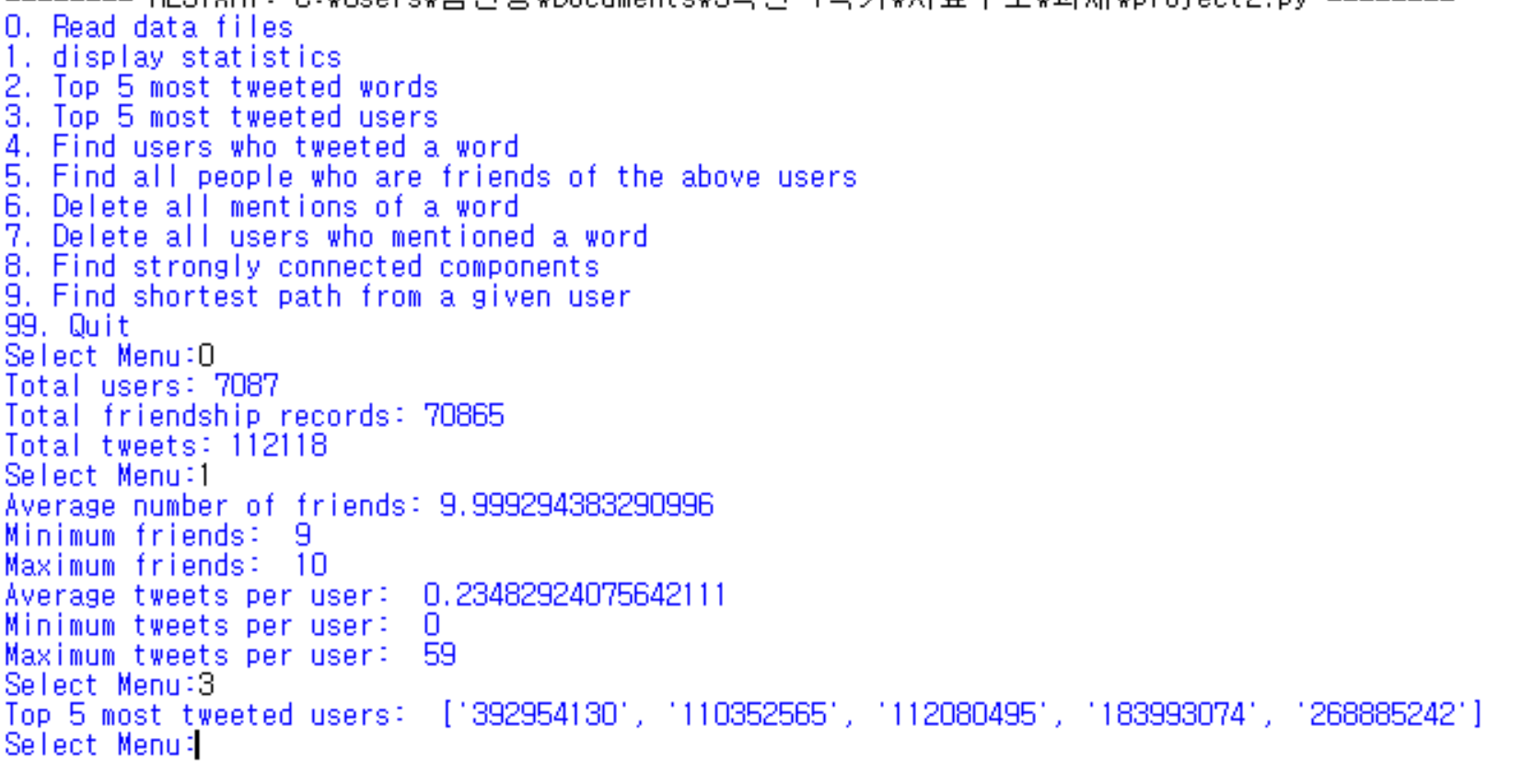
**자료구조 프로젝트**

**2014150031 김민강**

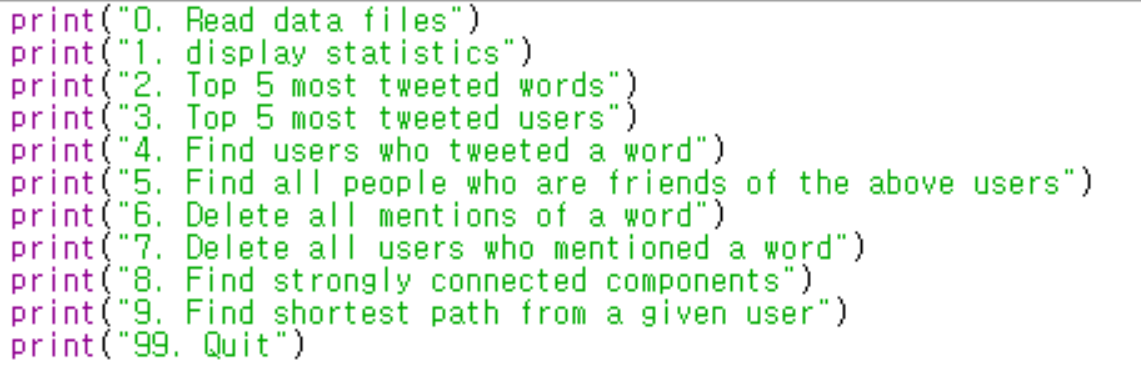
이 프로그램을 작성시 사용된 프로그래밍언어는 python이고 개발 환경은 python 3.5.1이다. 프로그램을 수행 시 다음과 같은 실행결과를 얻을 수 있다. 이 프로그램을 수행함으로써 총 세개의 텍스트 파일에서 사용자의 수, following하고 있는 관계의 수, 전체 트윗의 개수를 알고자 하였다. 또한 이에 더 나아가 평균 트위터 친구의 수, 최소 트위터 친구의 수, 최대 트위터 친구의 수, 가장 트윗을 많이 남긴 상위 5명의 사용자를 찾아내고자 하였다.

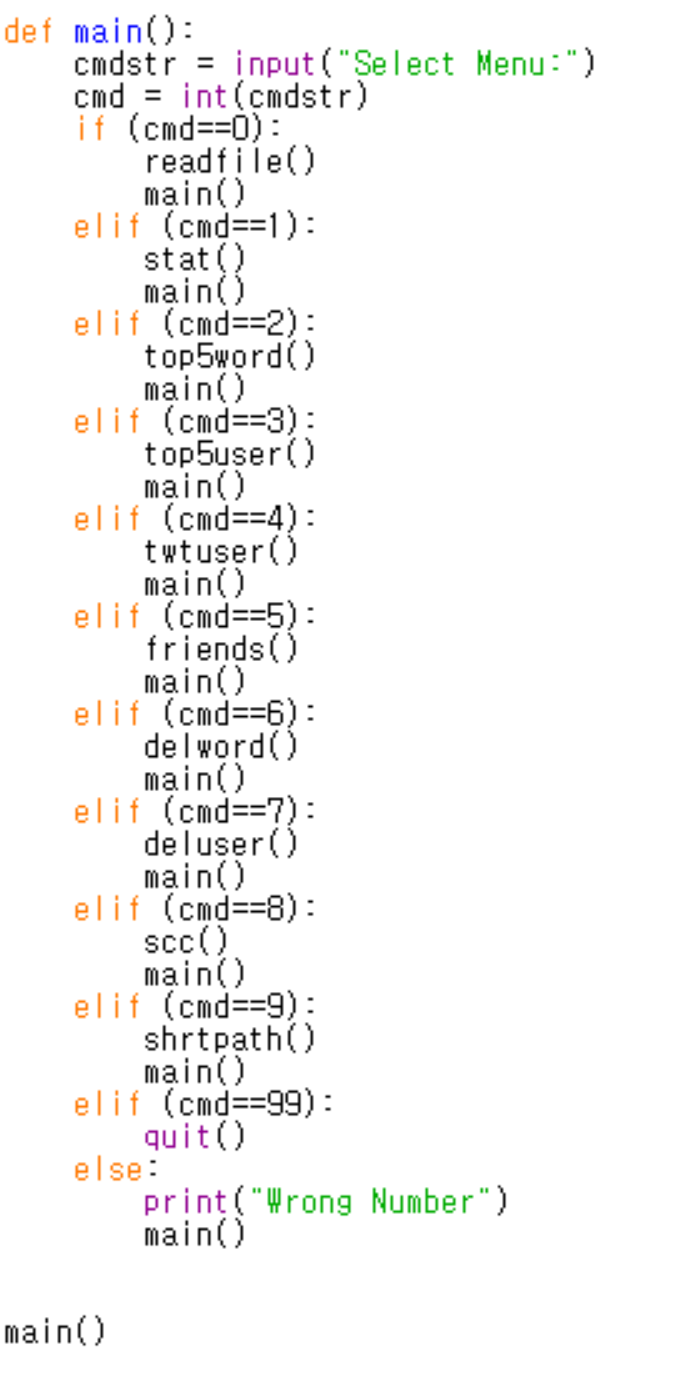
**실행결과**



**코드 설명**

**def main()**





사용자가 보고 싶은 내용을 보게 하기 위해서 main이라는 이름의 함수를 만들었다. 이는 사용자가 선택적으로 필요한 함수를 사용할 수 있도록 도와준다.

**def readfile()**

텍스트 파일 내의 content의 개수를 구하기 위해 만든 함수로 읽어드린 길이로 개수를 측정한다. P%==value를 이용해서 첫번째 줄의 경우 value가 0 두번째 줄의 경우 1, 이런 식으로 필요한 데이터를 선택적으로 읽어 들였다. 트윗을 남긴 시간은 위 프로젝트에서 중요하지 않으므로 이런 방법으로 데이터를 읽어 들이지 않고 필요한 데이터만 저장할 수 있었다. 단순히 데이터를 읽는 것일 뿐 따로 pointer가 필요하지 않기 때문에 배열을 사용했으며 시간 복잡도는 O(n)이다.

**def stat()**

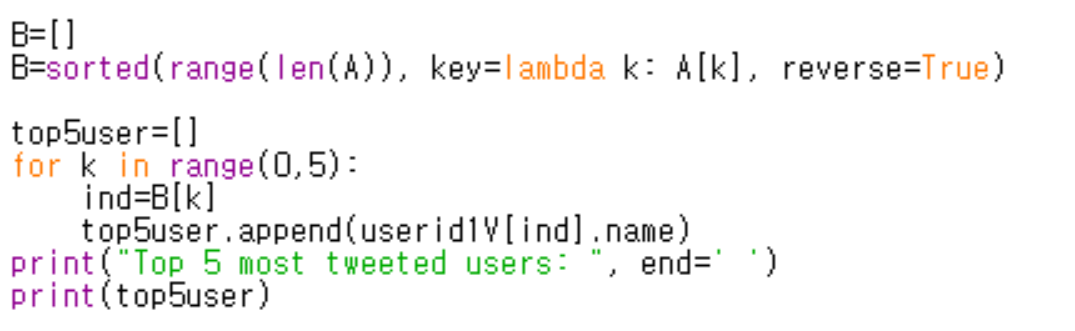
def readfile과 마찬가지로 텍스트 파일을 읽어 들인 후 리스트에 저장하지만 위의 코드와 같으므로 생략했다. 리스트의 이름 중에 1이 붙은 리스트는 중복 없이 한번만 사용된 리스트를 가리킨다. 즉 word1의 경우 여러 번 사용된 단어일 경우에도 이 리스트에는 한번만 들어가 있다. Word의 경우에는 n에 리스트 내에서의 순서를, name에는 단어를 input하지만 following의 경우에는 n에는 사용자를, name에는 사용자를 follow하는 사용자를 input한다.

친구의 수, 트윗의 수, 가장 트윗을 많이 한 사용자 5명을 찾기 위해 필요한 단계로 친구의 수에 관련된 정보는 cnt, 트윗의 수에 관한 정보는 cnt1, A를 통해 알 수 있다.

사용자 간의 관계, 단어 리스트를 만드는 데 걸리는 시간은 각각 O(n)이다. 그 후 중복되지 않은 사용자 list와 사용자 간 친구 관계 데이터의 윗줄이 같을 동안 해당 list에 있는 사용자에 친구 관계 데이터 아랫줄 사용자를 pointer로 연결시킨다. 이 때 걸리는 시간은 O(n)이다. 또한 프린트를 하는데 걸리는 시간은 O(1)의 constant time이므로 이 기능의 총 시간은 O(n)이다.

**def top5user()**

위와 매우 유사하나 사용자 당 트윗의 개수를 구하는 코드 부분에 사용자의 id 가 같으면 미리 만들어 둔 리스트에 append하는 것을 통해 word1에 있는 사용자 순서대로 트윗을 남긴 개수가 리스트로 기록 되도록 만들었다. 그 후 B라는 list를 만들어서 트윗의 개수를 큰 순서대로 원래 인덱스 값을 기록하게 한 뒤 인덱스 값을 이용하여 가장 많이 트윗한 사람을 구했다. 이는 O(n)의 시간 복잡도를 가지고 있으며 binary tree로 찾으면 O(logn)이다. 하지만 구현하지 못했다.



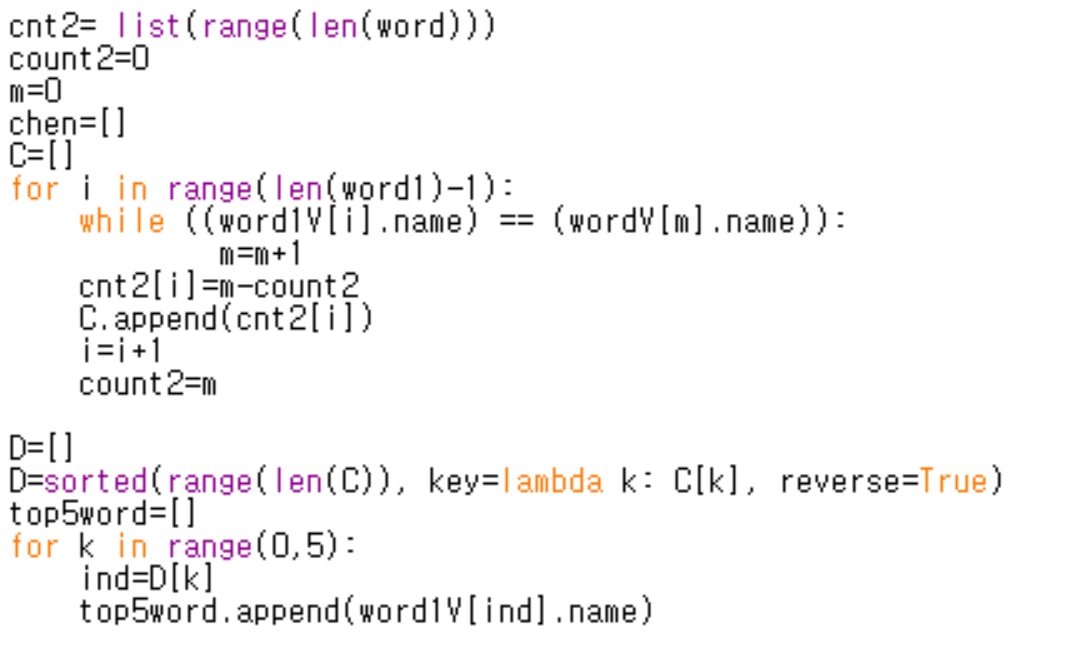
**선택한 자료구조와 그 이유**

선택한 자료구조는 리스트리고 이를 선택한 이유은 다른 구조보다 만드는 것이 간단했기 때문이다. 하지만 이는 작은 데이터일 경우에는 큰 차이가 없지만 큰 데이터를 다룰 경우에는 속도가 O(N)으로 (구현한 소스코드 내에서는) 실행되는데 꽤 오랜 시간이 걸렸다.

**개선하고 싶은 점**

처음에 프로그램을 구현하고자 목표하였던 것과는 달리 파이썬에 익숙하지 못해서 오류 수정을 하는 데에 많은 시간이 걸렸다. 그렇기 때문에 한학기 동안 배운 것을 사용하기 보다는 그냥 프로그램을 짜는 데에 급급해서 어떤 것이 더 속도를 줄일 수 있는지 더 좋은 코드인지에 대한 고민을 하지 못했던 것 같다. 지금 만든 코드는 리스트로 작성해서 속도가 느리므로 이 프로그램을 개선하기 위해서는 해쉬나 BST를 사용하는 등 속도 개선을 위한 자료구조를 사용하는 것이 좋을 것 같다. 또한 하나의 def 안에서 너무 많은 것들이 이루어져서 전체적으로 코드가 난잡한 것 같은 느낌이 들었다. 이를 개선하기 위해선 적당한 길이의 def 여러 개로 나누는 것이 필요해 보인다

**Top5 most tweeted word**



위는 가장 많이 트윗한 단어의 개수를 구하기 위해 만든 코드이지만 단어가 리스트 word에서 정렬이 되어있지 않아서 하나만 읽고 while문을 벗어나 개수의 측정이 안되었다. 이를 해결할 방법으로 sorting을 고려해보았는데 str.lower로 sorting을 하려고 보니 string이 아니라고 뜨고 vertex끼리는 비교가 안된다고 계속 오류가 났다. Word의 타입을 확인했을 때는 list였는데 왜 Vertex라고 하는지 찾아보다가 결국 구현하지 못했다. Sort라는 함수에 대해서도 이해가 부족해서 생긴 결과라고 생각한다. 이에 대해 파이썬 내장함수에 대한 공부가 이루어지면 더 좋은 코드를 만들 수 있을 것이라고 예상된다.

**구현하지 못한 코드에 대해 개선할 점**

**1. Find users who tweeted a word**

각 word를 언급한 사람들을 연결시켜야 하므로 singlely list로 구성한다. 특정 word 별로 그 word를 사용한 사용자를 list로 구성한다. list를 구성하는 시간은 O(n)이며 word를 찾을 때 중복되지 않은 배열을 이용해 찾으면 O(n), binary tree로 찾으면 O(logn)이다. 하지만 구현하지 못했다.

**2. Find all people who are friends of the above users**

친구관계를 정의해야 하므로 list를 이용한다. 단어를 언급한 사용자를 찾는 것과 마찬가지로 사용자를 찾을 때 배열로 찾는 다면 O(n), binary search tree로 구성된 사용자 구조를 이용해 찾게 되면 O(logn)이다. 그러나 구현하지 못했다.

**3. Delete all mentions of a word**

word를 Red Black Tree로 구성하고 입력 받은 단어를 지운다. 그러나 구현하지 못했다.

**4. Delete all users who mentioned a word**

word를 지울 때 연결된 사용자를 읽고, 사용자를 binary tree에서 찾는다. 그러나 구현하지 못했다.

**5. Find strongly connected components**

사용자 간의 관계를 이용해야 하므로 list를 사용하며 SCC를 찾아야 하므로 transpose해 실행한 DFS를 사용한다. 소요되는 시간은 O(n)이다. 그러나 구현하지 못했다.

**6. Find shortest path from a given user**

weight를 다 1로 둔 Dijkstra를 구현해 찾아본다. Dijkstra에는 min heap을 이용한다. 이 때 소요되는 시간은 min heap 구성시간에 따라 달라지며 최대 O(nlogn)까지 가능하다. 그러나 구현하지 못했다.