13주차 예비보고서

전공: 컴퓨터공학과 학년: 4학년 학번: 20212022 이름: 이예준

**1.**

**- DFS**

dfs는 그래프의 깊이를 우선적으로 탐색하는 알고리즘으로 한 노드를 시작으로 끝까지

내려간 다음에 더 이상 갈 수 없으면 되돌아와 다른 경로를 탐색한다. 일반적으로 stack 구조를

사용하거나 재귀 호출을 이용하여 구현한다.

**- BFS**

bfs는 그래프의 너비를 우선적으로 탐색하는 알고리즘으로 한 노드를 시작으로 인접한

모든 노드들을 탐색한 뒤 그 다음으로 인접한 노드들을 차례대로 탐색하는 방법이다.

DFS와 BFS의 시간 복잡도는 구현 방법에 따라서 달라진다.

1. 그래프가 인접 행렬일 경우

모든 노드를 한 번씩 방문하면서 각 노드에서 모든 다른 노드에 대해 간선이 있는지

확인하기 때문에 시간 복잡도는 O(V2)이다.

\* V = 노드의 수

2. 그래프가 인접 리스트일 경우

모든 노드를 한 번씩 방문하면서 각 노드에서 연결된 모든 간선을 탐색하므로

시간 복잡도는 O(V + E)이다.

\* V = 노드의 수, E = 간선의 수

**2.**

우선, DFS 알고리즘을 iterative한 방법으로 구현하기 위해 stack을 활용할 것이다.

미로의 출발점과 도착점을 설정하고, 이를 기록한다. 경로를 찾기 위해 이전 실습에서 사용한

인접 행렬(node)을 사용할 것이다. 첫 번째로, 시작점을 스택에 넣는다.

시작점을 스택에 넣으면 해당 지점을 방문했음을 표시하기 위해 visited 배열을 true로 설정한다.

스택이 비어 있지 않은 동안 반복문을 실행한다. 현재 위치에서 아직 방문하지 않은 인접한

노드가 있는지 확인합니다. 인접 노드 중 방문하지 않은 노드가 있다면, 그 노드로 이동한다.

이동한 노드를 스택에 넣고, 해당 노드를 방문했음을 표시한다.

만약 현재 위치에서 모든 방향의 노드를 이미 방문했다면, 스택에서 이전 위치로 되돌아간다

(pop 연산). 이 과정을 반복하면, 스택이 비어질 때까지 계속해서 경로를 탐색할 수 있다.

BFS 알고리즘은 큐를 활용하여 작동한다. 먼저, 큐가 비어 있지 않은 동안 인접한 노드를

방문하고, 방문한 노드를 표시하고 큐에 넣는다. 이 과정에서 중복된 경로를 고려하지 않는다.

DFS와는 다르게, 방문한 노드는 다시 방문하지 않는다. 큐가 비어질 때까지 이 과정을 반복하며,

인접한 노드가 없다면 큐의 맨 앞에서 노드를 꺼낸다. 방문한 노드들은 계속해서 큐에 삽입하여

이 과정을 반복하여 최단 경로를 찾을 수 있다.