Report

2014140071 생명과학부 김민유

- What data structure you chose and why

  - User의 경우 **binary search tree**로 관리하였다. 평균적으로 시간 복잡도가 O(nlogn)이기 때문에 어느 정도의 속도를 보장할 수 있기 때문이었다. 그러나 잘 balance되지 않은 경우 시간 복잡도가 O(n^2)이 될 수 있다는 점이 단점이다. Red black tree를 쓰고자 하였지만 코딩이 처음이라 어려워 중도에 포기하고 binary search tree를 사용하였다.

  - Friend의 경우 User node에 달려 있는 **doubly linked list**로 관리하였다. 삽입과 삭제가 용이하고, 적정한 시간 복잡도를 보장해주기 때문이었다. 이에 더하여 FF(friend follower)라는 attribute를 user node에 추가해 삭제 시 follower 역시 관리하여 조금 더 쉽게 삭제할 수 있게 하고자 하였다.

  - Word의 경우 100개의 원소를 갖는 리스트를 기본으로 충돌할 경우 doubly linked list로 관리하는 **hash table**을 사용하였다. 리스트의 원소의 개수는 전체 word 수의 중근(root) 만큼을 사용하려 하였지만 알 수 없는 오류로 인해 100개로 바꿀 수 밖에 없었다. 이 때문에 hash table의 장점인 시간 복잡도가 제대로 구현되지 않았다.

- What is your expected performance

  - Read data files의 경우 user와 friend는 O(nlogn), word는 O(n^2)의 시간 복잡도를 보일 것이다.

  - Statistics의 경우 O(n)의 시간 복잡도를 보일 것이다.

  - Top 5 most tweeted words와 Top 5 most tweeted users의 경우 mergesort를 사용하여 O(nlogn)의 시간 복잡도를 보일 것이다.

- How would you improve the system in the future

  - User를 RBtree나 AVL tree를 사용하여 balance 시키고 word를 제대로 hashing 한다면 시간 복잡도를 대폭 줄일 수 있을 것이다.

- 기능을 전부 구현하지 못했다. 특히 Find all users who mentioned a word 부분에서 '₩n' 을 제대로 처리하지 못해 몇 시간을 헤맸으나 해결책을 찾지 못하여 뒷부분은 시도 조차 해보지 못 했다.
- 자료구조를 알고리즘에 더 적합하게 조금 더 깔끔하게 처리할 수 있다면 더 향상될 수 있을 것이다.