Final Exam

Due: 20 January 2021 11:59 PM

Goal: Learn graph and sorting algorithms

In this Exam, you will work with two types of data, FinalExamGraphFile.txt and Positive.txt

FinalExamGraphFile.txt contains 3 columns and 32836 rows.

First and Second columns represents the ids of people and 3rd column shows the strength of relationship amount these people. By using FinalExamGraphFile.txt you need to create a weighted graph, where ids will be node ids and strength of the relationship will be edge weights.

1. (10 point) For this purpose, you need to write 3 classes Node.java, Edge.java and Graph.java.

By using above classes, you will need to create and Adjacency List of the graph.

Inside your main class, prompt the user display a given node’s adjacent nodes by asking a node id given **a certain range**. Such as, as a user if I enter 1518, your code should display adjacent nodes to 1518 as:

(1518,1690) 1

(1518,2222) 1

(1518,2478) 1

Here 1 represents the strength.

1. (10 point) Next, load Positive.txt file which represents people ids whom they become friends in the future while they were not friend in the graph file. Here are 5592 friendship relation.
2. (20 point) Now, create a pair of nodes set with 5592 friendships between 2 pairs of node, similar to Positive.txt file entries, that are not friend in the graph (they do not share an edge in the graph file but the negative pairs must be different than positive pairs). Example, 2251 16 is a positive pair while 2251 1518 is a negative pair and you must create 5592 such negative pairs.
3. (25 point) Now, take positive and negative pairs, look at Adjacency List to see each how many common friend each pair has. For example, (I am just making up the number here) 2251 16 pairs has 5 common friend while 2251 1518 pair has 2 common friend.
4. (35 point) Then, you create such common friend scores for all positive and negative pairs (this could be an array with length 5592 + 5592). Call this array score. Implement quicksort, heapsort, merge sort and insertion sort for this array and report time that takes for each sorting.

Find the midpoint of sorted array, label above pairs as positive and below pairs as negative (Here you have to be very careful not lose information about pair during sorting) from the midpoint. And, compare how accurate you guess as positive and negative comparing actual positive and negative pairs (see the hint below).

Hint: If you have positive pairs 1 2, 3 4, 5 6 with common friends as 2,5,7 respectively and negative pairs 7 8,9 10, 11 12 with common friends as 3,1,4, respectively.

You sort these friends as 1 2 3 4 5 7 and guess 11 12, 3 4, 5 6 as positive pairs and rest as negative pairs. Thus, your guess is correct with 4/6 accuracy.