Programming Assignment 2

Programming assignments are to be done individually. You may discuss the problem and general concepts with other students, but there should be no sharing of code. **You may not submit code other than that which you write yourself or is provided with the assignment. This restriction specifically prohibits downloading code from the Internet.** If any code you submit is in violation of this policy, you will receive no credit for the entire assignment.

This lab is due **Friday, April 3rd at 11:59PM.** If you are unable to complete the lab by this time you may submit the lab late until Monday, April 6th at 11:59PM for a 20 point penalty.

The goals of this lab are:

- Recognize algorithms you have learned in class in new contexts.
- Ensure that you understand certain aspects of graphs and greedy algorithms.

Problem Description

You are a vacation planner for a small travel agency, and your most recent clients are a newly-engaged couple, Rachel and Nick, who would like to plan their honeymoon to Europe. They have a list of cities they want to visit, but since Nick recently got disowned by his disapproving and incredibly rich Singaporean family, they are on a budget and will be taking the train between locations to cut costs. Because their budget isn't enough to visit all the locations on their list, they have chosen two cities that they agree are a must-see, and have asked you to find the cheapest way to get from one city to the other. Fortunately, you recall the time way back in college when you took an Algorithms class and assure them that their problem can be solved with the help of your extensive algorithmic knowledge.

Before you start creating Rachel and Nick's travel plans, you get sudden news from the European Railway Agency that due to efforts to cut costs, the government must close down some of the railways between cities. However, they must ensure that citizens are still able to get from any city to any other city (regardless of how long it takes). Each railway has a certain cost of maintenance, and they need to figure out the tracks to close down so the entire network has the least possible total maintenance cost. The government is asking for your help, since in addition to being a part-time travel agent, you are also a world-renowned network engineer.

In this project, you will implement a minimum heap by the `minCost` variable in City. You will also implement an algorithm to find the lowest-cost path between two given cities. Finally, you will implement an algorithm to find the minimum total cost to visit all cities given a list of cities and train ticket prices using the same list of cities and prices. Assume that trains between any two cities are priced equally regardless of direction (i.e. an undirected graph). We will provide you with the following classes:

- `City`
  Keeps track of `int minCost` (the least money needed to get from this city to the starting city), and `int cityName` (the numerical id of the city on the train schedule).

- `Heap`
  Needs to have all the properties of a heap. This class needs to function independently; the methods we ask you to implement will not all be needed for your algorithm, but we will test them separately.
- `Program2`
  The class that you will be implementing your algorithms in.
- `Driver`
  Reads file input and populates ArrayLists `cities`. You can modify `testRun()` for testing purposes, but we will use our own `Driver` to test your code.

You need to implement the `Program2` class. `Driver.testRun()` can be modified for your own testing purposes, but we will run our own `Driver` and test cases.

Part 1: Write a report [20 points]

Write a short report that includes the following information:

(a) Give the pseudocode for your implementation of an algorithm to find the lowest-cost path between two given cities. Give the runtime of your algorithm and justify.

(b) Give the pseudocode for your implementation of an algorithm to find the minimum total maintenance cost when shutting down railways with the condition that all cities are still connected. Give the runtime of your algorithm and justify.

(c) From a space-complexity and time-complexity perspective, does it make more sense to use an adjacency list or adjacency matrix for your implementations? Explain your answer.

Part 2: Implement a Heap [20 points]

Complete `Heap` by implementing the following methods:

- `void buildHeap(ArrayList<City> cities)`
  Given an ArrayList of Cities, build a minimum heap in $O(n)$ time based on each City's `minCost`.
- `void insertNode(City in)`
  Insert a City in $O(\log(n))$ time.
- `City findMin()`
  Find minimum in $O(1)$ time.
- `City extractMin()`
  Extract minimum in $O(\log(n))$ time.
- `void delete(int index)`
  Delete the City at a given index in $O(\log(n))$ time.
- `void changeKey(City c, int newCost)`
  Updates a City and rebalances the Heap in $O(\log(n))$ time. (HINT: Try to be more efficient than scanning the entire heap every time you want to update a City.)

Break any ties by `int cityName`.
For example if City 0 and City 1 both have `minCost = 4`, choose City 0 to be "smaller".

Part 3: Finding the Lowest-Cost Path [30 points]

Implement an algorithm to find the lowest-cost path between two given cities. You will be heavily penalized for a non-polynomial time (in terms of the number of cities) algorithm. The specific inputs provided and outputs expected are provided below.

Input(s):

- A starting `City start`
- A destination `City dest`

Output:

- A `cost` value where `cost` represents the minimum cost required to get from City `start` to City `dest`.

Method Signature:

- `int findCheapestPathPrice(City start, City dest);`

Part 4: Finding the Lowest Total Cost [30 points]

Implement an algorithm to find the minimum total maintenance cost to ensure that all cities are still connected given a list of cities and railway maintenance costs. You will be heavily penalized for a non-polynomial time (in terms of the total number of cities) algorithm. The specific inputs provided and outputs expected are provided below.

Input(s): none
Output:

- A `totalCost` value where `totalCost` represents the sum of all of the costs of maintenance for the remaining railways (i.e. the sum of all edge weights in a minimum spanning tree for the given graph).

Method Signature:

- `int findLowestTotalCost();`

Of the files we have provided, `City`, `Heap`, and `Program2` are what will be used in grading. Feel free to add any methods or fields but be careful not to remove any to ensure that your classes remain compatible with our grading. Also, feel free to add any additional Java files (of your own authorship) as you see fit.

Input File Format

The first line of file is the number of total cities, and the number of total connections between the cities. The first number on each even line is the name of a city, and every number that follows it

is a city that it has a connection to by train. The first number on each odd line is the name of a city, and every number that follows it is the cost of the ticket between it and the connected city in the line above.

For example, if the input file is as follows:
3 2
0 1
0 9
1 0 2
1 9 8
2 1
2 8
Then there are 3 cities, 2 connections.
City 0 has a connection to City 1 with ticket price 9.
City 1 has a connection to City 0 with ticket price 9 and City 2 with ticket price 8.
City 2 has a connection to City 1 with ticket price 8.

We will parse the input files for you, so you don't need to worry too much about the input file format except when trying to make your own testcases.

Instructions

- Download and import the code into your favorite development environment. We will be grading in Java 1.8 on the ECE LRC machines. Therefore, we recommend you use Java 1.8 and NOT other versions of Java, as we can not guarantee that other versions of Java will be compatible with our grading scripts. **It is YOUR responsibility to ensure that your solution compiles with Java 1.8 on the ECE LRC machines.** If you have doubts, email a TA or post your question on Piazza.

- Make sure you don't add the files into a package (keep them in the default package). Some IDEs will make a new package for you automatically. If your IDE does this, make sure that you remove the package statements from your source files.

- If you do not know how to download Java or are having trouble choosing and running an IDE, email a TA, post your question on Piazza or visit the TAs during Office Hours.

- There are several `.java` files, but you only need to make modifications to `Program2` and `Heap`. You may add additional source files in your solution if you so desire. There is a lot of starter code; carefully study the code provided for you, and ensure that you understand it before starting to code your solution. The set of provided files should compile and run successfully before you modify them.

- `Driver.java` is the main driver program. Modify `testRun()` to suit your liking. A main portion of the lab is debugging—be sure to leave time for that.

- Make sure your program compiles on the LRC machines before you submit it.

- We will be checking programming style. A penalty of up to 10 points will be given for poor programming practices (e.g. do not name your variables foo1, foo2, int1, and int2).
- Before you submit, be sure to turn your report into a PDF and name your PDF file `eid_lastname_firstname.pdf`.

What To Submit

You should submit a zip file titled `eid_lastname_firstname.zip` that contains all of your java files AND a **separate** pdf report `eid_lastname_firstname.pdf`. Do not put your java files in a folder before you zip them (i.e. the files should be in the root of the ZIP archive), and **do not include your pdf in the zip file**. Your solution must be submitted via Canvas by 11:59pm on the due date.