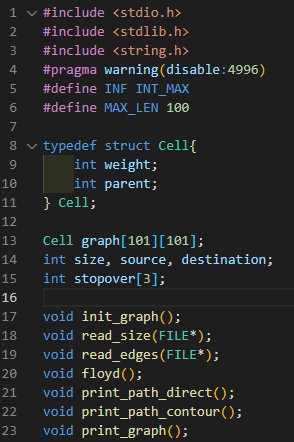
데이터 구조 및 알고리즘

HW EXTRA

2019171009

정성훈

1. 데이터 구조 및 함수 선언

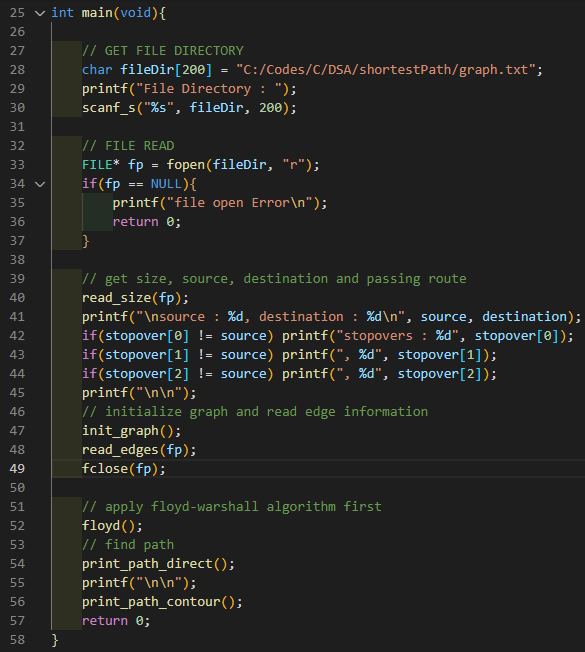


graph를 adjacent matrix 형태로 저장할 것이고, 각 edge마다 무게 정보와 parent 정보를 저장하기 위해 cell이라는 구조체를 만들어 구조체 이진 배열을 선언해 그래프 정보를 저장했습니다.

그래프의 size와 source, destination 그리고 경유지인 stopover는 프로그램 전체에서 사용할 것이기 때문에 global로 선언하였습니다.

아래는 함수 선언부입니다.

2. 메인 함수

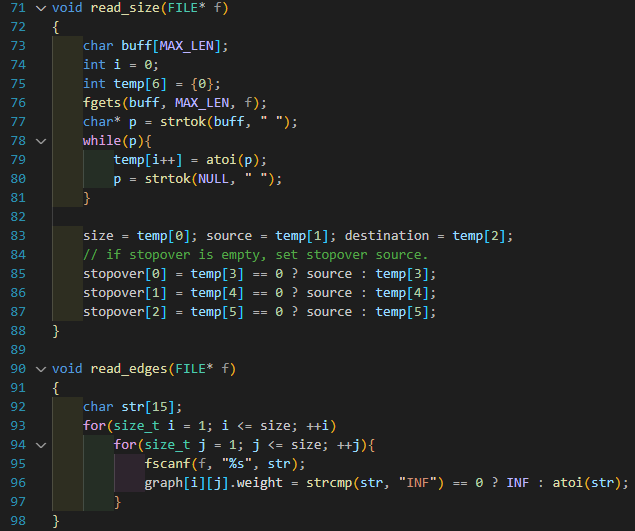


함수를 실행하면 먼저 graph 정보가 들어있는 파일의 경로를 scanf로 입력받도록 했습니다. 이후 read\_size 함수로 파일을 읽어 그래프의 size와 source, destination과 경유지를 읽도록 했습니다.

이후 init\_graph로 그래프를 초기화시키고 read\_edges 함수로 그래프정보를 파일에서 읽어왔습니다. 다 읽었으면 파일을 닫고, 플로이드-와샬 알고리즘을 그래프에 적용했습니다.

그를 통해 얻은 정보로 source에서 destination으로 바로 가는 direct path와 경유지를 거쳐 가는 contour path를 찾아서 출력하도록 했습니다.

3. 파일 읽기 (read\_size, read\_edges)

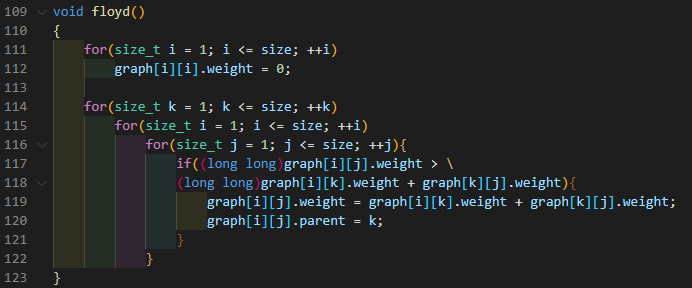


먼저 read\_size 함수입니다. 파일의 첫 줄을 string으로 읽은 뒤 띄어쓰기 단위로 strip한 뒤 정수형태로 순서대로 size, source, destination, stopovers로 저장했습니다. 이 때 경유지의 개수는 고정되어있지 않기 때문에 코드의 일반화를 위해 입력받지 않은 경유지는 source로 지정해 항상 모든 경유지를 지나도록 코딩했습니다.

read\_edges 함수입니다. 이차원 배열 형태로 graph 정보가 들어오기 때문에 그대로 하나 씩 fscanf로 읽으며 그래프에 weight 정보를 저장하면 됩니다.

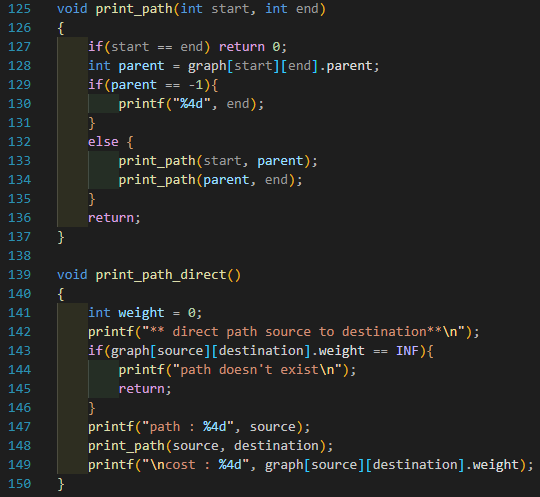
이 때 두 vertex가 연결되어 있지 않으면 ‘INF’라고 쓰여져 있기 때문에 “INF”라는 값이 읽히면 해당 간선의 가중치를 INF로 정했고, 그렇지 않으면 atoi를 통해 간선의 가중치를 저장했습니다.

4. Floyd-Warshall Algorithm



Floyd-Warshall 알고리즘입니다. 강의자료에 나와 있기 때문에 설명은 생략하도록 하겠습니다.

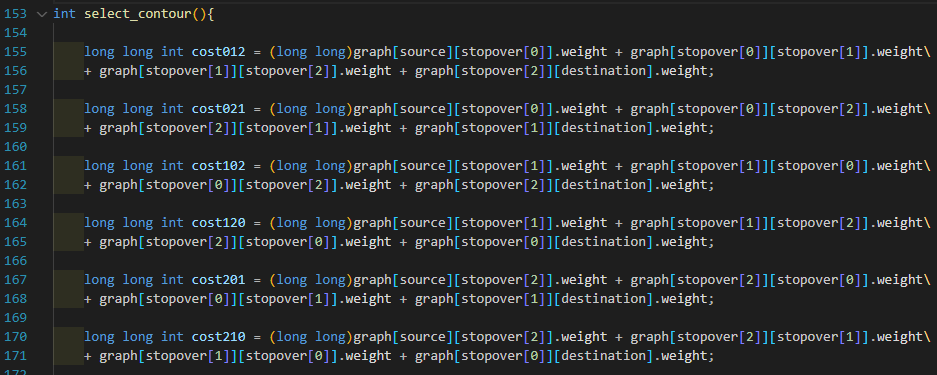
5. Direct Path



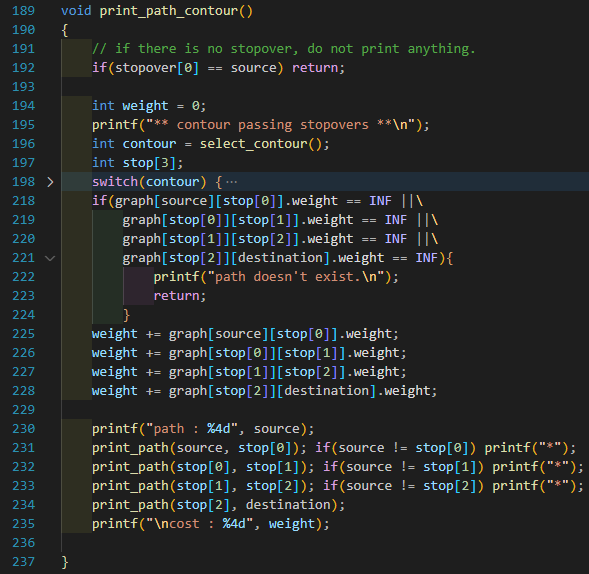
print\_path는 경로를 출력하는 함수입니다. 만약 start와 end 사이의 shortest path의 parent가 -1이면 그래프에 원래 있던 간선이므로 end vertex를 출력하고, 그렇지 않으면 start와 parent의 path, parent와 end의 path를 찾도록 재귀함수로 구성했습니다.

아래는 경유지를 무시하고 출발지에서 목적지로 바로 가는 shortest path를 출력하는 함수입니다. 만약 경로가 존재하지 않으면 해당 문구를 출력하고 함수가 종료되고, 경로가 있으면 해당 경로와 비용을 출력합니다.

6. Shortest Contour



경유지가 1, 2, 3으로 3개가 있을때 이를 거치는 경우의 수는 총 6개가 있습니다. 위 함수는 그 6개의 경로중 가장 비용이 작은 경로를 찾는 select\_contour 함수입니다. 6가지 경우의 수를 모두 계산해서 가장 짧은 경로를 리턴하도록 했습니다.

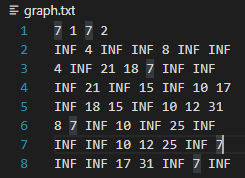
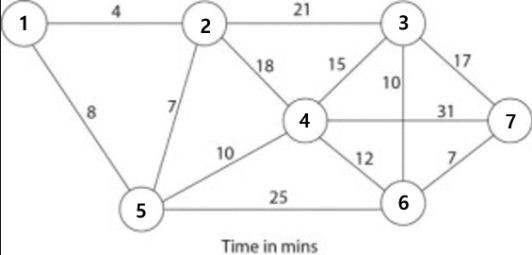


위는 경유지를 거쳐가는 shortest path를 출력하는 함수입니다. 만약 시작 파일에서 경유지가 주어지지 않았다면 그대로 함수를 종료하도록 했습니다.

select\_contour 함수를 이용해 경유지를 거치는 6가지 경우의 수 중 가장 짧은 길을 택하고, 만약 path가 존재하지 않으면 해당 문구를 출력하고 함수를 종료하도록 했습니다.

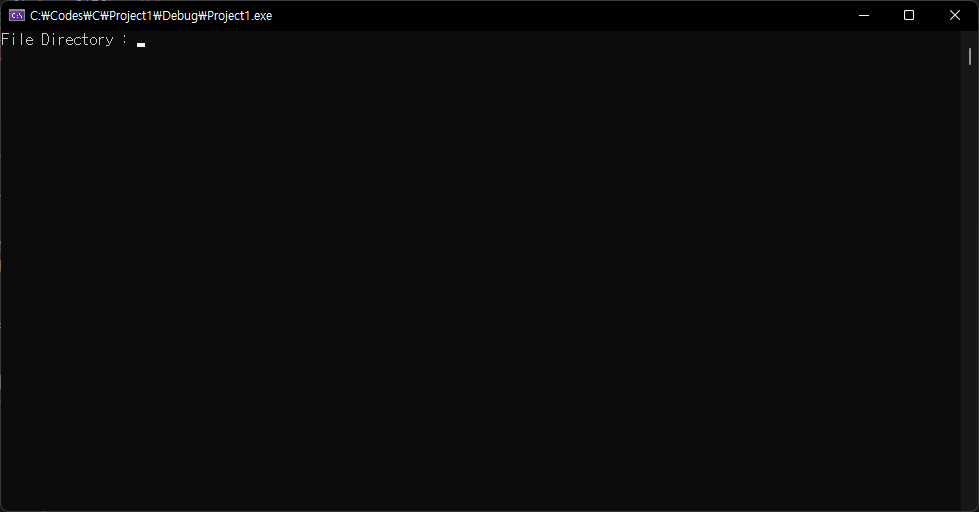
path가 존재하면 path를 출력하고 비용까지 출력하도록 했고, 알아보기 쉽게 하기 위해 경유지를 거친 직후에 ‘\*’을 찍어서 경유지를 거쳤다는 것을 표시했습니다.

7. 실행 예시

왼쪽은 입력으로 준 file인 graph.txt이고 오른쪽은 해당 graph를 시각화한 것 입니다.

file의 첫줄에는 vertext의 개수인 7과 시작 vertex인 1, 끝 vertex인 7, 그리고 경유지인 2가 주어져있습니다.

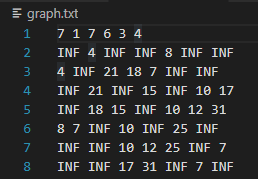


프로그램을 실행하면 위와 같이 File의 Directory를 입력하라고 합니다.



위와 같이 graph.txt의 경로를 입력하면 파일을 읽어서 source, destination과 stopover들을 식별한 뒤에 출력하고, direct path와 contour를 출력한 뒤 프로그램을 종료합니다.

이 때 경유지인 2를 거친 후에 ‘\*’이 찍혀 2를 언제 거쳤는지 확인할 수 있습니다.



이번에는 3, 4, 6을 경유하도록 해보겠습니다.



path를 찾은 결과가 출력된 것을 확인할 수 있습니다.

이상입니다. 감사합니다.