13주차 결과보고서

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

**1.**

**< DFS >**

텍스트, 폰트, 스크린샷, 대수학이(가) 표시된 사진

자동 생성된 설명

자료구조 - maze\* ExitMaze : 미로 탐색을 종료하기 위한 미로의 출구를 저장한다.

maze\* curidx : 미로 탐색 시작을 위한 시작점을 저장한다.

stack<maze\*> MazeStack : DFS 탐색을 위한 스택을 선언한다.

알고리즘 - 초기화 : 모든 노드를 방문하지 않은 상태로 초기화한다.

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

알고리즘 - 시작점을 스택에 추가하고 방문했다는 표시를 한다.

- ( 스택이 비어 있지 않은 동안 ) {

- 현재 노드를 스택의 최상위에서 가져온다.

- 현재 노드가 출구인지 확인하고, 출구라면 탐색 종료한다.

- 아니라면, 현재 노드의 이웃 노드를 방문하지 않았는지 확인하고,

방문하지 않았다면 이웃 노드를 스택에 추가하고 방문했다는 표시를 한다.

- 방문할 이웃 노드가 없으면 현재 노드를 재방문했다는 표시를 하고,

스택에서 현재 노드 제거한다.

시간 복잡도는 O(V + E)이며,

stack 의 크기는 최대 V이므로 공간 복잡도는 O (V)이다.

\* V = 노드의 수, E = 간선의 수

**< BFS >**

텍스트, 폰트, 스크린샷, 대수학이(가) 표시된 사진

자동 생성된 설명

자료구조 - maze\* ExitMaze : 미로 탐색을 종료하기 위한 미로의 출구를 저장한다.

maze\* curidx : 미로 탐색 시작을 위한 시작점을 저장한다.

queue <maze\*> MazeStack : BFS 탐색을 위한 큐를 선언한다.

알고리즘 - 초기화 : 모든 노드를 방문하지 않은 상태로 초기화한다.

**텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명**

알고리즘 - 시작 노드를 큐에 추가하고, 방문 및 재방문했다는 표시를 한다.

- ( 큐가 비어 있지 않은 동안 ){

- 현재 노드를 큐의 front()에서 가져오고, 해당 노드는 큐에서 pop한다.

- 현재 노드의 각 이웃 노드들을 확인하여 방문하지 않았다면,

큐에 추가하고, 해당 이웃 노드의 parent는 현재 노드로 저장한다.

마지막으로 해당 이웃 노드를 방문했다는 표시를 한다.

- 방문할 이웃 노드가 출구인지 확인하고, 출구라면 parent 포인터를 이용해

출발점까지의 경로 추적하면서 재방문을 했다는 표시를 한 뒤, 탐색을 종료한다.

시간 복잡도는 O(V + E)이며,

queue의 크기는 최대 V이므로 공간 복잡도는 O(V)이다.

\* V = 노드의 수, E = 간선의 수

**1.**

**2.**

**< DFS >**

- 장점

stack을 이용하여 구현이 간단하고, 출구가 깊은 곳에 위치하는 경우 빠르게 찾을 수 있다.

- 단점

정답이 없거나 cycle이 있는 경우 모든 경로를 탐색하기 전까지 종료되지 않을 수 있다.

또한 그래프의 depth가 너무 높을 경우 stack overflow의 위험성도 존재한다.

**< BFS >**

- 장점

각 level의 노드를 순차적으로 탐색하므로 최단 경로를 항상 찾을 수 있고,

이미 방문한 노드를 다시 탐색하지 않기 때문에 cycle이 있는 그래프에서도 잘 동작한다.

- 단점

현재 level의 모든 노드를 저장해야 하므로, 넓은 그래프에서는 메모리 사용량이 매우

높아질 수 있다. 또한 출구가 깊은 곳에 위치한 경우, 모든 노드를 탐색해야 할 수도 있어

비효율적이다.

따라서 미로가 복잡하고 깊이가 깊은 경우 또는 최단 경로를 구하는 것이 중요한 목표가

아닌 경우, DFS 알고리즘을 사용하는 것이 더 적합하고, 미로가 넓거나 최단 경로를 구하는 것이

중요한 목표인 경우에는 BFS 알고리즘을 사용하는 것이 더 적합하다.