**Report**

2015410006 배민근

1. What data structure you chose and why

저는 이 안에 총 3개의 구조가 있습니다. Graph, Hash Table, RBT. 먼저 그래프를 택한 이유는 user와 friendship을 표현하기 위함입니다. 각 유저는 numf라는 friend 수 변수도 가지고 있으면 각각의 node에 Adj라는 데이터 타입으로 edge들이 add 됩니다. 그리고 Hash Table과 RBT는 트윗을 관리하기 위함입니다. 일단 word.txt가 들어오면 hash 함수를 거쳐 key를 찾고 그 key값에 해당하는 것을 찾아 거기에 Adj형식으로 tweet이 add됩니다. 각 tweet의 내용을 담는 곳과 그 tweet을 한 사람들의 uid를 담는 배열을 포함합니다. 그리고 tweet 삭제가 용이하게 하기 위해 doubly linked list로 사용하였습니다. 입력이 완료되면 각 tweet을 노드로 하여 RBT를 구현합니다. 그러면 2번과 같은 것들이 좀 더 쉽게 탐색될 것입니다.

2. What is your expected performance

대체적으로 대부분의 기능들이 O(1), O(log n), O(n), O(nlog n)으로 구현하도록 노력했습니다. 프로그램의 시작시 user.txt와 friend.txt, word.txt read하는데 O(n), tweet으로 만든 Hash\_table에서 RBT를 만드는데 O(n)의 기능들이 수행됩니다. 이후 0은 O(1), 1은 O(n), 2는 평균 O(log n) 최악은 O(n), 3은 정렬로 인해 O(nlog n)의 퍼포먼스를 보입니다. 4는 tweet을 찾는데 O(log n) best case는 O(1)이 걸리고 그 후, user를 찾는데 O(n)의 퍼포먼스를 보입니다. 5는 찾은 user 당 O(n) (여기서 n은 전체 유저 수를 뜻합니다.)의 퍼포먼스를 보입니다. 6, 7은 아직 구현이 덜 되어 완벽하게 performance를 묘사하기는 어렵습니다. 죄송합니다.. 8, 9는 일반적인 Dijkstra+Priority Queue, DFS 알고리즘들과 같은 퍼포먼스를 보입니다.

3. How would you improve the system in the future

여러 개의 자료구조를 쓸려고 노력하다 보니 코드가 이리저리 난잡해보입니다. Python에 대한 이해 부족으로 C언어처럼 자유롭게 주소값을 참조하지 못해 변수와 배열 등이 좀 더 많아져 Memory를 좀 더 잡아먹는 듯합니다. 그리고 Python의 언어 특성상 2번째 test case같이 많은 양의 input이 들어오면 파일을 read하는데 오랜 시간이 걸리는데 이걸 O(n)보다 줄일 방법이 없어 너무 안타깝습니다. 다음 번에는 hash 함수를 향상시켜보고 싶고, uid만 담는 배열을 만들어 uid에서 각 트윗, name, friend로 자유롭게 탐색될 수 있도록 주소값을 활용해 코딩하면 좀 더 성능이 향상되지 않을까 싶습니다. 그리고 특히 7번이 너무 어렵고, 기말고사가 끝나고 남은 시간동안 코딩에 실패하였는데 이걸 꼭 방금 제시한 방법을 통해 해결해보고 싶습니다.