

 REPORT

자료구조 과제 보고서

[학생위치정보 관리 프로그램 설계]

과 목 명 : 자료구조

담당교수 : 심정섭 교수님

제 출 일 : 2015년 12월 8일 화요일

01041975132

yoohoogun114@naver.com

컴퓨터공학과 2학년 12121518 이름:유호균

**설계의 목적**

그래프와 그래프의 멤버함수를 설계하고 코딩함으로써 구현 능력, 설계능력을 향상시켜 실제 프로그램 작성 능력을 향상시키고 그래프를 이해 하는데 그 목적이 있다.

**요구사항**

파일을 입력 받아 그래프를 구현, 표준입력, 학생 추가함수, 학생정보 출력함수, 도로 추가함수, 학생 삭제함수, 도로 삭제함수, 연결된 도로 출력함수, 도로정보 출력1(incident)함수, 도로정보 출력2(incident available)함수, 도로존재 여부 확인(adjacent)함수.

**개발 환경**

MacBook Pro OS X 10.11 엘케피텐 Xcode 7.1.1 에서 코드작성, 디버그 작업 후 메일로 코드를 옮겨 Windows 8.1 Visual Studio Professional 2013에서 디버그 및 테스트, 자료 입출력

**필요한 자료구조 및 기능**

자료구조: Struct(학생: vertex), Struct(학생 시퀀스: vertex sequence), Struct(매개 시퀀스: 학생과 도로를 연결), Struct(도로: edge), Struct(도로 시퀀스: edge sequence) ,Class(학생 위치정보: 그래프)

기능: 파일로부터 데이터를 불러와서 파일에 저장된 회원정보를 바탕으로 그래프 구축, 학생 추가, 학생정보 출력, 도로 추가, 학생 삭제, 도로 삭제, 연결된 도로 출력, 도로정보 출력(incident), 도로정보 출력(available), 도로존재 여부 확인(adjacent)

**과제를 구현하기 위해 사용한 세부 자료구조 (구조체, 클래스 위주설명)**

struct road{ //road struct

para\_sequence\_node\* pointer\_first\_para\_node = NULL; //첫번째 매개시퀀스 가리키는 포인터

para\_sequence\_node\* pointer\_second\_para\_node = NULL; //두번째 매개시퀀스 가리키는 포인터

road\_sequence\_node\* pointer\_road\_sequence\_node = NULL;//도로 시퀀스를 가리킴

int road\_id = NULL; //도로 번호

int first\_student\_id = 0; //도로에 연결된 학생1

int second\_student\_id = 0; //도로에 연결된 학생2

};

struct student{ //student struct

student\_sequence\_node\* pointer\_student\_node = NULL; //학생 시퀀스 포인터

para\_sequence\_node\* pointer\_para\_node = NULL; //매개 시퀀스 포인터

int student\_id = 0; //학번

int degree = 0; //degree 차수

string email = ""; //email 주소

};

struct student\_sequence\_node { //student sequence node

student\_sequence\_node \*previous = NULL; //이전 학생 시퀀스

student\_sequence\_node \*next = NULL; //다음 학생 시퀀스

student \*point\_student = NULL; //학생 포인터

};

struct para\_sequence\_node{ //student struct와 road struct 매개

para\_sequence\_node\* next = NULL; //이전 매개 시퀀스

para\_sequence\_node\* previous = NULL; //다음 매개 시퀀스

road \*point\_road = NULL; //도로 포인터

};

struct road\_sequence\_node{ //도로 시퀀스 node

road\_sequence\_node \*previous = NULL; //이전 도로

road\_sequence\_node \*next = NULL; //다음 도로

road \*pointer\_road = NULL; //도로 포인터

};

class location\_graph{ //Graph Class

private:

road\_sequence\_node \*first\_road = NULL; //도로 리스트(시퀀스) 첫번째 node

//para\_sequence\_node \*first\_para = NULL; //매개변수 리스트(시퀀스) 첫번째 node

student\_sequence\_node \*first\_student = NULL; //학생 리스트(시퀀스) 첫번째 node

int total\_student; //전체 학생수

int total\_road; //전체 도로수

public:

location\_graph(int, int); //생성자

~location\_graph(); //소멸자

void insert\_student(int, string); //1.학생 추가

void print\_student(int); //2.학생정보 출력

void insert\_road(int, int, int); //3.도로 추가

void remove\_student(int); //4.학생 삭제

void remove\_road(int); //5.도로 삭제

void print\_incident\_road(int); //6.연결된 도로정보 출력 INCIDENT

void print\_road\_info(int); //7.도로정보 출력1

void print\_available\_incident(int, int); //8.도로정보 출력2

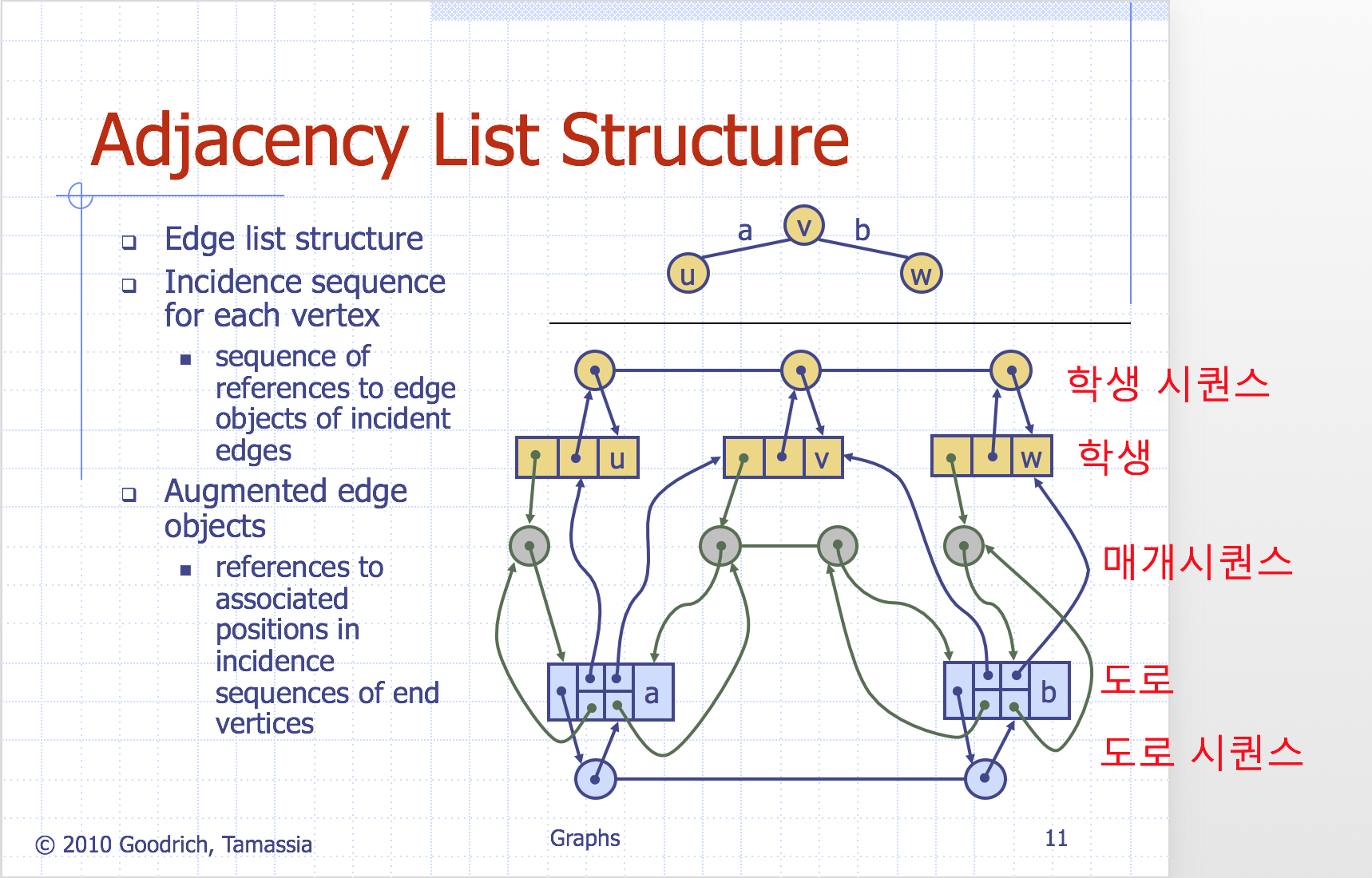
void print\_adjacent\_student(int, int); //9. 도로존재 여부 확인

student\_sequence\_node\* search\_stduent\_by\_id(int); //학번으로 학생시퀀스노드 찾기

road\_sequence\_node\* search\_road\_by\_id(int); //학번으로 도로시퀀스노드 찾기

};

**설계 그림**



**기능별 알고리즘 명세**

기능은 함수위주로 설명하겠습니다.

Big O 시간 측정은 해당위치 찾기(Search)를 제외한 함수이름만의 기능을 수했을때 시간을 기준으로 합니다. Search시간은 모두 선형 O(N)이 걸립니다.(선형탐색)

**location\_graph(int, int);**

생성자 O(1)

멤버 변수들을 초기화함

**~location\_graph();**

소멸자 O(1)

**void insert\_student(int, string);**

1. 학생 추가 O(1)

학생들을 그래프에 추가한다. 입력받은 학번을 통해서 학생 시퀀스의 끝을 찾아서 학생 시퀀스노드를 추가하고 학생 노드를 생성후 서로 연결한다. => 상수시간

**void print\_student(int);**

1. 학생정보 출력 O(1)

입력받은 학번을 통해 학생 시퀀스의 학생을 찾아서 학생의 정보를 출력함 => 상수시간

**void insert\_road(int, int, int);**

1. 도로 추가 O(1)

입력받은 두 학생의 학번을 통해 두 학생 시퀀스를 찾고 매개 시퀀스 노드를 두개 생성하고 도로 노드를 생성한다. 도로 시퀀스 노드도 한개 생성한다. 학생노드 – 매개 시퀀스 – 도로 – 도로 시퀀스를 서로 연결한다. => 생성, 연결하는데 모두 상수시간

**void remove\_student(int);**

1. 학생 삭제 O(deg(v)) => 도로 삭제 포함

입력받은 학번을 통해 학생 시퀀스 노드에 접근한다. 학생시퀀스에서 학생노드로 접근하고 학생노드에서 매개시퀀스들을 각각 탐색한다. 이는 학생의 degree만큼 수행된다. 매개 시퀀스 노드와 연결된 각각의 도로 연결들을 제거한다. 본 도로의 시퀀스의 앞뒤를 연결하고 본 도로 시퀀스를 제거한다. 그리고 나서 도로 노드를 앞뒤로 연결한다. 그 후 도로 노드를 제거한다. 본 매개 시퀀스 노드의 앞뒤를 연결하고 매개 시퀀스 노드를 삭제한다. 본 학생 시퀀스 노드들의 앞뒤를 연결하고 본 학생 시퀀스를 연결한다. 본 학생 시퀀스 노드를 제거한다. 마지막으로 본 학생 노드를 제거한다.

**void remove\_road(int);**

1. 도로 삭제 O(1)

도로 번호를 입력받아 도로 시퀀스 노드에 접근한다. 도로 시퀀스에서 도로로 접근하고 도로와 연결된 매개시퀀스 노드의 연결을 제거한다. 본 도로의 매개 시퀀스 노드의 앞뒤를 연결하고 본 도로의 매개 시퀀스노드를 제거한다. 본 도로 시퀀스 노드의 앞뒤를 연결하고 본 도로 시퀀스 노드를 제거한다. 마지막으로 본 도로 노드를 제거한다.

**void print\_incident\_road(int);**

1. 연결된 도로정보 출력 INCIDENT O(degree(v))

학번을 입력받아 학생 시퀀스 노드에 접근한다. 학생 시퀀스 노드에서 학생으로 접근하고 매개 시퀀스 노드로 접근한다. 여기 매개 시퀀스 노드와 연결된 매개 시퀀스 노드가 각각의 도로를 의미한다. (degree(v)). 각각의 매개 시퀀스 노드에 접근해서 도로 노드로 접근하고 도로 노드에서 정보를 출력한다.

**void print\_road\_info(int);**

1. 도로정보 출력1 O(1)

도로 번호를 입력받아 도로 시퀀스 노드에 접근한다. 도로 시퀀스 노드에서 도로에 접근하고 도로의 정보를 출력한다. 큰 학번을 먼저 출력하기 위해 큰 학번과 작은 학번을 변경하는 과정도 상수시간이다.

**void print\_available\_incident(int, int);**

1. 도로정보 출력2 O(1)

도로 번호와 학번을 입력받는다. 입력받은 도로 번호로 도로 시퀀스 노드에 접근 할 수 있다.

도로 시퀀스 노드를 통해 도로 노드에 접근할 수 있고 각 도로에는 first\_student\_id, second\_student\_id 정보가 저장되어 있으므로 상수시간에 도로와 학생이 연결되어 있는지 알 수 있다.

**void print\_adjacent\_student(int, int);**

1. 도로존재 여부 확인 O(Min(deg(v),deg(w)))

학번을 2개 입력받는다. 입력받은 학번을 통해 학생 시퀀스 노드로 접근이 가능하고, 각 학생 시퀀스 노드에서 학생 노드로 접근이 가능하다. 각 학생 노드에서 각 매개 노드로 접근이 가능하고 각 매개 노드에서 도로 노드로 접근이 가능하다. 각 도로 노드에서 두 학생이 연결되어 있나 확인이 가능하다. 각 학생이 연결된 매개 노드를 각각 탐색하면(Min(deg(v), deg(w))만큼 탐색하게 된다.

**student\_sequence\_node\* search\_stduent\_by\_id(int);**

학번으로 학생시퀀스노드 찾기 O(n)

학생 시퀀스 노드의 첫번째로 접근하여 학생 노드에 접근한다. 입력받은 학생의 학번과 같을때 까지 학생 시퀀스 노드를 따라가며 선형탐색한다.

**road\_sequence\_node\* search\_road\_by\_id(int);**

학번으로 도로시퀀스노드 찾기 O(n)

도로 시퀀스 노드의 첫번째로 접근하여 도로 노드로 접근한다. 입력받은 도로 번호와 같을때까지 도로의 시퀀스 노드를 따라가며 선형탐색한다.

**인터페이스 및 사용법**

인터페이스는 올려주신 2015-2 자료구조 설계 과제 3.PDF에 있는 인터페이스와 동일하게 만들었습니다. 실행되었을 때 화면에 아무것도 나오지 않으나 이때부터 입력을 받습니다. 예를 들어 data.txt를 입력하면 같은 폴더 내에 있는 data.txt의 내용을 파일 입력을 통하여 읽어옵니다. 읽어온 데이터를 기반을 둬 그래프를 구축합니다. 예를들어, 이러한 입력들을 바탕으로 A 41795908 poker@game.uk을 입력하면 21을 출력합니다. 올려주신 PDF의 입력 예시, 출력 예시가 완전히 동일하며 출력 결과도 정상작동합니다. 공지사항으로 올려주신 테스트케이스로 테스트 결과 정상작동합니다. 결국 한마디로 인터페이스는 올려주신PDF 파일과 완전 동일합니다.

**Ps.1) 제 컴퓨터에서는 visual스튜디오 세팅으로 인해서인지 원인모르는 버그때문에 Visual studio로**

**ctr + F5로 실행했을때 파일명을 입력받으면 경로를 제대로 잡아내지 못합니다. Vsual studio로 ctr + F5로 빌드, 컴파일 후에 해당 폴더에 접근해서 .exe파일을 실행해 테스트 바랍니다. 이 부분에 대해서는 그 외 특이사항에 정리하겠습니다.**

**Ps2) 파일 입력으로 작동하면 조금 다릅니다. 파일 입력을 받으면 System(‘cls’)를 하여 콘솔창을 지운 부분조차도 파일에 입력되기 때문에 insert\_student함수와 insert\_road 함수에 포함된 cout함수가 그대로 실행됩니다. 따라서 1,2,3,4,5 … 총 학생수와 1,2,3,4,5 … 총 도로수가 그대로 파일에 찍혀나옵니다. 그 부분 이후로는 테스트케이스의 결과와 완전히 동일합니다**

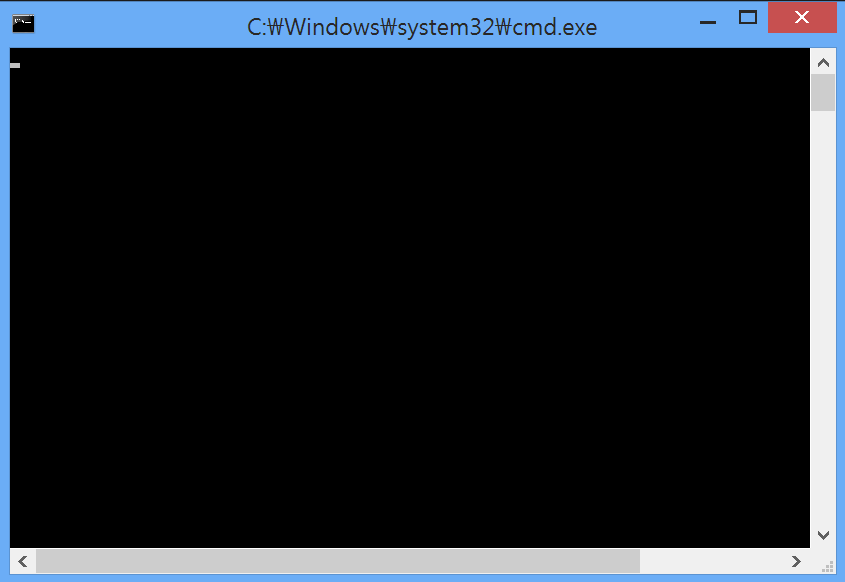


그림 1인터페이스는 없지만 이때부터 입력받음

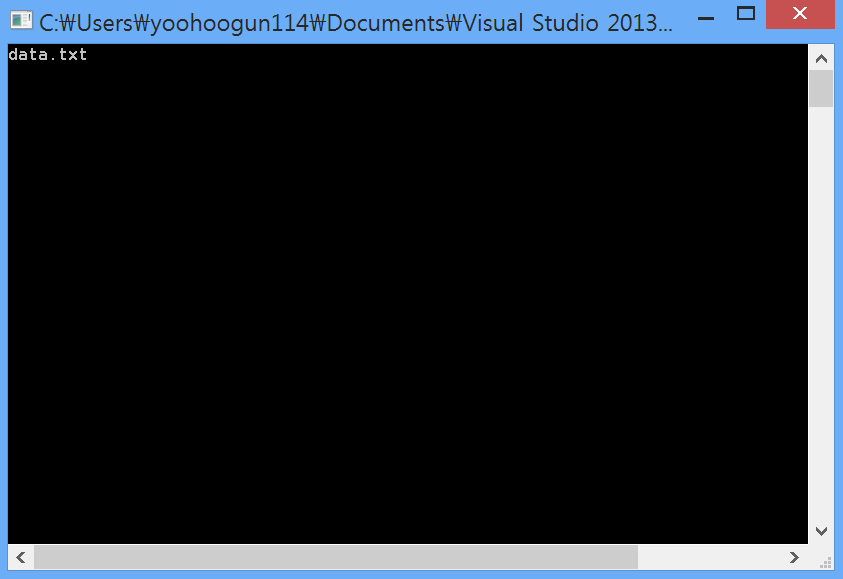


그림 2 data.txt를 입력해서 해당파일에 있는 data를 기반으로 그래프 구축

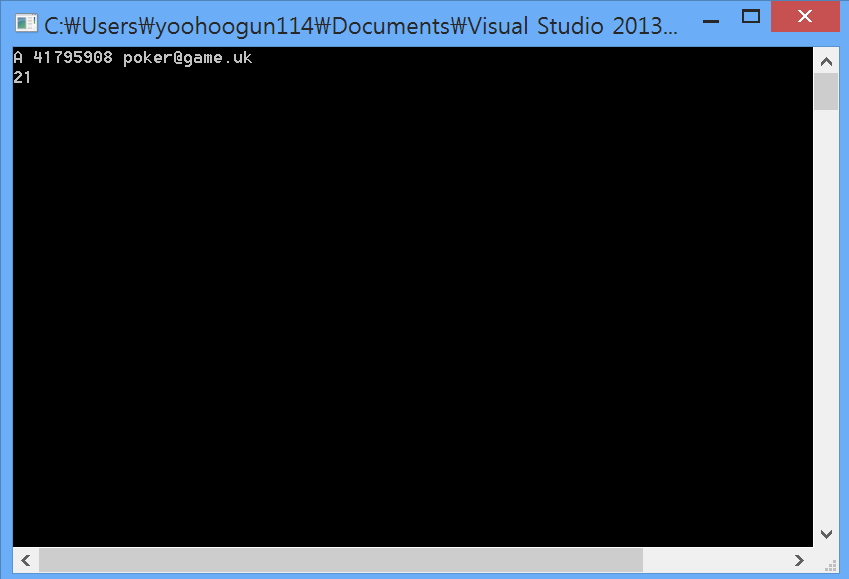


그림 3 학생 추가

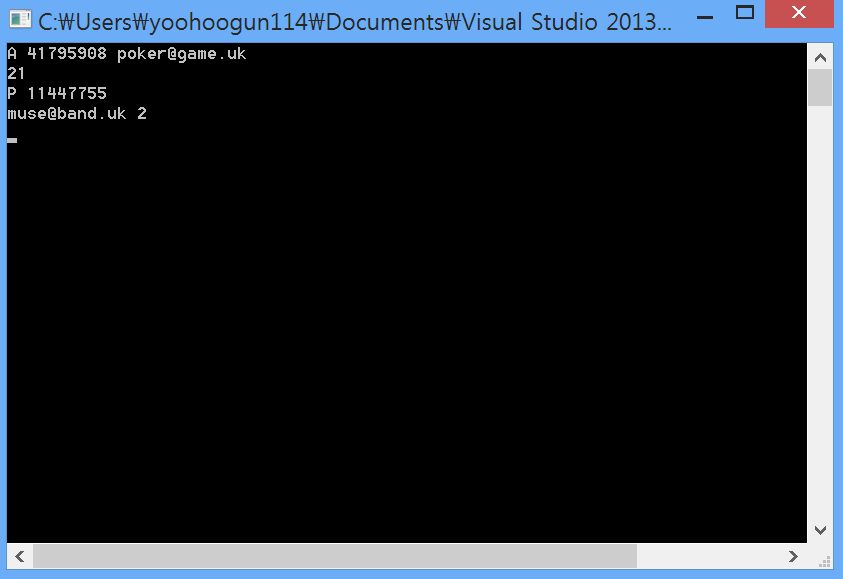


그림 4학생 정보 출력

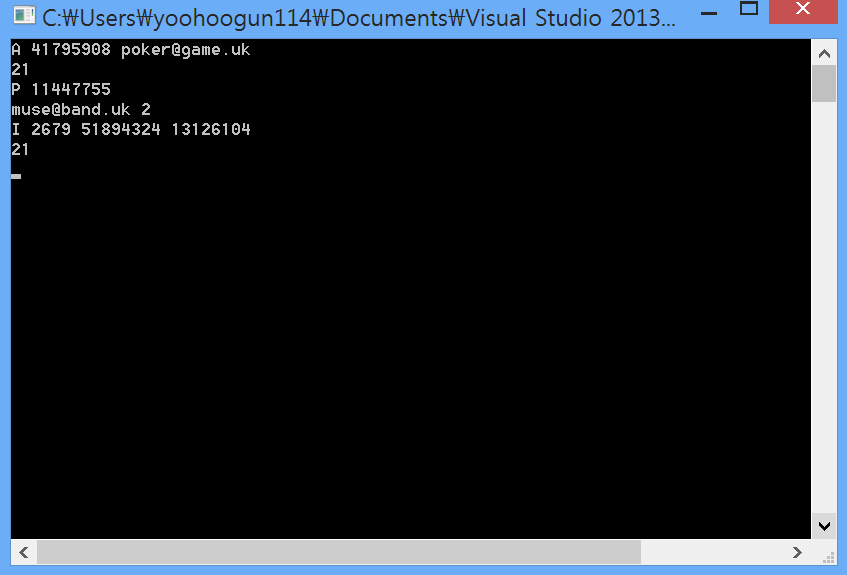


그림 5도로 추가

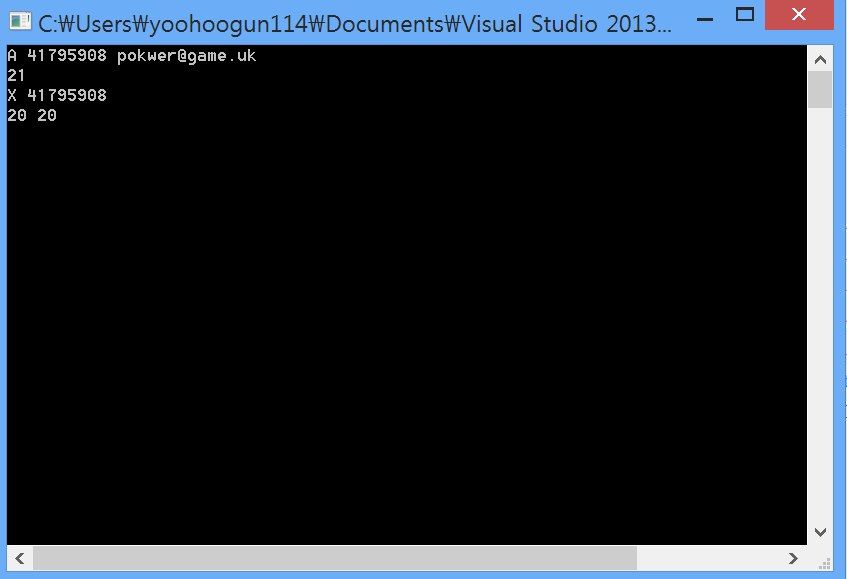


그림 6학생 삭제

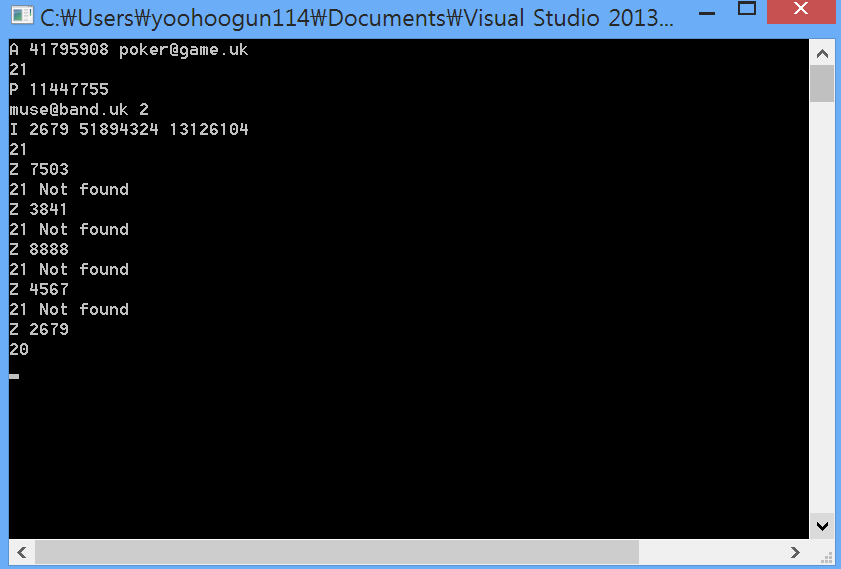


그림 7도로 삭제

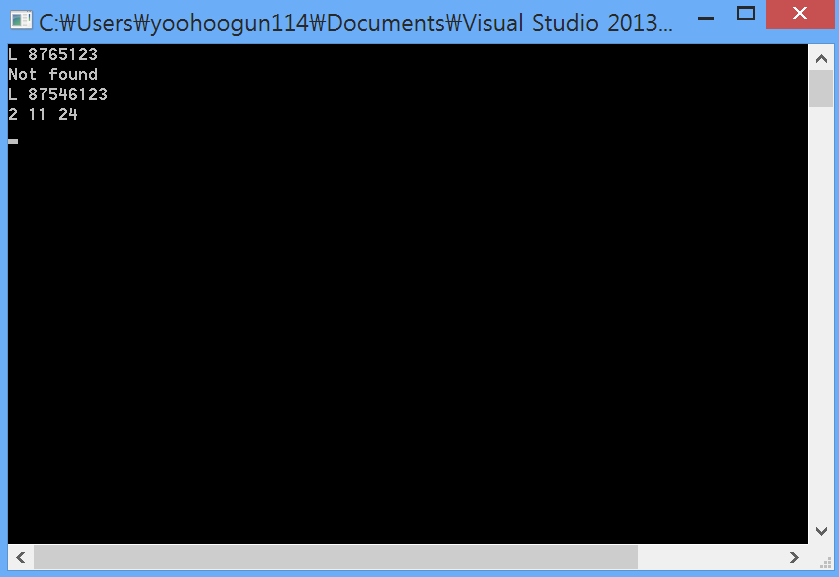


그림 8연결된 도로 출력

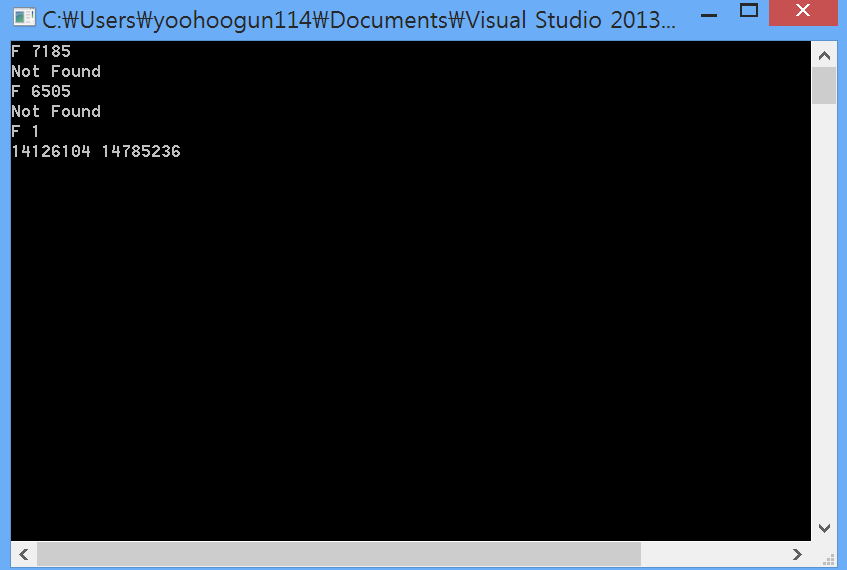


그림 9도로정보 출력 1

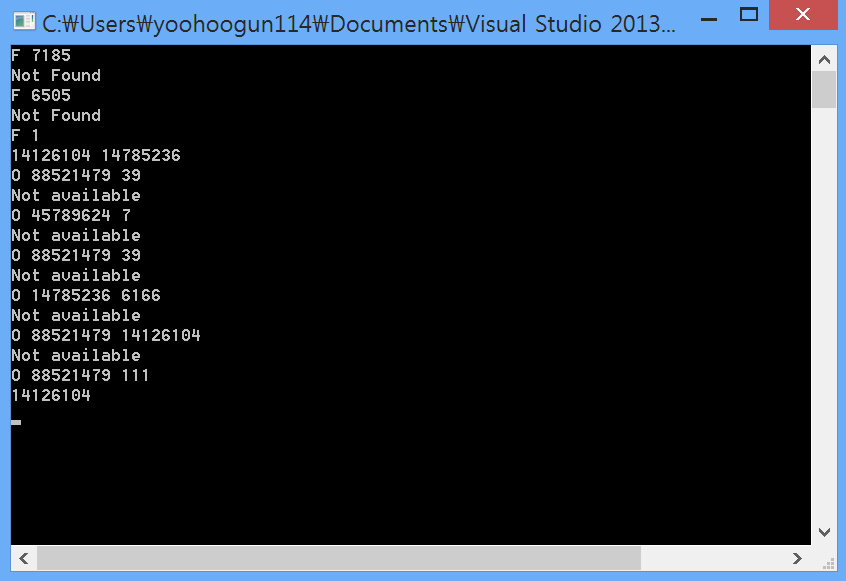


그림 10도로정보 출력 2

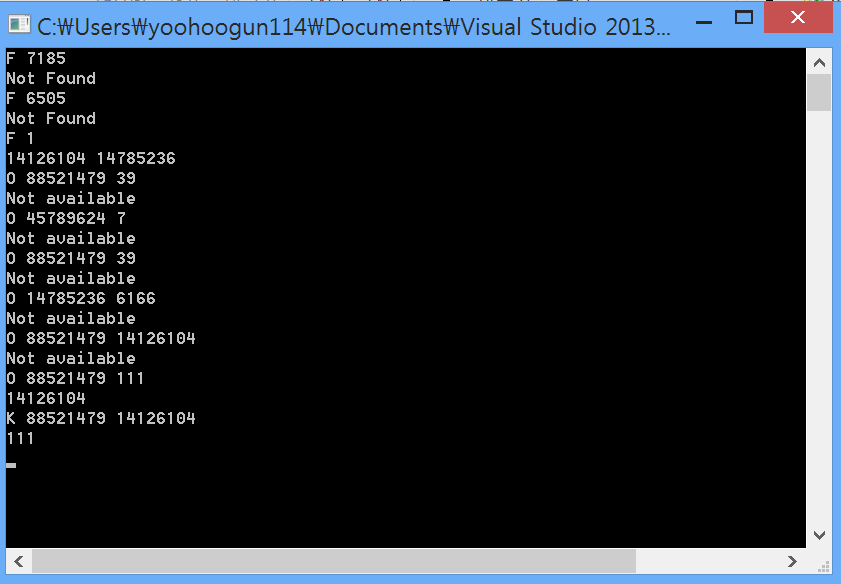


그림 11도로정보 출력 3

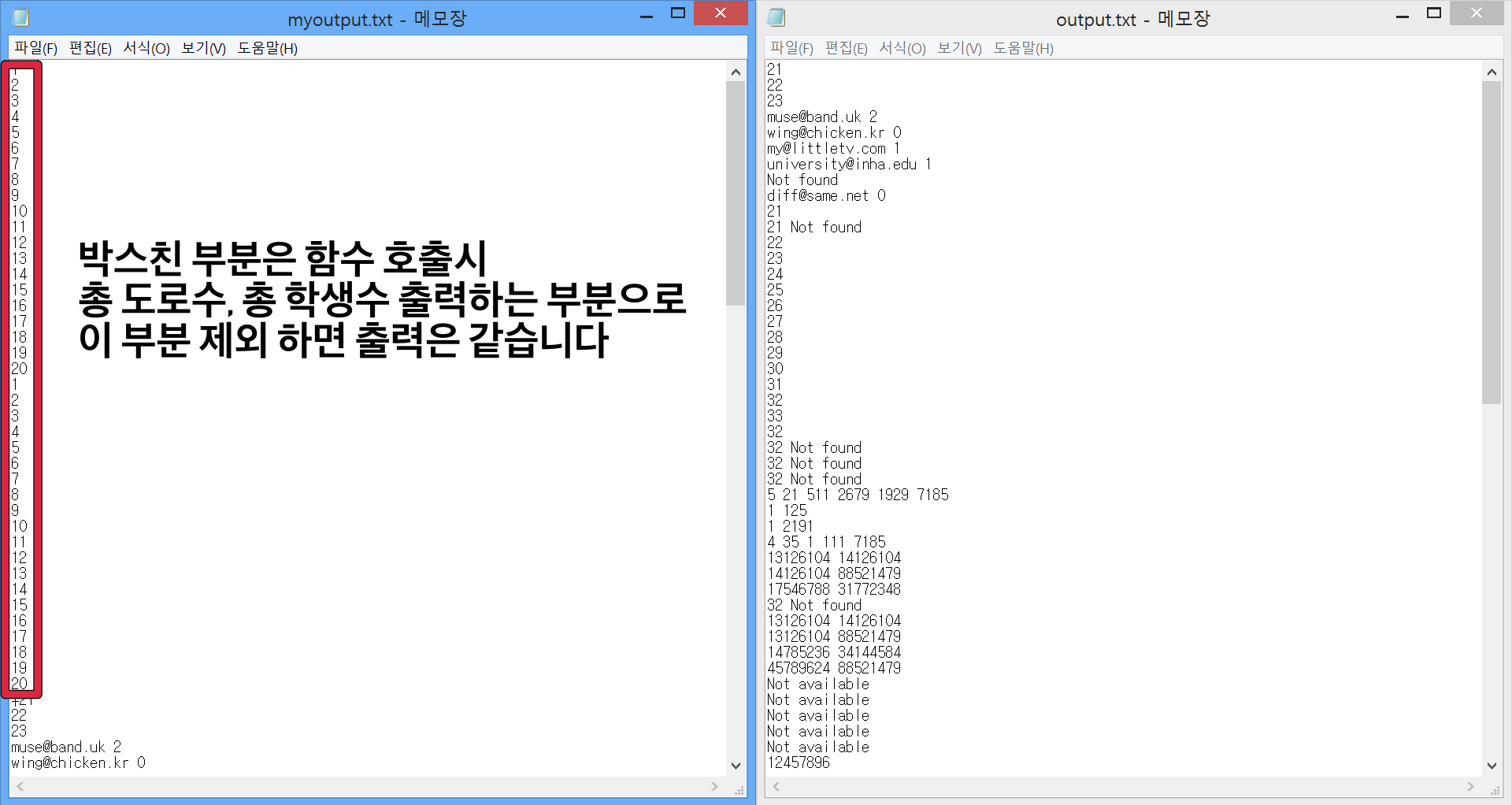


그림 12왼쪽이 제 프로그램, 오른쪽이 정상출력

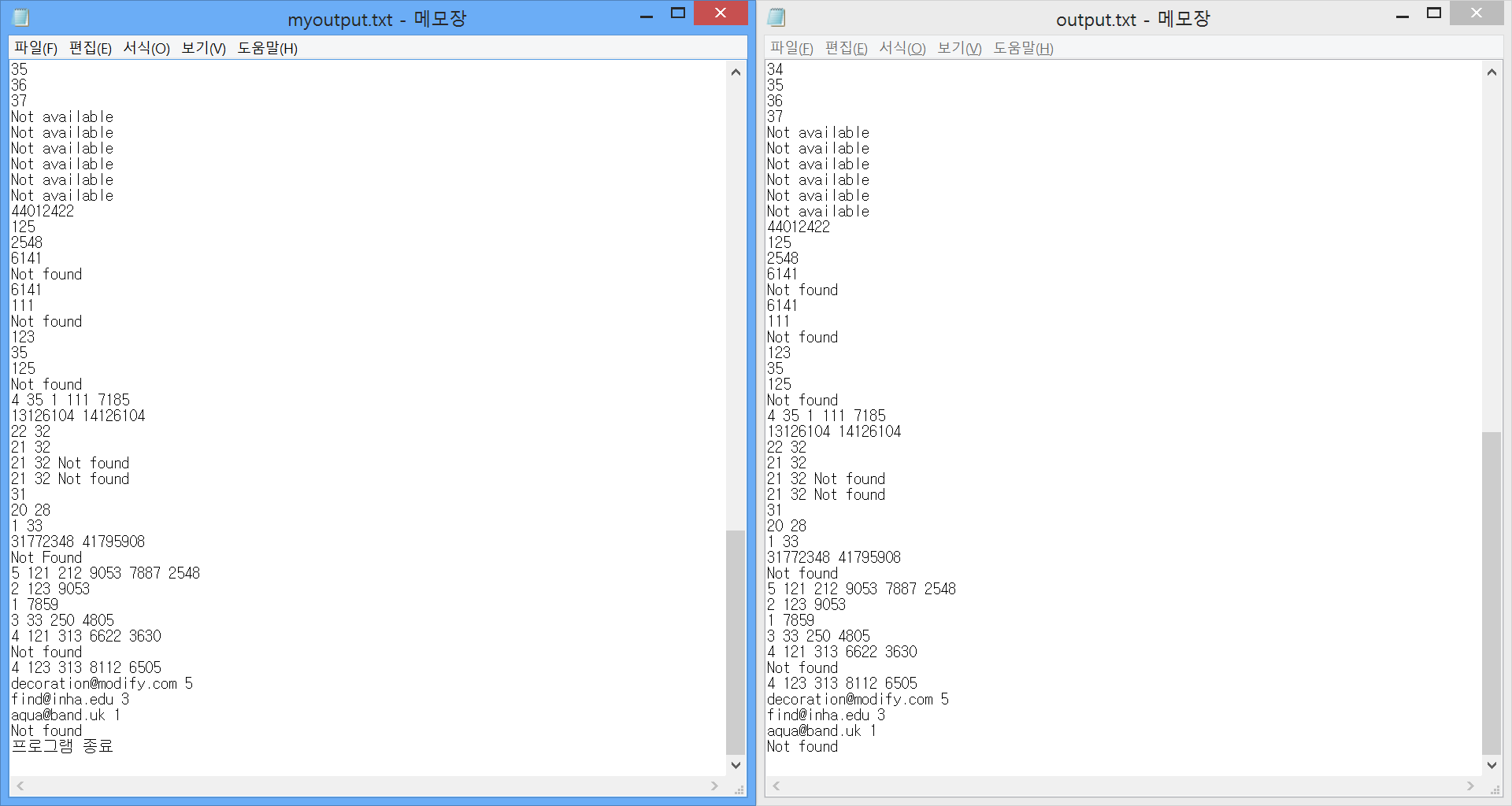


그림 13왼쪽이 제 프로그램, 오른쪽이 정상출력

Q를 입력하면 프로그램 종료

**평가 및 개선 방향**

- 본 결과의 장점 및 단점 - 향후 개선 방향

장점: 본 프로그램을 Adjacency List Structure를 이용해 구현하였기 때문에 Vertex(학생), Edge(도로) 의 삽입 시간이 O(1)시간으로 매우 빠른 퍼포먼스를 보여준다. **void print\_available\_incident(int, int);** 함수를 실행하면 각 road node struct에 첫번째 연결된 학생과 두번째 연결된 학생이 데이터로 저장되어 있기때문에 빠른 수행시간O(1)을 가진다.

Incident, adjacent도 각각 O(deg(v)), O(Min(deg(v),deg(w)))로 나쁘지 않은 수행시간을 보여준다. 이는 각각의 struct를 시퀀스로 연결했기 때문에 삽입과 삭제가 간편하고 빠르기 때문이다. (사실 삭제 삽입과정 자체는 복잡하지만 노드를 생성하고 연결하는 작업이기 때문에 배열을 밀고 당기는 것 보다는 편하고 빠르다.)

단점,개선방향: 코드를 다 짜고나서 보니 시퀀스의 마지막 노드를 찾는 부분이 선형 탐색으로 O(N)이 걸린다. 만약 class에 road, student sequence의 Head, Tail을 각각 설정했다면 처음과 끝으로 접근하는데 O(1)시간이 걸리므로 훨씬 더 빠른 입력 수행시간을 보일 수 있을 것 같다. 또한 학번, 도로명을 입력받아 시퀀스들을 선형탐색 하는데 O(N)이 걸리게 된다. 이는 그다지 빠른 수행속도가 아니다. 더 빠르게 탐색하려면 Adjacency Matrix Structure 구조를 이용하면 될 것이다. Adjacency Matrix Structure를 이용하면 isAdjacentTo와 같은 메서드를 상수시간에 수행할 수도 있기 때문에 이러한 연산을 자주 이용하고 삽입 삭제가 적은 프로그램이라면 Adjacency Matrix Structure로 구현하는 편이 더 효율적이다.

프로그램을 작성하면서 struct를 5개로 나누니 상당히 신경쓸 부분이 많았다. 포인터들도 서로가 서로를 가리키게 되니 삭제 부분에서 상당히 디버깅 하는데 애를 먹었다. 만약 다시 그래프를 작성하게 되면 student sequence 부분과 road sequence 부분을 각각 student와 road부분에 통합하여 포인터로 가리키면 더 편할 것 같다.

매서드가 많은 큰 프로그램이 되다 보니까 각각의 메서드에 중복된 연산, 흐름이 들어가게 되었다. 코드를 리뷰하면서 깔끔히 정리하고 중복된 코드를 막는법을 구상해보아야 겠다.

**그 외의 특이사항이 있으면 정리.**

1. 이번에도 역시 맥 Xcode에서 작업후 윈도우 Visual Studio로 코드를 옮겨서 디버그 작업을 했는데 맥의 XCODE에서는 잘 동작하던 코드가 Visual Studio에서 Ctr + F5를 통해 빌드하니 파일이름을 입력했을때 파일 이름을 정확히 입력했음에도 불구하고 파일이 계속 없다고 출력되었다. 이상한 점은 비쥬얼 스튜디오에서 ctr + F5를 통해 실행하면 파일 이름을 입력했을때 없다고 출력되었지만 빌드되어 실행된 파일 디렉토리로 들어가서 빌드된 .exe파일을 직접 실행하니 정상적으로 실행되었다. 무척 당황하여 구글링을 해보니

<http://stackoverflow.com/questions/798798/ifstreamopen-not-working-in-visual-studio-debug-mode>

라는 글을 검색하게 되었다.

자세한 내용은 개인 블로그 <http://dev-timeros.tistory.com/69> 에 자세히 정리해 두었다.