COMP 272
Take-Home Finals
Due in Sakai Saturday, 18th December, 2021 by 5 pm
Also, code files to be shared with Marius Ebert via private repository

- The exam or its solution is not to be discussed with any human being by any means electronic or otherwise. No posting to Chegg or other websites.
- I reserve the right to interview anyone after the finals to ensure they have done the work on their own.
- Failure to follow honor code or any late submission will result in a finals score of 0.
- Email submission to me will not be accepted. It will result in a finals score of 0. So please pay attention to the due date and time.
- Please do not submit the data file. Only that are explicitly listed in the exam.

You are given two data files -- List of Airports worldwide with 3 letter codes based on IATA. (These are real airport codes). List of direct routes in the format of (a1,a2), where a1, and a2 are in the list of airports. The routes are computer generated. Number of airports = 9135. Number of routes= 24,992. You have two goals:

1. Write a method
   public int minRoutes(String airport) {

   }
   that returns the minimum number of direct routes to be added starting from the parameter *airport* so that every airport can be reached from the *airport*. Find the answer for minRoutes("ORD"), and you know that "ORD" is Chicago O'Hare. You should get the answer 546.
2. Assume each direct route is bidirectional, meaning that if route (p,q) exists then (q,p) also exists, for each route in the data file. (By the way this assumption is false in the datafile *routes.txt* , but ignore it for the time being. Even as it is false in the dataset, you can expect this to be true in reality, and hence this line of posing a problem is justified). Find how many flight networks (each is connected within itself but to not other networks) exist in the *routes* database. Write a method
   public int numFlightNetworks(…) {
   }
   returns the number of such flight networks.

Methodology

3. First and foremost, this problem is to be modeled as a *directed graph G* where each airport is the (label of a) vertex, and the routes are the directed edges.
4. *minRoutes(..)* can be solved by a reduction of strongly connected components of *G* to a DAG and then finding the number of *sources* (those with indegree 0 in the reduced graph).
5. *numFlightNetworks(…)* can be solved by first by obtaining the *undirected* underlying graph *H* of G and then finding the connected components of H. (These will be the weakly connected components of the original graph G). The trick is to not actually do any transformation of G to H. But simply combine the incoming and outgoing edges of each vertex of G into a larger list and run the regular depth-first traversal (DFT) on it. You don't have to do this upfront but do it as you go about DFT. By the way, this could be done without necessarily solving *minRoutes()*. You should get 590 weakly connected components of G.

Finer points

6. You can do the finals in one of two ways.
   (i) Either you use a bunch of HashMap based data structures (and other data structures) and treat the vertex labels as Strings (for the airports), and the edges as ordered-pairs of Strings (p,q) (for the direct routes). OR
   (ii) You map airport codes into integers 0, 1,2, through 9134 and map the routes as a pair of integers in the same range. If you map this way, you can use the solution to Homework 10 and solve most of it rather quickly.
7. Since you are given the list of airports in the file, you know the size of the vertex set in advance. First create all the vertices and then add edges by reading from the routes file. You have worked with some datasets in which you did not know the size of the vertex set in advance but that is not the case here and that should make it a bit easier.
8. Between each depth first traversal (post-order or regular), **don't forget to initialize** the *marked* data structure. This initialization gets even trickier with approach (i) because you will need a HashMap<String, Boolean> (or in my case HashMap<E, Boolean>).
9. I didn't try approach (ii) but I did (i). Overall, while the ideas are straightforward with approach (i), the coding was rather tedious, mostly because I used Java generics, namely I developed the entire solution based on generic type E and then instantiated it with String type. If you used Strings straight instead of generic type E, you definitely would have an easier time than I did. Even with that simplification, I am of the opinion that approach (ii) would be easier for you to try but don't quote me on that because as I said, I did not implement (ii). I think it would be easier because you have worked with Integer labels for graphs more than String labels and hence easier to debug if you run into any difficulty. With approach (ii), you set up proper data structures to go back and forth between integer labels and airport codes and vice versa and transform String based routes to integer-based routes. Because minRoutes would need for the parameter to be a String.
10. Some intermediate results might be useful for you to check. The maximum sized SCC in G is 7725. Total number of SCCs = 1411. The number of edges in the reduced graph is 1784.
11. I will post some code files from Homework 10. Also, you must use the given class AirportRoutes for your submission and complete the methods provided therein.
12. I decided against providing short answer questions as part of the finals – because there is a bit of work to do here.
13. Best wishes.