CSE3013 (컴퓨터공학 설계 및 실험 I)
PRJ-2 미로 프로젝트 3주차 결과 보고서

서강대학교 컴퓨터공학과 박수현 (20181634)

서강대학교 컴퓨터공학과

1 목적

작성한 알고리즘과 자료구조를 기술한다. 알고리즘의 시간 및 공간 복잡도를 보이고 실험 전에 생각한 방법과 어떻게 다른지 보인다. DFS, BFS 알고리즘 각각의 장단점을 비교하고, 설계한 자료구조에는 어떤 알고리즘이 더 적합한지 기술한다.

2 문제

자료 구조의 구현으로 `std::vector<std::vector<char>>`를 사용하였다. 빈 칸은 '.', 벽은 '+', '-', '|' 중 하나로 이루어진 2차원 배열이다. .maz 파일을 읽어들여 그대로 2차원 배열에 저장하는 방식이다.

배열의 좌상단 인덱스 $(0,0)$, 크기를 $n \times m$이라고 할 때, 미로의 시작점은 $(1,1)$이고, 종점은 $(n-2, m-2)$이고, 어떤 칸 (x,y)를 노드라고 생각할 때 그 노드에 연결된 노드들은 인접한 상하좌우 4칸, 즉 $(x+1,y)$, $(x-1,y)$, $(x,y+1)$, $(x,y-1)$로 생각할 수 있다.

각 탐색 알고리즘의 구현은 다음과 같았다.

- **DFS** – 스택 S를 정의한다. S에 첫 노드를 넣고, S의 첫 원소 s에 대해 $N(s)$ 중 아직 방문하지 않은 정점들을 전부 S에 추가하고, S의 첫 원소를 제거한다. 이를 S가 빌 때까지 반복한다.
- **BFS** – 큐 Q를 정의한다. 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.

세부적으로, $p = (p_x, p_y)$에 대해 $N(p) = \{(p_x+1, p_y), (p_x-1, p_y), (p_x, p_y+1), (p_x, p_y-1)\}$ 으로 정의되었다.

방문 여부를 체크하기 위한 $n \times m$ 배열을 따로 만들어 관리했다. 각각의 알고리즘에서 최악의 경우 스택이나 큐에 $n \times m$개의 원소가 들어가고 나온다. 따라서 각각 알고리즘의 공간 복잡도는 $\mathcal{O}(n \times m)$이며, 시간 복잡도도 마찬가지다.

스택 자료구조로는 `std::stack`, 큐 자료구조로는 `std::queue`를 사용했다. 실험 전에 생각한 방법과 크게 달라진 점은 없었지만, 탐색 경로를 저장하기 위해 `std::pair<int, int>`의 2차원 동적 배열을 만들었다. 이 배열의 각 칸은 그 칸에 도달하기 직전의 칸의 좌표를 저장하고 있다.

2.1 DFS와 BFS의 장단점

DFS를 할 경우 스택에 저장된 좌표들의 시퀀스는 항상 시작점부터 끝점까지의 경로를 구성함을 보장하지만 BFS의 경우 그렇지 않다. 반면 BFS는 시작점부터 끝점까지의 경로가 최단 거리임을 보장한다. 따라서 완전 미로를 해결해야 하는 경우 DFS가 효율적이겠으나, 시작점부터 끝점까지의 경로가 다수 존재하는 불완전 미로를 해결해야 하는 경우 DFS는 최단 거리를 보장하지는 못하므로 이 때는 BFS를 사용하는 것이 바람직하겠다. 다만 굳이 최단 경로가 필요하지 않고 공간 최적화가 중요하다면 BFS는 일반적으로 큐에 동시에 저장된 최대 원소 수가 DFS의 스택에서의 그것보다 크므로 DFS를 쓰는 것이 좋겠다.