14주차 결과보고서

전공 : 컴퓨터공학 학년 : 2 학번 : 20151616 이름 : 최승환

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

typedef struct \_node

{

int visit;

int right;

int down;

int left;

int up;

} room;

이번 과제를 해결하기 위해 사용한 자료구조는 다음과 같다. 이에서 각 변수 right,down,left,up은 각각 오른쪽 아래,왼쪽 위에 벽이 존재하는지를 표시해주는 변수이다. 추가된 변수로는 visit 변수가 있다. visit 변수는 해당하는 칸을 방문했는지를 표시하는 변수이다. 하지만 숙제의 경우 BFS로 경로를 찾는 데에는 문제가 없음에도 불구하고, BFS로 방문한 위치를 그림으로 그릴 때 문제가 발생하게 된다. 이는 칸을 방문하는 순서 때문이다. DFS의 경우 자신이 방문할 노드의 전칸에서부터 길이 막힐때까지 움직인다. 하지만 BFS의 경우는 이런 방식을 이용하면 연속된 칸이 아닌 거리가 멀고 서로 연결되지 않은 칸으로 움직일 수 있기 때문에 대각선이 그려지는 등 DrawBuffered 함수에서 문제가 발생한다. 이에 따라서 방문한 경우 이를 미로와 같은 크기의 배열에 그려주어 이 정보를 이용해 그림을 그리도록 해야 한다. 이 때문에 선언한 것이 visit 배열이다. visit배열은 미로와 같이 m,n의 크기를 가지고 있고, 이 배열은 순전히 해당 칸이 방문된 적이 있었는지만 확인한다. 후에, 이 배열을 이용해 그림을 그리게 되며, 노드를 방문한 순서와 상관 없이 인접한 칸의 모양에 따라 그림을 그리기 때문에 앞서 말한 대각선을 그리는 등의 문제가 발생하지 않는다.

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

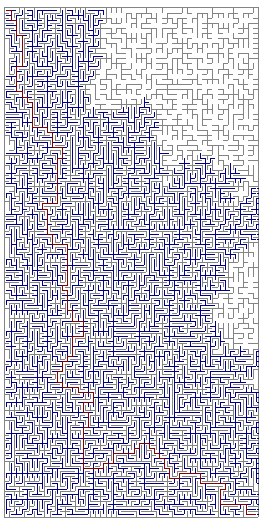
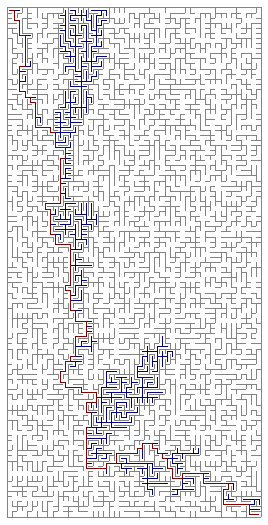
O(e)(e는 방문한 노드의 개수) 따라서 최악의 케이스에는 O(m\*n)의 시간복잡도를 가지게 된다. 더 정확히 계산하자면 BFS는 항상 인접한 네 칸을 방문하고 DFS는 인접한 네 칸 중 하나씩만 비교하기 때문에 한 번 돌 때 BFS가 4배로 많이 돌 수 있지만 이는 소수의 경우에 해당하고, 이마저도 상수배이기 때문에 시간복잡도는 O(e)로 동일하다.

**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.