13주차 예비 보고서

전공: 아트앤테크놀로지학과 학년: 3학년 학번: 20201116 이름: 이수빈

1.

- DFS

DFS는 깊이 우선 탐색 기법으로, 그래프에서 찾고자 하는 정점이 그래프의 레벨이 큰 곳에 있을 경우 (깊은 곳에 있을 경우), 비교적 빠르게 경로를 구할 수 있는 방법이다. 그러나 이 해가 최적의 경로, 즉 가장 빠른 경로가 된다는 보장은 없다. 이때, 그래프를 무슨 자료 구조를 사용해서 구현하는지에 따라 시간 복잡도가 달라진다.

만약 인접 행렬을 이용해서 구현을 하게 된다면 O(V^2)의 시간 복잡도를 갖게 된다. 이는 공간 복잡도도 마찬가지이다. 그 이유는 모든 정점에서 다른 정점까지의 경로 존재 여부를 나타내는 메트릭스를 정의를 해야 한다. 그렇기 때문에 특정 정점에서 다른 임의의 정점까지 경로가 있는지 확인하고 있다면 그 정점들을 스택에 넣어주게 되는데 이 확인하는 과정이 O(V) 또한 이 DFS과정이 최대 V번 일어나기 때문에 결과적으로 O(V^2)의 시간 복잡도를 갖게 된다.

그러나 만약 인접 리스트를 이용하여 구현하게 된다면 위의 인접 행렬보다는 짧은 O(E+V)의 시간 복잡도를 갖는다. 이는 각 정점마다 모든 정점이 아닌 이어져 있는 정점의 정보만 저장한다. 따라서 공간 복잡도도 O(V+E)이다. 탐색도 이에 대해서만 진행하면 되기 때문에 결과적으로 O(V+E) 시간 복잡도를 갖기 때문에 위의 인접 행렬보다 훨씬 효율적이다.

* BFS

BFS는 DFS와 달리 넓이 우선 탐색하는 알고리즘으로, 그래프에서 찾고자 하는 정점이 그래프의 레벨이 낮은 곳에 있을 경우 더욱 빠른 알고리즘이다. 그러나 이 또한 DFS와 마찬가지로 최적의 해를 돌려준다는 보장은 없다.

시간 복잡도는 DFS와 동일하다. 그 이유는 같은 과정을 반복하지만 깊이를 우선으로 탐색하느냐 넓이를 우선으로 탐색하느냐만 다르기 때문이다. 따라서 만약 인접 행렬을 사용했다면 O(V^2)의 시간 복잡도를 갖을 것이고 인접 리스트를 사용 했다면 O(V+E)의 시간 복잡도를 갖을 것이다. (V: 정점의 개수 E: 간선의 개수)

2.

DFS를 구현하는 방법에는 iterative한 방식과 recursive한 방식이 있다. recursive한 방식은 DFS함수를 재귀적으로 호출해 구현한다. 그러나 본 실험에서는 iterative한 방식을 이용해 구현 할 예정이다.

iterative하게 구한하기 위해서는 우선 정점들을 저장하기 위한 스택이 필요한다. 우선 시작하는 정점을 스택에 넣어준다. 그 후 while문을 이용해 이 stack이 빌 때까지 반복한다. while문 안에서는 stack의 top에 있는 정점을 pop하고 이에 인접 리스트로 구현 되어 있는 인접한 정점들을 stack에 push해준다. 이를 반복하다 스택의 top에 있는 정점이 목표한 정점과 동일하다면 while문을 break하는 방식으로 프로젝트에 구현할 예정이다.