자료구조-2분반

2015410087 김지헌

-----------------------사용한 자료구조 설명----------------------------

먼저 각각의 사용자가 number로 되 있다는 것에 착안. Number(mod 1000)을 기준으로 hashTable을 만들어 사용자를 등록. 각각의 사용자는 자기 넘버, 자기 닉네임, 전체 친구 수, 전체 트윗 수, 트윗들에 대한 4중 연결 리스트, 친구에 대한 단순 연결 리스트를 가지고 있다.  
 (코드는 adj.py파일안에 사용자는 class USER로, 트윗들과, 친구들의 연결 리스트는 class Adj로 쓰여져 있다.)

단어들은 각각의 단어를 key값으로(기준으로) rbtree를 구성되어 있다. 각각의 노드는 트윗한 string값, 단어가 트윗된 횟수, 단어를 트윗한 유저에 대한 연결 리스트를 가지고 있으며, 유저에 대한 연결 리스트는 아까 위에서 말했던 4중 연결 리스트로 연결이 된다.  
 (코드는 rbtree.py파일 안에 노드는 class Node로, rbtree는 class Tweet으로 쓰여져 있다.)

4중 연결리스트는 상, 하, 좌, 우 에 대한 포인터를 가지고 있으며, 각각 좌, 우는 한 유저가 트윗한 단어들에 대한 연결 리스트가 되며, 상, 하는 그 단어를 트윗한 유저들에 대한 연결 리스트가 된다. 그래서 단어를 사용한 사람을 찾거나, 한 사람이 트윗한 단어를 체크할 때 링크드 리스트로 O(N)속도를 보여주게 된다.

그렇기 때문에, 통계를 보여줄때는 hashTable을 통해 전체 친구수, 전체 트윗수만 더해, O(N)의 시간복잡도를 보여주며,

가장 많이 트윗된 단어를 보여줄때는 rbtree의 서치를 이용해 O(N logN)의 시간

가장 많이 트윗한 사람은 hashTable을 통해 O(N)

단어를 찾을 때는 rbtree의 서치를 이용해 O(NlogN)과, 그 단어에 대해 링크드리스트로 O(N)의 시간을 보여준다.

친구를 찾을 때는 유저number를 저장해둔 배열을 통해, hashTable에서 유저를 검색 하고 O(N), 각각 유저들에 대한 친구 연결 리스트로 O(N)으로 출력을 한다.

Scc를 찾을 떄는 유저들의 친구 연결 리스트에 대한 DFS를 통해 SCC를 구하게 되며

Shortest path또한 dijstra알고리즘을 사용하게 된다.

----------------------고칠 점---------------------------

평가 기준에 Large size data test 에 대해 O(N log N)을 만족하라고 되어있지만, 공간복잡도에 대한 이야기는 없어서 공간복잡도를 완벽히 무시하고, 시간복잡도만 최대한 빠르게 하려고 하였다.

속도의 향상을 위해서, 같은 트윗에 대해, rbtree에도 노드가 존재하고, 연결 리스트에도 노드가 존재하게 되어, 시간복잡도는 대부분 O(N)안에 해결하려고 했지만, 공간복잡도가 엄청나게 큰 것 같다. 공간복잡도까지 생각을 하면서, 시간을 줄이는 방법을 찾아봐야 할 것 같다.

Rbtree에 대한 공부가 완벽하게 되지 않아, delete\_fix를 완성시키지 못했다. 그래서 현재는 각 Node에 대해 tweet된 count가 0이면 죽어있는 노드로 생각하게 만들어놨는데, delete\_fix를 완성시킨다면 더 좋은 자료구조가 되지 않을까 생각한다.