13주차 예비보고서

전공: 컴퓨터공학과 학년: 2학년 학번: 20221532 이름: 김민기

1.

정점의 수 : N, 간선의 수 : E라 하면,

DFS()의 시간 복잡도는 모든 정점과 간선을 한번 씩 탐색하므로 최대 O(N + E)이다. 공간 복잡도는 O(N)이다.

DFS알고리즘은 모든 정점을 탐색하는 알고리즘이다. 특정 정점에서 다음 간선으로 넘어가기 전 해당 간선의 모든 정점을 탐색한다. 단계는

1) 시작 정점을 방문한 것으로 표시

2) 인접한 정점중 방문하지 않은 정점 선택

3) 이동하여 방문한 것으로 표시

4) 해당 정점에서 방문하지 않은 정점을 선택

5) 위의 단계를 반복하여 모든 정점을 탐색

BFS()의 시간 복잡도는 DFS와 마찬가지로 최대 O(N +. E), 공간 복잡도 또한 O(N)이다.

BFS알고리즘은 시작 정점에서부터 레벨별로 탐색을 진행한다.

1) 시작 정점을 큐에 넣고 방문한 것으로 표시

2) 큐에서 정점을 하나씩 꺼내고 해당 정점과 인접한 정점을 확인

3) 인접한 정점들 중 방문하지 않은 정점을 큐에 넣고 방문한 것으로 표시

4) 큐가 비어 있을 때 까지 위의 단계를 반복

2.

12주차에서 자료구조를 배열로 선택하여 텍스트 파일의 값을 그대로 저장했기에 배열을 탐색하며 DFS알고리즘을 구현한다. 아래는 pseude code 이다.

S에 시작점 push

시작점 visited 변경

while (S 가 비어있지 않으면) {

if (S.top이 도착점이라면) {

return;

}

if (이동할 수 있는 정점이라면) {

S에 해당 정점 push

최적경로 벡터에 해당 위치 push

Visited 변경

}

Else{

S.pop()

}

}

이동할 수 있는 정점인지 판단하기 위해선 배열을 이용해야 하기 때문에 상하좌우 4방향을 모두 검사한다.

BFS알고리즘에 대한 pseudo code이다.

void BFS(int y, int x) {

Q에 시작점 push

시작점 visited 변경

while (Q가 비어있지 않으면) {

front에 Q.front 저장

Q.pop

int i = front / WIDTH;

int j = front % WIDTH;

int curY = i \* 2 + 1;

int curX = j \* 2 + 1;

if (현재 정점이 도착점이라면) {

while (첫 시작점까지 반복) {

최적경로 벡터에 prev값 push

}

bestLine 벡터를 역순으로 뒤집음

return;

}

if (이동할 수 있는 정점이라면) {

Q에 해당 정점 push

Visited 변경

Prev에 저장

}

}

}