13주차 결과 보고서

전공 : 컴퓨터공학과 학년 : 2학년 학번 : 20211558 이름 : 윤준서

**1. 알고리즘과 자료구조**

**- DFS -**

텍스트, 스크린샷, 폰트, 문서이(가) 표시된 사진

자동 생성된 설명

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

***1.*** 칸 마다 탐색의 유무를 저장하기 위해 미로와 같은 크기의 배열 visit를 생성한다.

***2.*** 최단 경로 스택 d\_path, 전체 경로 스택 ad\_path를 초기화한다.

***3.*** 시작점 start(1, 1)를 d\_path에 저장한다. 이후 while문을 통해 d\_path가 모두 빌 때까지 탐색을 진행한다.

***4.*** 탐색 시작 시 가장 먼저 현재 좌표를 ad\_path에 저장한다. 이후 탐색 완료 조건을 판단한다.

***5.*** 하, 우, 좌, 상 순서대로 이동 가능 여부를 판단한다. 이동이 가능한 경우, 이동한 좌표를 d\_path에 저장한다. 그리고 해당 좌표에 알맞는 visit 위치를 true로 바꾼다.

***6.*** 만약 이동이 불가능한 경우, 그 전 좌표로 돌아가기 위해 d\_path.pop()을 실행한다.

해당 알고리즘의 시간 복잡도는 최악의 경우 모든 미로의 칸을 계산하므로 O(WIDTH \* HEIGHT)이다. 공간 복잡도는 최악의 경우 모든 미로의 칸을 방문하므로 O(WIDTH \* HEIGHT)이다. 예비 보고서와는 변수의 이름, 방향의 순서 이외에 차이점은 없다.

**- BFS -**

***1.*** 칸 마다 탐색의 유무를 저장하기 위해 미로와 같은 크기의 배열 visit를 생성한다.

***2.*** 탐색을 직접 실행하는 큐 Q와, 최단 경로를 생성할 때 이전 좌표를 추적하기 위한 pair parent를 생성한다. 그리고 최단 경로를 저장하는 큐 b\_path를 초기화한다.

***3.*** 다음 칸의 방향을 나타내는 두 배열 x\_move, y\_move를 생성한다. while문을 통해 Q가 빌 때까지 탐색을 시작한다.

***4.*** 현재 좌표를 나타내는 Block curr를 생성한다. curr을 통해 탐색을 하므로 Q에서는 pop()을 통해 제거한다.

텍스트, 스크린샷, 폰트이(가) 표시된 사진

자동 생성된 설명

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

자동 생성된 설명

***5.*** 탐색 완료 조건을 판단한다. 조건을 달성한 경우, 최단 거리를 b\_path에 저장한다. 이 때 pair process를 통해 curr좌표를 불러 r\_path에 저장하고 부모 좌표를 부른다. 이를 부모 좌표가 처음 시작 좌표인 (1, 1)에 도달할 때까지 반복한다.

***6.*** r\_path의 좌표 순서는 미로의 도착 지점에서 시작 지점이므로 d\_path는 r\_path와 반대로 저장해 순서대로 다시 조정한다. 경로 출력만을 위한 것이면 r\_path를 굳이 할 필요는 없다.

텍스트, 폰트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

***7.*** ab\_path의 문제점은 모든 방향을 함께 저장하는 BFS의 특징으로 인해 좌표들을 순서대로 저장할 경우 서로 이어진 좌표가 아닌 뒤죽박죽이 된다. 이 상황에서 DFS와 같은 방법으로 출력을 하면 경로가 복잡한 형태로 출력이 된다. 이를 해결하기 위해 처음 미로를 출력할 때와 같은 방법으로, 큐의 좌표를 이어서 출력하는 것이 아닌 선의 시작점과 끝점을 직접 저장해 출력한다. 그럼 좌표의 순서가 엉망이어도 출력에는 이상이 없다. 네 방향으로의 이동 가능을 판단하고, 가능할 시 Q에 다음 좌표를 저장하고, 현재 좌표와 다음 좌표를 선으로 갖는 Line을 PATH 벡터에 저장한다. 전체 경로 출력 시 PATH를 출력한다.

해당 알고리즘의 시간 복잡도는 최악의 경우 모든 미로의 칸을 계산하므로 O(WIDTH \* HEIGHT)이다. 공간 복잡도는 최악의 경우 모든 미로의 칸을 방문하므로 O(WIDTH \* HEIGHT)이다. 예비 보고서와는 변수의 이름, 방향의 순서가 변경됐고, 최단 경로를 저장하기 위해 부모 좌표의 정보가 담긴 pair를 이용했다. 전체 경로를 출력할 때 큐 내의 좌표를 순서대로 연결하는 출력 방법에 문제가 있으므로 경로의 좌표와 바로 다음 좌표를 선으로 저장해 간접적으로 출력하는 방법을 이용했다.

----------------------------------------------------------------------------------------------------

**2. 장단점**

**- DFS -**

DFS의 장점으로 경로를 탐색하며 삭제와 저장을 반복하기에 메모리 사용량이 적다. 그리고 재귀적으로 구현하기에 코드가 비교적 간단하다. 방향을 하나씩 정해서 순차적으로 탐색하기에 방향의 순서가 미로와 잘 맞고, 미로가 클 수록 효율적이다.

단점으로 재귀적인 형태 때문에 잘못 구현하면 무한 루프에 빠질 수 있다. 그리고 방향의 순서를 정해서 탐색하기 때문에 그 순서가 미로의 모양과 상극이면 비효율적인 탐색을 진행한다.

**- BFS -**

BFS의 장점으로 재귀적인 형태가 아니기에 무한 루프에 빠지는 오류의 가능성이 적다. 그리고 모든 방향을 함께 탐색하므로 빠른 탐색 시간이 빠르다.

단점으로 모든 방향을 탐색하기 때문에 메모리 사용량이 많다. 그리고 재귀 형태가 아니기에 형태가 복잡하다.

DFS와 BFS는 서로 상반되는 장단점을 가지고 있는 것이 특징이다. 시간 복잡도와 공간 복잡도 모두 같은 알고리즘이다. 하지만 모든 경로를 출력할 때 DFS는 직접 저장한 스택을 바로 출력할 수 있어 용이하다. BFS의 경우 선을 새로 만들어 간접적으로 출력해야 하므로 전체 경로를 출력할 땐 DFS가 더 적합하다. 또한 미로의 경우 시작점이 좌상단, 도착점이 우하단에 위치하므로 방향을 하, 우, 좌, 상 순서로 탐색하는 DFS가 미로의 모양에 따라 효율성이 크게 좋다.