Adam Ryder

CS4050

Reference Web Project

Cover Letter

Total hours spent: 15

* Data Structures
  + I used an adjacency list to represent the graph. I found it was easier to implement than a 2d array.
  + The adjacency list was made with a Dictionary with the key being the person’s name and value being a list of Nodes.
    - Each Node is essentially an object that contains the reference’s name and the reference weight.
  + I chose a dictionary because it is essentially a hashmap, so looking up a person is O(1)
* Design decisions
  + I chose to use C# because I’m tired of using java.
  + I chose to have a GUI because it is quite simple to implement with Visual Studio/C# and also because it would be easier to display data to the user.
* Big O
  + Function:
    - Add a person
      * To add a key to a dictionary is O(1) every time.
    - Add a reference to a person
      * This adds a Node object to the list of Node objects that are associated with a key.
      * So to get they list of objects it is O(1) because it is just getting the values at the key. To insert the node object into the list you have to make sure it’s not already there so at best is O(1) and at worst O(n).
    - Remove a person and all references to/from.
      * To remove a person and all references to them is O(1) because you’re just removing a key.
      * To remove all references that person made, you have to traverse the list of Nodes for every other key and remove that node from the list if it exists.
        + This at best is a O(n) if every list is empty, and worst O(n^2) if every list has N nodes.
    - Remove a reference to a person
      * To do this, you grab the list of Nodes for that person which is O(1) and traverse that list till you see that person’s Node and remove that Node. This is at best O(1), if it’s the first reference, and O(n) if it’s the last reference.
    - Change reference to person1, from person2
      * O(1) to grab the list of Nodes for person1, and then traverse the list of nodes is best O(1) and worst O(n).
    - Display the web
      * This is accomplished by traversing every list of nodes associated with a person which is at best O(n), if there are no references, and O(n^2) if every list of nodes has n references.
    - Report on one person
      * This displays all references to that person and all references from that person.
      * To get the references to that person, you grab the list of nodes for that persons name, O(1), then traverse those nodes which is at worst O(n), then traverse all other lists of nodes to see if he exists in there. This can be at worst O(n^2).
    - Written about
      * All references written about a single person.
      * Grab the list of nodes associated with that person is O(1), and traversing that list is at worst O(n) and best O(1), if there is no list.
    - Written by
      * Traverse all lists of nodes for every key, which is at best O(n), if every list is empty. At worst it will be O(n^2) if every list has n nodes in it.
    - All highly recommended
      * Traverse every list of every key, checking the weight of reference. If there exists a ‘3’ then stop and add that key to the list.
      * This is at best O(n) if the first node in the list is 3. If there are no ‘3’s in the list then this is O(n^2).
    - Statistics
      * This shows the total number of people in the graph, and total number of references by weight.
    - Top X candidates
      * This traverses every list of every key, summing up the weights for each key and adding that key plus it’s total weight to a dictionary of <String, Int> . The string is the persons name and the int is their total score. This is at worst O(n^2).
      * Then get the max of the values in the dictionary, which is at worst O(n).
      * For X times, traverse the dictionary of scores looking for the max. This is O(n), if you find the max, add that person and their score to a list to be passed to the GUI. Do this X times.
      * So total, this is O(n^2) + O(n) + O(xn) which is O(n^2).
    - Save as report
      * This saves the graph to a specified file, in a format that is readable by humans.
      * This is just a traversal of the dictionary
    - Save web
    - This saves the graph to a specified file that can be used later as input to this program.
    - Accepted input format
      * Add a person
        + Add person joe
      * Add a reference to a person
        + Add reference joe sam highly recommend
        + Add reference joe sam recommend
        + Add reference joe sam not recommended
      * Change a reference to a person
        + Change reference joe sam highly recommended
        + Change reference joe sam recommend
        + Change reference joe sam not recommended
      * Delete a person
        + Delete person sam
      * Delete a reference
        + Delete reference sam joe