전공 : 컴퓨터공학과 학년 : 4 학번 : 20212020 이름 : 박민준

1. DFS와 BFS의 시간 복잡도를 계산하고 그 과정을 설명한다.

- 깊이 우선 탐색(DFS)과 너비 우선 탐색(BFS)은 그래프 또는 트리 구조에서의 탐색 알고리즘이다.

1) DFS

텍스트, 스크린샷, 소프트웨어, 폰트이(가) 표시된 사진

자동 생성된 설명

- 위 사진은 DFS 알고리즘의 예시 코드이다. DFS 알고리즘의 탐색 과정은 다음과 같다. 먼저 DFS는 초기화 과정에서 스택(또는 재귀)을 사용하여 탐색할 노드를 저장한다. 처음에는 시작 정점을 스택에 넣고 방문 표시를 한다. 이후 스택에서 정점을 하나 꺼내 방문하며 탐색을 진행한다. 꺼낸 정점의 인접한 정점들을 확인하고, 방문하지 않은 정점을 스택에 넣고 방문 표시를 한다. 이 과정을 스택이 빌 때까지 반복한다. 스택이 빌 때까지 탐색을 계속하며, 모든 정점과 간선을 한 번씩 확인한다.

- DFS 알고리즘에서 정점의 개수 탐색 시, 그래프의 모든 정점을 방문해야 하므로 각 정점에 대해 방문 여부를 체크한다. 각 정점은 스택에 들어갔다가 다시 나오는 과정에서 한 번씩만 방문 되므로 정점 수만큼의 연산이 필요하다. 이 때의 시간 복잡도는 O(V)이다. 또한 간선의 개수 탐색 시, 각 정점에서 인접한 모든 간선을 탐색해야 한다. 각 간선을 한 번씩 방문하게 되므로 이 때의 시간 복잡도는 O(E)가 된다. 따라서 DFS에서 각 정점과 간선을 모두 탐색하므로 시간 복잡도는 두 합인 O(V+E)가 된다.

2) BFS

텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

- BFS는 DFS와 달리 스택이 아닌 큐를 사용하여 탐색할 노드를 저장한다. 먼저 시작 정점을 큐에 넣고 방문 표시를 한다. 이후 큐에서 정점을 하나 꺼내 방문한다. 꺼낸 정점의 인접한 정점들을 확인하고, 방문하지 않은 정점을 큐에 넣고 방문 표시를 한다. 이 과정을 큐가 빌 때까지 반복한다. 큐가 빌 때까지 탐색을 계속하며, 모든 정점과 간선을 한 번씩 확인한다.

- 각 정점은 큐에 들어갔다가 다시 나오는 과정에서 한 번씩만 방문 되므로 정점 수만큼의 연산이 필요하다. 또한, 각 간선은 BFS 탐색 과정에서 최대 두 번(출발 정점에서 도착 정점을 방문하는 과정에서) 확인된다. 따라서 BFS 알고리즘의 전체 시간 복잡도는 DFS와 마찬가지로 O(V + E)이다.

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

- 미로는 char\*\* input 배열과 vector<vector<char>> maze로 표현된다. input은 파일에서 읽어 들인 미로 데이터를 저장하고, maze는 탐색 및 경로 추적을 위해 사용하는 2차원 벡터이다. 각 노드의 방문 여부를 추적하기 위해 vector<vector<bool>> visited 배열을 사용하며, 탐색 과정에서의 부모 노드를 저장하여 경로를 추적할 때 vector<vector<pair<int, int>>> parent 배열을 사용한다. 이를 통해 탐색이 완료된 후 경로를 역추적하여 시작 지점에서 목표 지점까지의 경로를 구성할 수 있다.

- DFS는 깊이 우선 탐색으로, 경로를 한 방향으로 끝까지 탐색한 후, 막다른 길에 도달하면 돌아가서 다른 경로를 탐색하는 방식이다. 재귀를 사용하는 방식과 달리, 반복문 방식의 DFS는 스택을 활용하여 명시적으로 경로를 관리한다. 미로의 경로를 찾기 위해 먼저 visited 배열을 초기화하여 모든 위치를 방문하지 않은 상태로 설정한다. parent 배열을 초기화하여 각 노드의 부모 노드를 {-1, -1}로 설정한다. 스택을 초기화하고 시작 지점을 스택에 넣는다.

- 이후 탐색을 진행하는데, 스택에서 노드를 꺼내 현재 위치로 설정하고, 이 위치를 방문 처리한다. 현재 위치가 목표 지점(출구)인지 확인한다. 목표 지점에 도달하면 경로 추적을 시작한다. 현재 위치에서 갈 수 있는 4가지 방향(위, 아래, 왼쪽, 오른쪽)을 확인하고, 방문하지 않은 인접 노드를 스택에 추가하며, 이 노드의 부모 노드를 현재 위치로 설정한다. 목표 지점에 도달하면 parent 배열을 사용하여 시작 지점으로 돌아가는 경로를 추적한다. 이를 통해 전체 경로를 구성하고 저장할 수 있다.

- BFS는 너비 우선 탐색으로, 현재 레벨의 모든 노드를 탐색한 후, 다음 레벨로 이동하여 탐색을 계속하는 방식이다. 큐를 사용하여 탐색할 노드를 관리하고, 경로를 찾기 위해 각 노드의 부모 노드를 저장한다. 알고리즘 구현 방식은 DFS와 매우 유사하지만 탐색을 진행하는 과정에 있어 스택이 아닌 큐를 사용한다는 것에 차이가 있을 뿐이다.