14주차 예비보고서

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

1. DFS와 BFS의 시간 복잡도를 계산하고 그 과정을 설명한다.

int dx[4] = {1,0,-1,0};

int dy[4] = {0,1,0,-1};

for(i=0;i<4;++i)

{

search(현 x좌표+dx[i],현 y좌표+dx[i])

}

이전 예비보고서에 기재했던 것처럼 한 좌표의 노드에 대해서 상 하 좌 우 이 네가지에 대한 확인을 해야 한다. 따라서 m\*n 행렬에서 DFS 시간복잡도는 4\*e이므로 O(e) 이다. 이 때 e는 미로를 빠져나가는데 걸린 정점의 개수이다. 이 정점의 개수는 같은 크기의 미로라도 다를 수 있으며 최악의 경우는 O(m\*n)이라고 할 수 있다. BFS의 경우도 최악의 경우 시간복잡도가 O(m\*n)으로 같다. 하지만 최악의 경우가 아니라면 BFS와 DFS는 다른 시간이 걸릴 수 있다. 이 경우에 BFS의 시간복잡도도 4\*e로 시간복잡도는 O(e)로 같지만 e의 크기가 DFS를 쓰는 경우와 다를 수 있다.

2. 자신이 구현한 자료구조 상에서 DFS와 BFS 방법으로 실제 경로를 어떻게 찾는지 설명한다. 특히 DFS 알고리즘을 iterative한 방법으로 구현하기 위한 방법을 생각해보고 제시한다.

1번에서 언급한 search()를 사용하면 상 하 좌 우 중에 벽이 없는 곳의 노드로 접근하게 된다. DFS의 경우 그 노드로 이동한 후 그 노드에서 다시 search를 실행한다. 이를 재귀적으로 반복하도록 하고, 막다른 곳에 도착한 경우 0을 리턴하도록 하면 모든 경우의 수를 다 실행하도록 만들 수 있다. BFS의 경우 그 노드로 이동한 후 이 결과를 queue에 푸시하게 된다. 하지만 푸시를 하는데 있어서, 노드로 이동하기 전에 현 노드에서 접근할 수 있는 방을 방문했을 경우들에 대해서 먼저 푸시를 해준다.

int queue[MAX], front = -1,rear = -1;

다음과 같은 queue를 먼저 선언해 준 후 이 queue를 일반적인 queue와 같이 사용해주면 된다. BFS의 경우 DFS와 다른 점이 있다면 불완전 미로에 대해서도 해법을 찾을 수 있다는 것이다. BFS의 경우 위처럼 queue를 선택하고 푸시와 팝을 반복하며 각 경우의 수를 먼저 판별해준다. 따라서 DFS와 같이 순환구간에서 무한루프를 도는 것이 아니라, 순환 구간을 돌고 있는 경우와 순환미로가 아닌 곳을 가고 있는 경우를 동시에 진행하며 먼저 도착지에 도착한 경우를 답으로 따르게 된다.