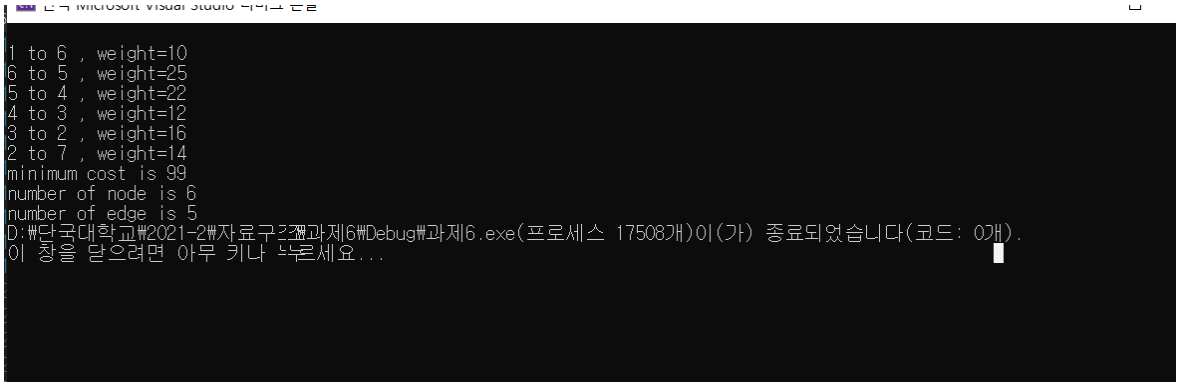
Problem 1. 최소 길이 스패닝 트리를 찾는 Prim 알고리즘을 구현하시오.

입력 그래프의 크기 조건(노드 수와 엣지 수)을 만족하기 위해 크기를 사용자에게 입력 받아 랜덤으로 노드수와 엣지수를 할당하려고 했으나 어려움을 느껴 구현하지 못하였고, main 함수에서 입력 그래프를 2차원 배열로 미리 지정하여 사용하였다. 가중치를 가지지 않는 엣지는 모두 0으로 설정하였다.

Prim() 함수를 함수 모듈로 구현하여 main()에서 입력 그래프와 여러가지 출력하고자 하는 변수들의 주소를 포인터 형식으로 전달하여 호출해 사용했다.

cycle여부 확인을 위해 int visited[노드의 개수]={0} 라는 배열을 만들어 방문한 적 있는 노드는 1로 표시해줬다.

이런 방식은 기존의 입력그래프 원본을 변경시키기 때문에 그닥 좋은 방법이라고는 생각되지 않는다. 따라서 입력 그래프의 복사본을 출력 그래프로 하여 이것을 변경시키는 방법을 시도해보는 것도 좋을 것 같다. 또한 2차원 배열에서 0의 값이 많고 2차원 행렬은 연결 리스트에 비해 시간복잡도가 오래 걸리는 단점이 있기 때문에 차라리 이 행렬을 희소행렬도 만들면 시간복잡도를 개선할 수 있지 않을까 싶다.



Problem 2. 최소 길이 스패닝 트리를 찾는 Kruskal 알고리즘을 구현하시오.

prims 알고리즘은 처음 최소 가중치 엣지를 찾은 후 각 노드에 인접한 최소 가중치를 찾지만 Kruskal 알고리즘의 경우 매번 전체에서 최소 가중치를 찾기 때문에 cycle 여부 확인을 위해서는 해당 노드를 선택하기 이전에 선택했던 노드와 같은지에 대한 확인을 따로 함수로 만들어 구현했다.

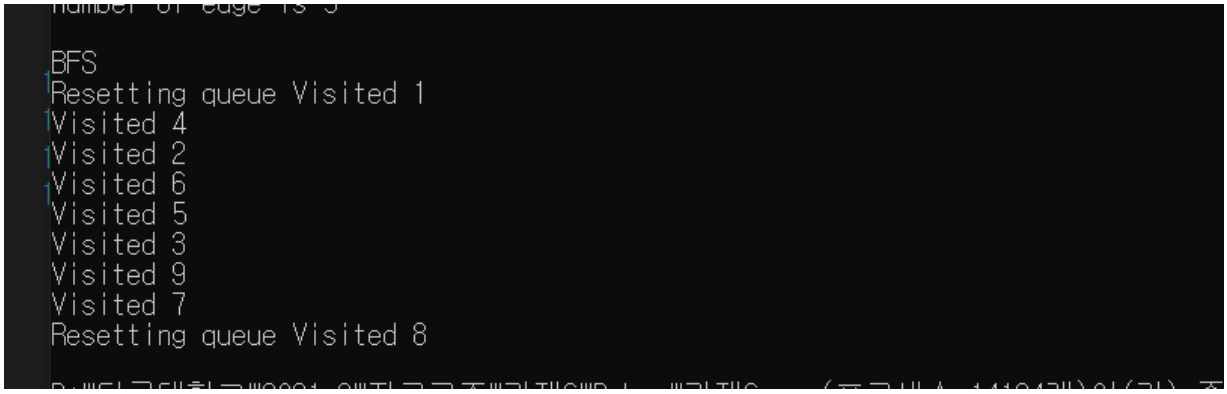
Prim 알고리즘과 선택된 경로는 똑같지만 Kruskal 알고리즘의 경우 최소 가중치 순으로 찾기 때문에 출력 순서가 다름을 알 수 있었다.

Problem 3. 다음 탐색 알고리즘을 구현하시오.

- Breadth-first search algorithm

문제 1,2번 에서는 행렬을 사용했지만 이번엔 구조체를 사용해 연결리스트를 사용해 입력 그래프를 만들었다. 해당 문제들에서 입력 그래프를 출력하진 못했는데 모두 수업 자료에 있는 그래프를 입력 그래프로 사용하였다.

큐를 사용하여 탐색하기 때문에 큐의 기본 구현을 했고, BFS 알고리즘을 구현하여 탐색을 하고 큐에서 dequeue 할 때마다 노드를 출력해줬다.



총평

저번 트리 과제에 이어 이번 과제 난이도도 어렵게 느껴져서 문제를 모두 풀지 못한 것이 아쉽다. 시간을 더 할애하여 자료구조 알고리즘에 대해 더 제대로 이해하고 다시 풀어보며 내 것으로 만

들어야 할 것 같다.

main함수는 1개만 만들고 위해서 구현한 함수들을 호출하여 사용하였는데 이러한 방식이 모듈화가 덜 되었다는 생각이 들어 보완하고 싶고,