13주차 결과보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20211522 이름: 김정환

**1.**

.................

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명텍스트, 스크린샷, 폰트, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

우선 DFS는 stack을 사용하여 iterative하게 구현하였다. 함수 시작에 visited의 메모리와 stack의 노드를 저장할 array를 할당해주었다. stack 같은 경우에는 row와 col을 각각 다른 배열에 따로 저장하는 방식으로 구현하였다. 그 후에는 stack에 시작지점인 row = 1, col = 1인 지점을 넣어준다. 그리고 stack이 비어있지 않은 동안 반복하면서 우선 방문을 표시해주고 도착 지점에 도착했는지 확인한다. 도착한 경우에는 isdfs를 1로 바꾸어 dfsdraw가 실행될 수 있도록 하고, true를 반환한다. 아닌 경우에는 상하좌우에 대해서 이동 가능하고 방문한 적 없는지 확인하여 해당하는 경우에 stack에 push해주고 flag를 1로 바꾸어 pop이 되지 않도록 하고 방문 표시 후 해당 위치로 row와 col을 이동한다. 위의 4방향에서 push 되지 않은 경우에는 이동할 수 없는 것이므로 pop을 해주어 이전 위치로 이동한다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

dfsdraw()에서는 우선 visited에 표시된 위치들에 대해서 우측과 아래 방향에 대해 두 지점이 다 방문했던 적이 있는 경우에 대해서 선을 연결하여 표시하도록 하였다. 이 부분은 최단 경로가 아니기에 연한 색으로 표기하였다. 그 후 진한 색으로 변경한 뒤 stack에 들어있는 row와 col을 이용하여 순서대로 연결해주어 최단 경로를 표시할 수 있도록 했다.

텍스트, 스크린샷, 디스플레이, 폰트이(가) 표시된 사진

자동 생성된 설명

BFS는 우선 visited와 before 두 가지에 대한 배열을 할당해주었다. visited는 방문 여부를 저장하고, before는 row와 col만 구조체로 저장하는 배열이다. before는 bfs 방문 시 이전 노드의 위치를 저장하는 배열이다. 그 후 row와 col을 저장할 queue 배열을 할당하였다. 시작 지점을 queue에 넣어주고, 방문 표시 후에 before를 row = 0, col = 0으로 하여 마지막임을 표시했다.

텍스트, 스크린샷, 소프트웨어이(가) 표시된 사진

자동 생성된 설명

그 후엔 queue에서 하나씩 빼주며 각각에 대해 상하좌우 모두 탐색한다. 각 방향마다 갈 수 있는지와 방문여부를 확인하고 해당할 경우에 queue에 넣어주고 방문 표시, 그리고 before에 현재의 위치를 넣어준다. 도착지점에 도달한 경우에는 isBFS를 1로 바꾸어 bfsdraw를 실행할 수 있도록 하였고 true를 return한다.

텍스트, 스크린샷이(가) 표시된 사진

자동 생성된 설명

bfsdraw는 우선 최단 경로가 아닌 부분은 dfs와 동일하다. 그리고 최단경로는 도착지점부터 before로 이동시키며 시작지점에 도달할 때까지 현재 위치와 before를 연결시킨다.

DFS와 BFS 모두 알고리즘 상의 시간 복잡도와 공간 복잡도는 O(N\*M), O(N\*M)이다. 그러나 구현하는 과정에서 데이터를 불러올 때 search를 통해서 Linked List를 탐색하는 부분이 O(N\*M)의 시간 복잡도로 추가되어 있어 구현한 프로그램은 시간 복잡도는 O(N\*N\*M\*M), 공간 복잡도는 O(N\*M)이다.

................

**2.**

.......................

DFS는 한 방향에 대해서 우선 끝까지 탐색하기에 최적의 경우에는 정말 빠르게 찾을 수 있다는 장점이 있다. 또한 stack 상에 최적의 경로가 바로 저장된다. 그러나 단점으로는 최악의 경우에 모든 경우의 수를 탐색하기에 오랜 시간이 걸릴 수 있다. 반면에 BFS는 모든 방향을 탐색하며 가므로 항상 최적의 해를 찾는 것이 보장된다. 그러나 DFS에서 최적의 경우인 경우에 대해서도 DFS에 비해 느린 시간에 최적의 경로를 찾는 단점이 존재한다. 또한 최단 경로를 저장하기 위해서 추가로 다른 배열을 더 이용해야 하는 단점도 존재한다.

해당 프로그램 내에서는 DFS가 더 효율적이라고 판단하였다. 그 근거로는 우선 주어지는 데이터가 항상 경로가 존재하는 미로이므로 DFS의 depth에 대한 고려가 필요하지 않은 점이 있다. 그리고 자료구조를 구현하면서 Linked List와 simple array를 이용하기도 했고, BFS에는 최적의 경로 저장을 위해서 배열을 추가 선언해줘야 하므로 메모리의 사용량 상 DFS가 더 효율적이라고 판단하였다.

.........................