저장 자료구조로는 해쉬 테이블을 사용했고, 이유는 값에 대한 접근 속도가 O(1)로 가장 빠르기 때문이다. 자세한 구조와 작동 방법은 매뉴얼에 소개되어있다. 따라서 유저 프로필 정보나 친구정보, 단어와 유저 사이의 관계를 나타날 때 해쉬 테이블 또는 해쉬 테이블을 중첩시킨 구조를 사용하였다.

또한 정렬은 힙 소트를 이용했는데 이유는 최대값이나 최솟값을 모두 구하지 않아도 되고 한 개 또는 5개정도만 구하면 되는 경우가 많기 때문이다. 따라서 O(nlgn)의 시간복잡도로 정렬과 출력을 모두 할 수 있다. 또, 다익스트라 알고리즘의 경우에는 우선 순위 큐를 만들어야 하는데 여기에도 힙 정렬이 사용된다.(좀 더 시간이 있었으면 heap class를 여러 개 만들지 않고 하나의 heap class에 타입을 object로 받아서 정렬하는 것을 구현해보고 싶었는데 시간이 부족해서 구현하지 못했고, 코드가 지저분해졌다.)

0번은 데이터 파일을 읽고 통계를 낸다. 1번은 0번에서 낸 통계에 대한 결과를 말한다. 2,3번은 각각 제일 많이 트윗한 단어와 사람을 5명까지 출력한다. 4번은 특정 단어를 입력 받고 그 단어를 말한 사람을 모두 출력한다. 5번은 위에 찾은 유저들의 친구들을 모두 출력한다. 6번은 특정 단어를 입력받고 그 단어를 모두 지운다. 7번은 특정단어를 입력받고 그 단어를 말한 유저들을 모두 지운다. 이때, 다른 파일(userprofile, friendship)의 데이터에서도 모두 지운다.. 8번 scc를 찾는 문제 같은 경우에는 내가 직접 짠 그래프는 정상적으로 실행되는데 샘플의 경우는 실행이 되지 않고, 모든 vertex를 하나의 scc로 본다. 9번 같은 경우는 한 유저를 입력받고 그 유저부터 다른 유저들까지의 거리를 내림차순으로 출력하고, 그 중 top 5까지 가는 경로를 출력한다.

앞에서 말했듯이 여러 중복되는 기능을 가진 클래스 또는 메소드들을 정리하지 못했다. 그에 따라 코드들이 지저분해졌고, 어떤 기능을 구현하는 데에 일관성이 없다. 시간이 조금 더 주어졌다면 클래스와 메소드를 정리하고 일관성 있는 프로그램을 만들 수 있었을 것이다. 또 8번에 시간을 쏟았는데 결국 실패해서 9번까지도 만들지 못했는데 모두 수업시간에 배운 내용으로 작성할 수 있지 않았을까싶다. 또한 해쉬맵을 중복으로 만들다보니 크기가 큰 파일의 경우 dfs를 하는데 stackoverflow가 발생하기도 했다. 따라서 친구 목록의 adj list는 링크드 리스트로 바꾸는 게 좋을 것 같다.

하나 아쉬운 점이 있다면 내 힘으로 여러 자료구조를 만들려는 시도를 한 것이다. 해쉬테이블도 만들다가 그만두고, 힙 같은 경우에도 굉장히 복잡해졌다. 이 시도 자체는 실력 향상에 도움이 된 것 같지만 촉박한 시간에 과제를 해야 되는 입장에서 라이브러리에서 갔다 쓰고, 그 다음에 다시 만들어봐도 좋을 것 같았다. 마지막에 만든 다익스트라 같은 경우는 라이브러리를 갖다 써서 다른 코드보다 보기가 좋다.