항상 보장되어있다. 그렇기에 불완전한 미로의 경우 BFS를 사용하여 길을 찾는 것이 DFS를 사용하는 것 보다 적합하다.

하지만, 본 실험에서 인풋으로 주어지는 모든 미로는 완전미로임을 가정한다. 따라서, 특정한 두 방을 이어주는 길은 항상 하나이며, 따라서 *최적의* 길을 찾을 필요가 없기 때문에, DFS를 사용하여 길을 찾는 것이 BFS를 사용하는 것 보다 적합하다. 이는, 다음의 예시에서 알 수 있다.



*DFS BFS*

다음은 좌측 최상단의 방에서, 우측 최하단의 방 까지 이어주는 길을 찾기 위하여 DFS, BFS를 이용한 예시이다. 파란색 길은 길을 찾기위해 방문한 모든 방들을 표시해주며, 빨간색 길은 각 알고리즘을 사용하여 찾은 길을 나타낸다. 그림에서도 알 수 있듯이 DFS를 사용할 때 훨씬 적은 방을 탐색하고 길을 찾을 수 있다는 것을 알 수 있다.

따라서, 완전미로의 경우 DFS를 사용하는 것이 BFS를 사용하는 것 보다 적합하다는 것을 알 수 있다.