CO322: Data Structure and Algorithms

Lab 04: Graph Applications – 24 hour Challenge!

Aim: Solve a problem by reducing it to a Graph representation and developing/applying efficient algorithms.

Problem Statement: An overseas friend of yours is planning to visit Sri Lanka on a sightseeing tour. (S)he asks for your help in planning the tour, reminding you that (s)he's traveling on a student's budget. S(he) wants to visit Kandy, NuwaraEliya, Galle, Jaffana, Trincomalee, Anuradhapura, Arugambay, Yala and Pinnawala; (s)he doesn't care about the other cities s(he) visits on the way. The list of cities and the cost of travel for feasible routes between these cities are given in the "data.txt" file (please note that cities that your friend is not interested in are also in the list; these cities may help you to find a better route). Your objective is to find the minimum cost tour that will satisfy the following conditions:

1. The tour must start & finish in Colombo.
2. You planned tour should cover ALL the places that your friend wants to visit.
3. You should try to maximize the routes you cover. To quantify this you'll use the following cost calculation method: If you take a route (between two cities) that you have already taken, you'll regard the money spent of that leg of the tour as a waste. Therefore, you'll add the cost of that route as a penalty to the total cost (e.g., if the cost between cities D and E is 100, and you take route DE or ED more than once, you'll add 100+100(penalty) to your cost calculation for each consequent travel on that route).

Challenge: Develop an *efficient* algorithm to find the "*minimum cost tour*" for the above problem. Your program should output the following:

- Your proposed best route (i.e., the list of routes (edges) travelled, in order of travel), the cost associated with each leg, and the total cost of the tour.
- Submit your source code, a sample output of your program and a short (1/2 page) description of the strategy you used to solve the problem, on FEeLS before the deadline(s) given below.

You have two deadlines:

NOTE: In both cases, your source code should compile & run in Instructor's machine.

1. **Win prizes: Submit by 02:00 pm tomorrow (06/05/2016)**
    a. Prizes (chocolates?) and boasting rights for the best tour (minimum cost), the most efficient algorithm (minimum time taken – in Instructor's run) and the "most efficient algorithm for the best tour" under the following guidelines:
        i. The best tour will get a prize if the time taken to find the route doesn't take 50% more time than the most efficient algorithm.
        ii. The most efficient algorithm will get a prize if the cost of the tour is no more than 50% higher than the minimum cost tour found.
        iii. The tour costs and the running times will be given percentage marks w.r.t. the minimum cost tour and the most efficient algorithm, respectively. The solution with the minimum combined marks will get the "most efficient algorithm for the best tour" prize.
        iv. In case of ties, code clarity and better coding practices will be used as the tie-breaker.
        v. Plagiarism will result in disqualification and ZERO marks.
        vi. The judges (Instructors and the LiC) word/decision will be final.
    b. All working and original solutions will get full marks.

2. **Normal Submission: 05:00 pm on Friday May 12th, 2015**
    a. Working solutions will get marks based on tour cost and efficiency.

Help:

You are given the following:

- Basic Adjacency List implementation of Graph Data Structure in Java (Graph.java).
    - Please note that this is a basic implementation of the Graph Data Structure using Adjacency Lists. Feel free to add/remove/modify this to suit your needs.
- The main file (GraphApp1.java) with sample usage of the Graph implementation to create a Graph DS from the given data file.
- The data file (data.txt) in the following format:
    - Line one contains all Cities (i.e., vertices) separated by Spaces
    - Following lines contain routes (i.e., edges), one route per line, with cost
    - Sample File Content:

```
A B C
A B 54
B C 39
C A 4
```