CSE3013 (컴퓨터공학 설계 및 실험 I)
PRJ-2 미로 프로젝트 2주차 예비 보고서

서강대학교 컴퓨터공학과 박수현 (20181634)

서강대학교 컴퓨터공학과

1 목적

DFS와 BFS 알고리즘을 이해하고 미로 문제 해결에 어떻게 사용할 수 있을 것인지 보인다.

2 문제

2.1 그래프 탐색 알고리즘

너비 우선 탐색Breadth-First Search과 깊이 우선 탐색Depth-First Search는 그래프를 탐색하는 대표적인 알고리즘들이다.

어떤 노드 u와 인접한 노드들의 집합을 $N(u)$라고 하자. BFS는 큐 Q에 첫 노드를 넣고, Q가 빌 때까지 Q의 첫 원소 q에 대해 $N(q)$ 중 아직 방문하지 않은 정점들을 전부 Q에 추가하면서 그래프를 탐색해나간다. 다시 말하면 인접한 노드들부터 차례로 탐색해나가며, 트리의 경우 깊이가 얕은 노드부터 탐색해나간다.

DFS는 BFS와 반대의 전략을 취하는데, $N(u)$의 모든 원소 v_i에 대해 $N(v_i)$를 전부 방문한 후 v_{i+1}을 방문하는 방식으로 동작한다. 이를 위해 스택을 사용한다. DFS는 한 방향으로 깊이 탐색하다가 더 이상 진행할 수 없으면 이전 노드로 되돌아와 다른 방향으로 탐색하는데, 이전 노드로 되돌아오는 것을 백트래킹backtracking이라 한다.

인접 리스트로 표현된 그래프 $G = (V, E)$의 경우 각각의 방법의 시간 복잡도는 모두 $\mathcal{O}(\|V\| + \|E\|)$를 보인다.

2.2 미로 문제에의 적용

$h \times w$ 미로의 (i, j)에 위치한 셀을 u_{ij}라 하자. 그러면 u는 그래프의 노드로 생각할 수 있으며, 이웃한 노드는 최대 4개 – $u_{(i-1)j}$, $u_{(i+1)j}$, $u_{i(j-1)}$, $u_{i(j+1)}$ – 가 가능하다. 이는 미로의 가장자리에 있는 노드들을 제외하고 모두 동일하므로 간선 정보는 1주차에사 사용한 자료구조를 그대로 사용하고, 추가로

DFS/BFS의 수행을 위해 방문 순서와 여부를 저장하는 $h \times w$의 배열을 도입한다. 이 때 공간 복잡도는 $\mathcal{O}(h \times w)$가 된다. 또한 이 경우 두 알고리즘 공통적으로 현재 노드에서 인접한 4방향에 존재하는 노드 중 벽으로 막혀 있지 않으며 아직 방문하지 않은 노드를 쿼리하는 것은 단순히 2차원 배열의 원소에 4번 접근하는 것과 동일하다.

2.3 탐색 알고리즘의 구현 방법

- **DFS** – 인접한 노드를 방문하는 재귀함수 DFS(u)를 정의한다. u에 대해 $v \in N(u)$마다, v를 아직 방문하지 않았다면 DFS(v)를 호출한다.
- **BFS** – 큐 Q를 정의한다. 위에서 소개한 대로 Q에 첫 노드를 넣고, Q의 첫 원소 q에 대해 $N(q)$ 중 아직 방문하지 않은 정점들을 전부 Q에 추가하고, Q의 첫 원소를 제거한다. 이를 Q가 빌 때까지 반복한다.