자료구조 트위터 구현

2015410020 문재호

1. 저는 BFS기반의 vertex를 이용하여 유저, 친구, 트윗을 구성하는 자료구조를 만들었습니다. 이러한 자료구조를 선택한 이유는 이미 만들었었던 지난 5월의 과제에 코드를 활용할 수 있기도 하고 SCC나 Dijkstra 알고리즘을 이용하기도 쉽기 때문이었습니다.
2. Readdata: O(n^3)…각각의 tweet을 생성하여 비교하다보니 시간복잡도가 증가하였습니다. 약 4분정도 걸리는 것 같습니다….  
   statistic: O(n) 이미 구해놓은 값을 비교하는 과정입니다.  
   mostW: O(n) 처음 트윗 5개 중에서 가장 작은 값을 가지는 번호로 모두 저장한 후에 비교합니다.  
   mostU: 위와 같습니다.  
   findU: O(n) 유저에 저장된 트윗을 조사하여 비교합니다  
   findRecent: O(n^2) 최근 검색한 유저의 친구를 찾아야 하므로 유저내의 친구목록을 비교해야 합니다.  
   deleteMW: O(n^2) 유저의 트윗 또한 같이 소멸해야 하므로 일일히 하위항목을 비교하여야 합니다.  
   deleteUMW: O(n^2) 위의 이유와 같지만 삭제해야할 것이 더 많습니다.  
   SCC: O(n logn) 지나간 길을 각각 표시하면서 반복구간을 찾아야 합니다.(미구현)  
   shortest: O(n logn) 다익스트라 알고리즘을 이용해 각 노드까지의 최단거리를 비교해야 합니다.(미구현)
3. 제가 만든 자료구조는 지난 과제에 사용하였던 BFS Vertex를 이용하여 관련 노드들을 더 추가하는 형식으로 진행하였는데 처음부터 그래프 형식으로 구조를 만들었기 때문에 하위 항목들이 많은 자료구조의 특성 상 새로운 노드가 생길 때 값의 크기에 따라 정렬하기 위해서는 하위 항목들도 또한 같이 옮겨져야 하는데 이것은 시간이 상당히 오래 걸리는 작업이어서 정렬하지 않는것과 같은 시간복잡도를 가지겠다고 생각하여 ID값에 따라 정렬하지는 않았습니다. C언어의 경우 구조체의 포인터를 서로 바꿔주는 것 만으로도 하위항목들이 같이 옮겨지는데 비해 Python에서는 제가 찾아본 바로서는 그러한 기능이 없는 것 같기에 초반 readdata의 시간을 단축시키지 못하였습니다. 하지만 이 같은 경우는 초반 입력값으로 들어오는 데이터의 양이 상당히 많아서 생긴 경우이기 때문에 실제적으로 한 번에 수천개의 데이터가 들어오는 경우는 잘 없으므로 그 자체로는 괜찮을 것이라고 생각합니다.  
   트윗을 검색하는 과정에서 문제가 생겼는데, 처음에 readline()을 통해서 txt값을 읽어왔을 때 엔터까지 같이 읽어오지만 비교하는 트윗에는 엔터가 없기 때문에 목표 트윗을 찾을 수가 없는 상태입니다.  
   Vertex의 구조를 조금 수정하고 연결리스트들을 이어놔야 할 것 같습니다.