1. Read data files

User profile은 java에 구현된 HashMap<id, nickname>을 이용했다. Friendship은 HashMap<userID, Vertex>로 저장했다. 각 Vertex에 빠르게 접근하기 위해 그 Vertex의 userid를 key로 하는 hashmap 안에 vertex를 만들었다. Word tweet은 tweet을 key로하고 word를 value로 하는 HashMap<tweet, HashMap<user who used this word, number of a user who used this word>>과 그 반대인 HashMap<user, HashMap<word that this user used, number of use>>으로 저장했다. 또, 두 방법 모두 key에 해당하는 것의 총 사용횟수를 나중에 따로 계산하는 것을 피하기 위해 total을 미리 계산해줬다. 각각 Hash에 접근하는 시간은 O(1)이기 때문에 금방 파일을 읽을 거라고 예상했지만 의외로 시간이 걸리는 것 같았다.

1. Display statistics

거의 0번에서 미리 저장해둔 변수들을 사용했다. Min, Max tweets는 heap을 구성할 일이 있어서 그걸 사용했다. -> O(nlgn)

1. Top 5 most tweeted words

힙소트로 0에서 읽은 words를 key로 하는 해쉬를 total순으로 정렬하였다. ->O(nlgn)

1. Top 5 most tweeted users

2와 마찬가지로 user를 key로 하는 해쉬를 total 순으로 정렬하였다. -> O(nlgn)

1. Find user who tweeted a word

Word를 key로 하는 해쉬에 접근해서 그 안에 있는 user 정보를 불러옴 -> 만약 어떤 word를 말한 user가 상수만큼의 명수를 갖는다면 O(1)일 것이고 n명이면 O(n)일 것임. 하지만 한 사람이 거의 모든 단어를 말하는 경우는 별로 없으므로 O(1)로 생각

1. Find all people who are friends of above users

4번에서 뽑은 유저들의 정보를 ArrayList에 저장함. 여기서는 그 ArrayList를 불러와서 각 id를 key로 하는 friend 해쉬에 접근함. 거기서 adj를 불러와서 읽음. -> 만약 모든 유저에 대해 모든 유저가 모든 유저를 친구를 갖고 있다면 O(n^2)이 나오겠지만 실질적으로 생각하면 훨씬 작을 것임

1. Delete all mentions of a word

Word를 key로 하는 해쉬에 접근해서 그 안의 각 user들에 대해 이하를 반복 실행함. 그 user의 id를 key로 하는 해쉬에 접근해서 그 안에서 지울 word를 찾고 그 word를 지워줌. 어떤 word를 사용한 user의 수가 n이라면 O(n)이고 상수라면 O(1)일 것임

1. Delete all users who mentioned a word

User를 key로 하는 해쉬에 접근해서 그에 해당하는 word들에 대해서 이하를 반복 실행함. 그 word를 key로 하는 해쉬에 접근해서 각 user에 해당하는 value들을 모두 지운다. 또한 친구 목록, user profile에서도 모두 그 user를 지우고 처음 해쉬로 돌아와서 user를 지운다. 역시 O(1)이나 O(n)이므로 전체시간은 O(1)이나 O(n^2)

1. Find top 5 strongly connected components

미리 만들어 놓은 string-vertex 해쉬에서 구성한 트리를 사용함. Dfs를 실행하고 f 순서로 내림차순으로 정렬. 그리고 color와 parent를 모두 초기화 한 다음에 정렬된 순서대로 dfs를 실행하며 한 묶음이 나오는 것을 scc로 간주하고 그 중 가장 큰 5개를 뽑아 출력(직접 만든 그래프에 대해서는 실행이 되는데 주어진 샘플 파일을 이용하면 scc가 하나밖에 안 나와서 실행시키면 에러가 뜸) -> dfs : O(n), f sort: O(nlgn)(힙 정렬사용), scc\_find:O(n) 다 더하면 O(lgn)

1. Find shortest path from a given user

주어진 input 유저에 대해서 다익스트라 알고리즘을 시행하고(우선 순위 큐 사용) 그 유저가 갈 수 있는 모든 노드까지의 거리를 내림차순으로 출력한 후, 가장 큰 5개의 노드를 추려내고 그 유저까지의 경로를 출력함 -> 다익스트라 : O(nlgn) (힙사용하기 때문에,) 내림차순 : 퀵소트O(nlgn) -> O(nlgn)