Report for the project

2011120322 홍성민

# What data structure you chose and why.

우선 이 프로젝트는 rep파이썬에서 구현하였으며, 처음에 text 파일을 받아들이는 과정에서는 line 별로 슬라이싱을 한 후 각 사용자 별 정보를 node로 구성하였습니다. 그리고 각각의 node를 list에 저장하여, userprofilelist, userfriendship, tweets 의 이름으로 저장하였습니다.

하지만 text 파일의 정보에서는 위 프로젝트를 실시하는데 필요없는 id에 대한 정보나 날짜에 대한 정보가 있기 때문에, 좀 더 효율적인 자료구조 구현을 위해, 필요한 정보들을 뽑아 hash table을 만들었습니다. Hashing을 선택한 이유는 hashing이 시간복잡도가 가장 작고( O(1)), 제시된 문제해결을 위해 가장 효율적이라고 생각했기 때문입니다.

또한 list의 정렬은 파이썬 라이브러리 함수 대신, heapsort를 활용하여 n log n의 시간복잡도를 충족시켰습니다.

# What is your expected performance?

먼저 버튼 0번을 눌러 파일을 불러들이면, raw data에 대한 데이터 (total tweets, total friendship records, and total users)가 표시되고, 그 이후에는 메뉴의 버튼을 누르고 나오는 지시사항을 따라 진행하면 문제없이 프로그램이 작동합니다. 다만, 시간관계상 9번 shortest path의 경우 관련 코드만 만들어놓고 (Dijsktra) 구현하지 못했습니다. 또 SCC의 경우에는, 작동이 가능하기는 하지만, 코드가 깔끔하지 못하여 자료가 커질 경우 stack overflow가 발생합니다.

기본 요구사항 중 가장 복잡했던 프로세스가 어떤 단어를 언급한 유저를 지우는 것이었는데, 시간 복잡도를 줄이기 위해서 우선 그 단어 중심 해쉬테이블에 방문하여 그 value값을 받아놓고, 이에 저장되어 있는 유저 중심 해쉬테이블에 바로 방문하여 그 value값(words)를 제거하는 방법을 취했습니다. 이와 같은 방법은 유저와 그의 팔로잉이 기록되어 있는 해쉬테이블에서도 똑같이 수행하여 시간복잡도를 개선하였습니다.

# How would you improve the system in the future?

우선 완료하지 못한 extra-credit 요소들을 완료하고, 만일 이 프로젝트가 확장되어도 raw data의 쓰이지 않는 파일이 계속해서 쓰이지 않는다면, text파일을 받아들일때 바로 hashing의 형태로 저장하는 방법을 고안할 것입니다.

그리고 SCC의 오버플로우 발생을 통해, 해싱한 값을 다시 노드로 바꾸는 작업 대신, 처음에 받아들일때 바로 노드형태로 저장해서 직접 SCC를 구현하여, Recursion depth를 줄이는 방향으로 개선할 수 있을 거 같습니다.

지난 과제(transpose) 제출을 교수님께 직접 이메일로 보냈었습니다. 기한은 넘기지 않았고, 교수님께서 조교님께 전달하여 채점해 주신다고 했는데, 혹시 누락하시지 않으셨는지 확인부탁드립니다. 한 학기동안 감사합니다.