13주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20191619 이름: 이동석

**1. DFS**

그래프의 연결형태를 표현하는 방법에는 두 가지가 있다. 하나는 인접행렬이며, 다른 하나는 인접리스트이다. 어떤 방법으로 그래프를 설정하는지에 따라 시간복잡도가 달라진다. 우선, 인접 행렬은 보통 adj[i][j] 형태로 표현하며, 이때 i -> j로 가는 간선이 존재할 경우 간선사이의 가중치 값을 할당받으며 가중치가 따로 없을 경우 1로 설정한다. 반면, 인접 리스트의 경우에는 adj[i]로 표현한다. 예를 들어 정점 i와 j과 연결되어있을 경우 연결리스트(linked list)를 통해 연결해주는 방식이다. 즉, adj[i]에는 j의 주소가 저장되어 있다. j가 또 k와 연결되어 있다면 i의 경우 i->j->k 가 된다. 또한, 방향 그래프와 무방향 그래프가 존재하는데, 미로탐색에서는 방향이 존재한다고 생각하고 진행하도록 하겠다.

일반적으로 DFS의 경우 재귀나 stack을 이용하여 구현한다. 인접행렬을 이용한 DFS의 경우 시간복잡도는 간선의 수와 상관없다. 인접 행렬의 column과 row는 정점의 수를 V라 할 때, 정점 수 V가 된다. 따라서, 정점을 모두 탐색할 경우 V개의 정점마다 V번의 간선 유무를 확인한다. 따라서 시간복잡도가 O(V\*V) 가 된다.

반면, 인접 리스트로 구현하게 될 경우를 생각해보자. 하나의 정점을 고를 경우, 인접 리스트의 특성상 그 정점과 연결되어 있는 간선의 개수만큼 탐색한다. 즉, 전체 시간복잡도가 O(V+E)가 된다. E는 간선의 개수이다.

DFS를 통해 미로를 탐색하기 위해서 stack라이브러리의 stack과 방문여부를 체크하는 2차원 포인터 배열 visited을 사용했다. 재귀가 아닌, 반복문을 사용해 구현하는 방법은 다음과 같다. Stack이 비어있지 않다면 다음 명령을 계속 반복한다.

1. Stack을 통해 방문하지 않은 정점(방)이 있다면 push 한다.

2. 현재 정점에서 이동할 수 없다면 pop을 한다.

3. 도착했다면, break 한다.

visited는 텍스트파일에서의 높이\*너비 만큼의 동적할당을 해준다. 방과 방사이를 연결하는 통로는 간선으로 생각해 준다. 방문할 경우 visited에 표시한다. 시작점은 visited 배열 상(1,1)이 되며, 도착점은 visited배열의 (row -2, column -2)가 된다. 각 정점들은 visited 상 (j,i) 로 생각이 가능하다.

**2. BFS**

BFS의 시간복잡도는 DFS 경우와 동일하다. 또한, 구현은 보통 queue를 사용한다. 경로를 탐색하기 위해서, queue 라이브러리 queue와 DFS와 같은 visited 2차원 포인터배열을 사용한다. 반복문을 사용해 구현을 하게되면 다음과 같다. queue가 비어있지 않다면, 다음명령을 계속 반복한다.

1. 현재 queue에 들어있는 값을 pop한다.
2. 현재 정점을 기준으로 방문하지 않은 정점을 모두 방문하며, 차례대로 push한다.
3. 도착하면break 한다.

이때, DFS와 다르게 BFS의 경우 실제 경로를 따로 저장해주어야 한다. 혹은, 자신의 정점의 부모 정점을 알아야 한다. 즉, 방향이 있다고 생각한 후, 구현을 하며 도착점에서 부터 거슬러 올라가며 경로를 찾아야한다.