**Report on Making Tweeter**

**Pycharm 5.0.4 on Mac X El Captain 10.11.5**

2015410118

컴퓨터학과

오영진

1. **Design**

* User와 Word에 대한 정보를 각각 다른 트리에 저장을 하기로 했다. 두개의 정보는 레드 블랙 트리에 저장 하기로 했다.
* User Red Black Tree(코드 내의 class이름: UserRBTree)

UserRBNode

UserRBNode에 저장되는 데이터

1. User number(id) = Red black tree 정렬 기준인 key값

2. User name

3. User’s friends

Linked list 사용하려고 했으나 데이터가 너무 복잡해져서 Python 내장 List사용.

4. tweet words

5. followers

6. RBNode상속

* UserRBtree에 저장되는 데이터

1. RB tree 상속
2. Number of users
3. User id의 크기를 기준으로 레드블랙트리에 삽입을 함.

* Word Red Black Tree(코드 내의 class이름: WordRBtree)
* WordRBNode 저장되는 데이터

WordRBNode

1. RBNode 상속
2. Word 저장
3. 이 단어를 트윗 한 유저들의 리스트: Python 리스트
4. 이 단어를 사용한 User 리스트에 추가 / 삭제

* WordRBTree 저장되는 데이터

RBTree 상속

1. **Interface에 있는 각각의 경우에 대한 작동법 고안**
2. **Read data files**
   1. Total users: when UserRBNode is added, +1 deleted, -1.
   2. Total friendship: add all the number of friends in the tree
   3. Total tweets: when WordRBNode is added, +1 else, -1
3. **display statistics**
   1. Average number of friends: Total friendship / Total users
   2. Minimum number of friends: tree.minimum().key
   3. Maximum number of friends: tree.maximum().key
      1. B and C have tree that is rearranged by number of friends as key value.
   4. Average tweets per user: Total tweets / Total users
   5. Minimum tweets per user: tree.minimum().key
   6. Maximum tweets per user: tree.maximum().key
      1. E and F have tree that is rearranged by the number of the tweets as the key value.

**2. Top 5 most tweeted words**

**:** Take all nodes that are in WordRBTree then rearrange the tree by the number of the users in the nodes as key. Print the top 5 nodes by recursion.

**3. Top 5 most tweeted users**

**:** Take all nodes that are in UserRBTree then rearrange the tree by the number of the words a user tweeted as key. Print the top 5 nodes by recursion

**4. Find users who tweeted a word (e.g., ’연세대’)**

**:** Find the WordRBNode in WordRBTree that has same word as the input then print the list that is in the WordRBNode. The WordRBNode is saved in a variable.

**5. Find all people who are friends of the above users**

**:** The tree that is rearranged in 3 is used. Find the users in the UserRBTree and find the friends of them to print.

The node in 4 is saved so we can easily find the users who tweeted the word above then find the users in the UserRBTree. For all UserRBNode that have same name with the user, print all the friends that are in nodes.

**6. Delete all mentions of a word**

: Find the WordRBNode in WordRBTree that has same word as the input word. After that, for all users in the WordRBNode, search the users in UserRBTree. For UserRBNode that have been searched, delete the input word in the node.

**7. Delete all users who mentioned a word**

**:** Find the word in WordRBTree that matches with input word then find the UserRBNode in UserRBTree that matches with the user lists in word. Find the UserRBNode-2 who are friends of the users. Delete the users in the UserRBNode-2 of friends in the list of followers.

**8. Find strongly connected components**

**:** import sys

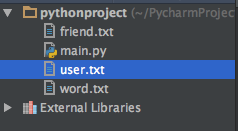
sys.setrecursionlimit(1500)

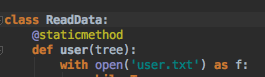
Since there were too many recursion, I used this code to lower the limit.

**9. Find shortest path**

**:** used Dijkstra to find shortest path from graph

**10.** **How the data is read in the code?**

The texts are in also in python project so that it is easier to access the texts so the data were accessed by the statement with open(‘user.txt’) as f.

**스크린샷%202016-06-25%20오후%201.55.36.png**

The word.txt in Pycharm cannot have all the data input in word.txt in larger data sample so I just opened the file directly from the text file.

1. **Report**
   * 1. **What data structure I chose and why**

: I choose red black tree because it is well balanced tree compared to normal binary search tree because the worst run time of normal binary search tree is O(n) when searching the key value while red black tree is O(logn). Because insertion and deletion is used frequently, the use of faster data structure was necessary. Also, I used the lists that is already in Python. I wanted to use linked list or another red black tree to save the information of friends and words but it was too complicated and too many bugs occurred.

* + 1. **What is my expected performance?**

I hoped it would work in same time as the red black tree does. Searching a specific user takes O(logn) and insert takes O(logn). However, when deleting the node in the UserRBTree, I also have to delete it in WordRBtree so it takes more time than usual red black tree. There are many trees that are made when running the program, which takes O(nlogn) time for each tree but I think it would take more time because there is a lot of information to store.

* + 1. **How would I improve the system in the future?**

1. The function - top\_five\_words() – is suing global variable which is not really nice and nit. I try to used count variable in the function but it did not work properly so I just choose to use global variable.
2. Trees that print out the most tweeted users and most tweeted words had to rearranged all over again every time any data in the UserRBTree or WordRBTree is deleted. I wanted to make this tree rearrange all nodes only once and whenever certain data is deleted, the nodes that their data had changed can be rearranged, not all of them.
3. I saved the input words in red black tree, but it is much better to use hash table to save words. It is much faster than red black tree. If I used hash table, I could have made less trees. I think there are too many trees I made.
4. The code for the “Delete all users who mentioned a word” is got too much complicated after fixing the bugs. I should have made functions to do the work instead of writing down in main().
5. There exist some bugs while running strongly connected component and the code in 8 and 9 is too complicate and I want to make it much simpler later.
6. **Self-Evaluation Form**

• Submit a github account (10) 10

• Commit source code displaying menu (10) 10

• Commit the first draft of manual (10) 10

• Read data files (20) 20

• Statistics (20) 20

• Top 5 most tweeted words (10) 8

• Top 5 most tweeted users (5) 4

• Find all users who mentioned a word (10) 8

• Find all users who are friend of the above user (5) 4

• Top 5 strongly connected components (10) 5

• Find shortest path from a user (id) (10) 8