**알고리즘 2차 설계과제 보고서**

**회원 커뮤니티 관리 프로그램 설계**

**12131469김경수**

**010 8030 3206**

**wlsl4239@naver.com**

**1. 개요**

- 설계 목적 :

파일에 저장 된 회원 정보와 Follow 정보를 읽어 와 유향 그래프를 인접 리스트 방식(adjacency list representation)으로 모델링하고 강결합 요소(Strongly Connected Components)를 사용하여 회원들 간의 Follow 정보를 관리하는 프로그램을 구현한다.

- 요구 사항 :

1) 파일에 저장 된 회원 정보와 Follow 정보를 읽어 와 유향 그래프를 인접 리스트 방식으로 구축한다.

2) 질의 ‘T’ : 입력받은 회원이 포함된 그룹의 리더 정보를 출력한다.

3) 질의 ‘A’ : 첫 번째 회원으로부터 두 번째 회원으로 가는 Follow 상태를 변경한다. Follow 상태가 아니라면 첫 번째 회원은 두 번째 회원을 Follow 하게 되고, Follow 상태였다면 Unfollow를 하게 된다.

4) 질의 ‘0’ : 현재 그래프에서의 SCC 개수를 출력한 뒤 프로그램을 종료한다.

- 프로그래밍 환경 :

1) OS : Windows 10, Mac OS

2) 언어 : C++

3: 개발 환경 : Visual Studio 2017, XCode

**2. 필요한 자료구조 및 기능**

int scc[5001];

: 회원이 가입된 그룹의 leader를 저장하는 배열이다.

bool visited[5001];

: 정방향 dfs의 visit 여부를 저장하는 배열이다.

bool rvisited[5001];

: 역방향 dfs의 visit 여부를 저장하는 배열이다.

pair< int, pair<string, string> > people[5001];

: 회원정보를 저장하는 배열이다. first : 회원번호, second.first : 이름, second.second : 연락처

vector<int> digra[5001];

: 정방향 그래프. 실제 그래프 정보

vector<int> rdigra[5001];

: 역방향 그래프. SCC를 구할때 사용한다.

stack<int> stac;

: 정방향 그래프의 DFS를 수행한 후 수행이 완료된 vertex부터 저장하는 stack.

int numberOfSCC;

: SCC의 수를 저장하는 변수.

int n, m;

: 정점, 간선의 수

**3. 기능별 알고리즘 명세**

- 기능을 위한 알고리즘 설명(시간 및 공간복잡도)

void dfs(int now)

: 정방향 DFS를 수행하며 visited[now]를 true로 설정해주고 수행이 완료된 vertex부터 stac에 push한다.

Vertex의 수를 n, edge의 수를 m이라고 할 때, 정방향 그래프의 vertex와 edge를 모두 확인해야 하기 때문에 시간복잡도는 **O(n+m)time**이다.

DFS를 수행 하는 데 공간 복잡도는 그래프를 구성하는 vertex와 edge를 합한 O(n+m) space이며 stac에 모든 vertex를 push하므로 여기에서 O(n) space가 필요하다. 따라서 공간복잡도는 **O(n+m) space**이다.

void rdfs(int now, int leader)

: 역방향 DFS를 수행하며 rvisited[now]를 true로 설정해주고 수행이 완료된 vertex의 leader를 rdfs함수를 호출한 vertex로 설정한다.

Vertex의 수를 n, edge의 수를 m이라고 할 때, 역방향 그래프의 vertex와 edge를 모두 확인해야 하기 때문에 시간복잡도는 **O(n+m)time**이다.

역방향 DFS를 수행 하는 데 공간 복잡도는 그래프를 구성하는 vertex와 edge를 합한 O(n+m) space이며 scc배열의 모든 vertex에 대해 leader를 설정해야 하므로 O(n) space가 필요하다. 따라서 공간복잡도는 **O(n+m)space**이다.

void func()

: 실제로 SCC를 구하는 함수이며 scc의 수와 visited배열, rvisited배열을 초기화 하는 과정도 포함되어 있으므로 초기화 과정에 O(n)time이 필요하다.

초기화 후 dfs함수를 수행하고, stac의 top부터 rdfs를 수행한다. 이 때 rdfs를 호출한 번호가 rdfs를 수행하며 방문할 모든 vertex의 leader가 되며 rdfs가 수행될 때 numberOfSCC 변수의 값을 1 증가시킨다. 수행할 번호가 이미 방문되었다면 rdfs를 수행하지 않는다. 그 후 stac에서 pop을 수행한다.

Dfs와 rdfs함수를 수행하기 때문에 각 각 O(n+m)time이 필요하다. 따라서 총 시간 복잡도는 **O(n+m)time**이다.

scc의 수를 저장하는 변수에서 O(1) space가, visited배열과 rvisited배열에서 각각 O(n)space가 필요하며 dfs, rdfs함수를 수행하기 대문에 각 각 O(n+m)space가 필요하다. 따라서 총 공간 복잡도는 **O(n+m)space**이다.

main함수

: 파일 명을 입력 받은 후 파일로부터 회원 정보를 입력 받는다. n명의 회원이 있기 때문에 회원 정보를 입력 받는데 O(n)time의 시간과 O(n)space의 공간이 필요하다. 그 후 회원들 간의 Follow정보를 입력 받으며 정방향 그래프와 역방향 그래프를 구축한다. m개의 edge가 있기 때문에 O(m)time의 시간과 O(m)space의 공간이 필요하다.

입력이 끝나면 func()를 수행하여 scc를 구한다. 이 때의 시간복잡도와 공간복잡도는 위에서 설명한대로 O(n+m)time과 O(n+m)space이다.

Scc를 구한 후 반복문을 돌며 질의를 처리한다.

질의 ‘T’

: 회원 번호를 입력 받은 후 scc배열에서 입력 받은 회원의 leader가 people배열의 몇 번째 인덱스에 있는지 찾아 index 변수에 저장한다. 그 후 index번의 회원 정보를 출력한다.

입력 받은 회원의 leader이 people배열의 어느 위치에 있는지 찾기 위해 people배열의 원소를 모두 봐야 하므로 질의 ‘T’를 수행하는 데 필요한 시간 복잡도는 O(n)time이며 공간 복잡도는 O(n)space이다.

질의 ‘A’

: e(회원번호 A)와 d(회원번호 B)를 입력 받은 후 digra[e]에서 d를 찾는다. d가 있다면 Follow상태이므로 flag를 true로 설정하고 index변수의 값을 설정해준다. Flag가 true라면 rdigra[d]에서 e를 찾아 rindex변수의 값을 설정해준다. Flag가 true라면 digra[e]에서 d를, rdigra[d]에서 e를 제거하고, Flag가 false라면 digra[e]에 d를, rdigra[d]에 e를 추가한다. 그 후 func()함수를 수행한 후 scc[e]를 출력한다.

digra[e]의 크기를 k, rdigra[d]의 크기를 l이라고 하면 digra[e]에서 d를 찾는 데 O(k)time과 O(k)space, rdigra[d]에서 e를 찾는 데 O(l)time과 O(l)space가 필요하다.

flag가 true라면 digra와 rdigra에서 제거하는 데 각 각 O(k)time과 O(l)time, O(k)space와 O(l)space가 필요하고 flag가 false라면 edge를 추가하는 데 O(1)time과 O(1)의 space가 필요하다.

그 후 func()함수를 수행하는 데 O(n+m)time과 O(n+m)space가 필요하다.

따라서 질의 ‘A’를 수행하는 데 필요한 시간 복잡도는 O(k+l+n+m)이고 이는 O(n+m)에 bound되므로 **O(n+m)time**이고 공간 복잡도 또한 O(k+l+n+m)이고 이는 O(n+m)에 bound되므로 **O(n+m)space**이다.

질의 ‘0’

: numberOfSCC를 출력한 후 프로그램을 종료한다. 시간 복잡도는 O(1)time이며 공간 복잡도는 O(1)space이다.

**4. 인터페이스 및 사용법**

- 입력 파일명을 입력한다. (캡쳐 화면에서는 list.txt)

- 질의를 입력한다.

질의 ‘T’

- 질의형식: “T S”

T: 임의의 회원이 포함된 그룹의 리더정보 조회 질의를 나타내는 기호

S: 회원번호

- 출력형식: “M N P”

M: S가 소속된 그룹 리더의 회원번호

N: S가 소속된 그룹 리더의 회원이름

P: S가 소속된 그룹 리더의 연락처

- 설명: 질의로 “T 10”이 주어지면, 10번 회원이 포함된 그룹의 리더 정보를 출력한다.

질의 ‘A’

- 질의형식:”A E D”

A: Follow 변경 질의를 나타내는 기호

E: 회원번호 A

D: 회원번호 B

- 출력형식: “R”

R: Follow 정보가 변경된 후, 회원번호 A가 소속된 그룹 리더의 회원번호



- 설명: 질의로 “A 1 2”가 주어지면, 1번 회원으로부터 2번 회원으로 가는 Follow 상태를 변경한다. 만약 Follow를 하고 있지 않았다면, <그림 1>과 같이, 1번 회원이 2번 회원을 Follow 하게 된다.

만약 Follow 상태라면, <그림 2>와 같이, 1번 회원으로부터 2번 회원으로 가는 Follow를 끊게 된다.

질의 ‘0’

- 질의형식 : “0”

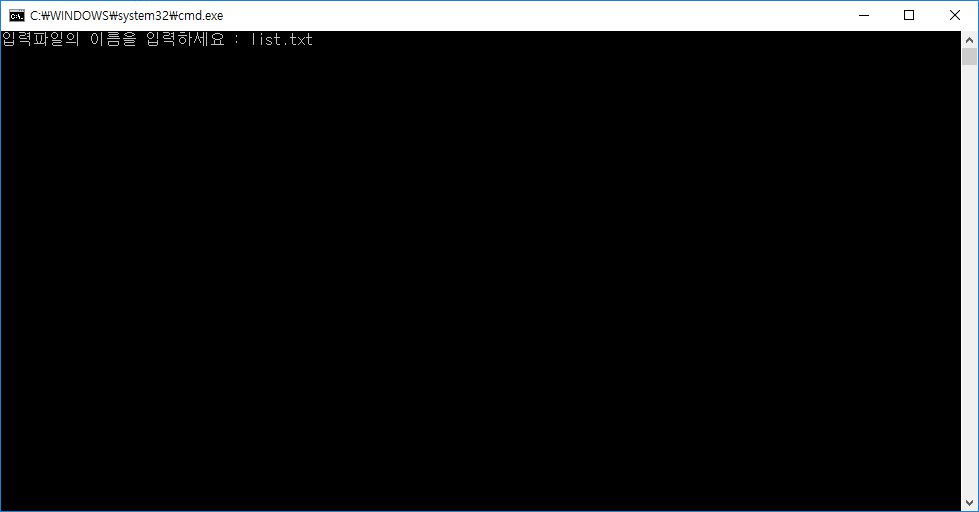
“0”: 프로그램 종료 질의를 나타내는 기호

- 출력형식: “C”

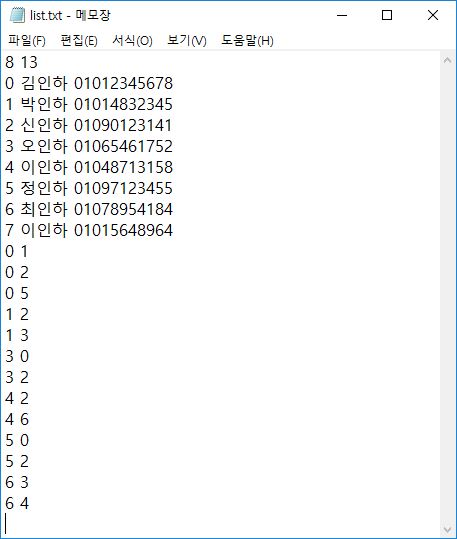
C: 현재 강결합 요소의 개수

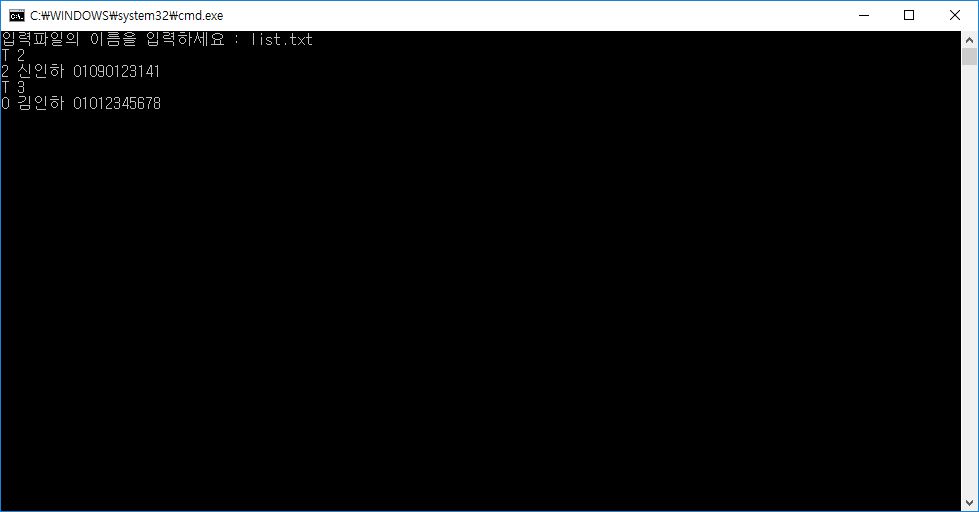
- 설명: 현재 그래프에서의 강결합 요소의 개수를 출력한 뒤, 프로그램을 종료한다.

- 실행 화면



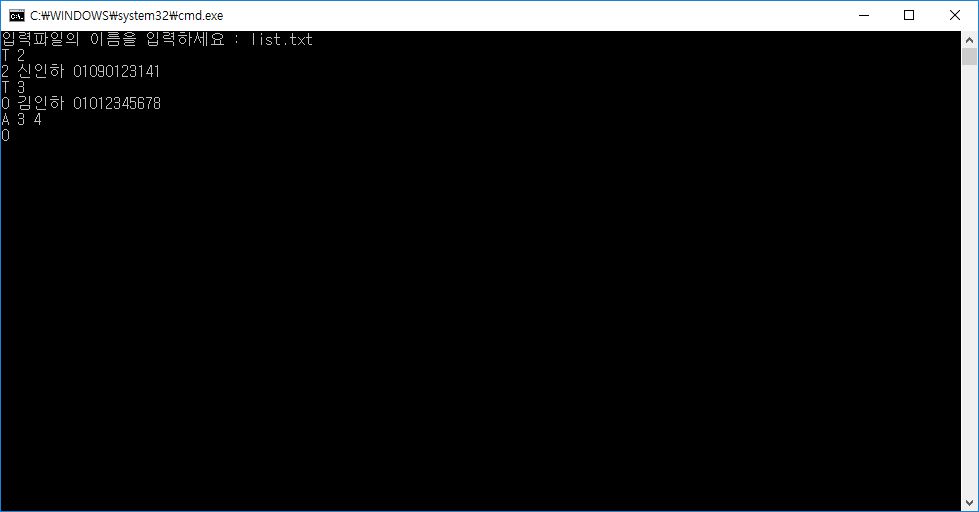
파일명을 입력한다. 사용한 list.txt파일의 내용은 아래와 같다.





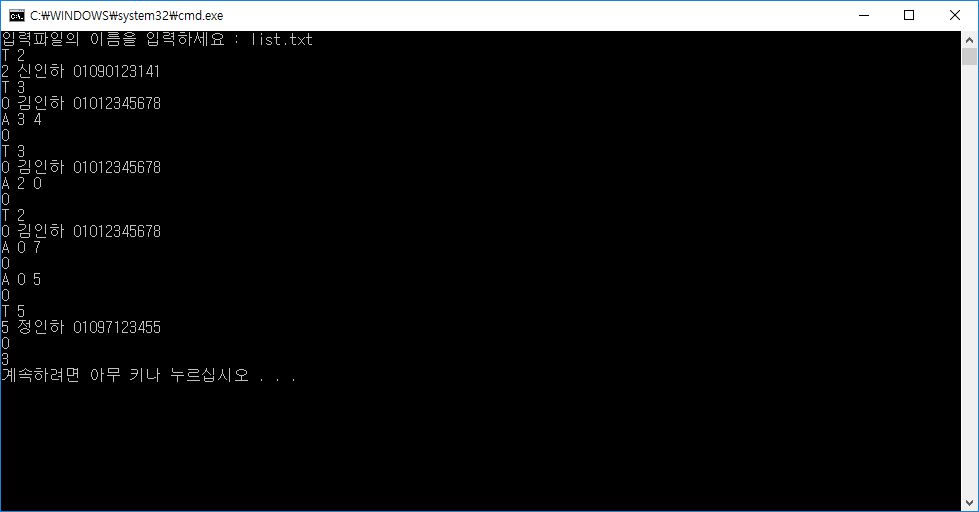
질의 “T”의 실행화면.

입력 받은 회원의 Leader의 회원정보를 출력한다.



질의 “A”의 실행화면.

3번 회원으로부터 4번 회원으로 가는 Follow 상태를 변경한다. 변경 후 3번 회원의 Leader번호를 출력한다.



질의 ‘0’의 실행화면.

SCC의 개수를 출력한 후 프로그램을 종료한다.

**5. 평가 및 개선 방향**

- 본 결과의 장점

수업시간에 배운 SCC를 구하는 방법대로 DFS를 수행한 후 수행이 완료된 vertex부터 stack에 저장하고, DFS가 끝난 후 stack의 top부터 역방향으로 DFS를 다시 수행하는 방식으로 SCC를 구했다.

그래프를 인접 리스트 방식으로 구현하는데 있어 vector STL을 사용했기 때문에 리스트방식임에도 index접근이 가능했다.

- 본 결과의 단점

회원정보를 담고 있는 people배열의 index와 회원번호가 다르기 때문에 질의 “T”를 수행할 때 people배열을 모두 확인해야 하는 단점이 있다.

질의 ‘A’를 수행할 때 Follow 상태였다면 Unfollow하는 과정에서 digra와 rdigra에서 간선을 삭제할 때 O(n)time이 수행될 수 있다.

- 향후 개선 방향

people배열이 pair< int, pair<string, string> >형 인데, index를 회원번호로 활용하고 pair<string, string>형으로 고친다면 회원번호로 바로 index접근이 가능해지고 메모리 또한 덜 사용할 것이다.

Vector가 아닌 List를 사용한다면 간선을 삭제할 때 O(1)time으로 수행할 수 있을 것이다.

**6. 기타**

- 특이 사항

- 1차 과제때와 마찬가지로 Mac과 Windows를 오가며 하는 게 비 효율적이었다.

- 수업시간에 배운 알고리즘 대로 잘 구현한 것 같다.