EE360C Algorithms - Summer 2021
University of Texas at Austin — Programming Assignment #2
Prof. Nur Touba — Due: July 25, 2021, 11:59pm

Programming Assignment #2

Programming assignments are to be done individually. Do not make your code publicly available (such as a Github repo) as this enables others to cheat and you will be held responsible. 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 programming assignment is due **Sunday, July 25, 2021 at 11:59pm.** If you are unable to complete the assignment by this time, you may submit the assignment late until Tuesday, July 27, 2021 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

Upon the failure of the Texas power grid, you realize that you haven't submitted your assignment yet. In order to make it to the deadline, you decide to take fate into your own hands and find an alternative source to restore the electricity in your house. You know that UT has its own power plant, so you plan to connect your house to UT via cheapest possible way. After some online searching, you find a list of m existing wires throughout the city (with fixed endpoints) that are up for sale at different prices. In the first part of this project, you have to come up with an efficient algorithm to determine the cheapest set of wires you need to purchase to make a connection between UT and your house.

After your hectic experience with the previous assignment, you lose trust in the power grid and decide to come up with a backup plan for all of your future assignments. You realize that your classmates have the same concern, and hence you may be able to share the wires you purchase in order to split the cost. For the second part of this project, you have to design an efficient algorithm to find the cheapest set of wires such that there is a connection between UT and every single one of your classmates. Here we are assuming that the wires have sufficiently large capacity, meaning that once you purchase a wire and pay its price, any number of students can use that wire on their path to UT. In addition, we are trying to minimize the total price of purchased wires, without getting into individual payments.

In this project, you will implement a minimum heap by the `minCost` variable that will facilitate implementing your efficient algorithms for the two problems described above. In addition to the heap, you will be implementing two algorithms: 1) an algorithm to find the lowest cost to connect yourself to UT, and 2) an algorithm that connects you and all your classmates to UT's power plant. For simplicity, we assume **all the wires have endpoints at either UT or a student's house.** We will provide you with the following classes:

- `Student`
  Keeps track of `int minCost` (the minimum cost needed to connect this student to UT) and `int name` (the numerical id of the student). Also contains `ArrayList<Student> neighbors` (list of students that are connected to this student via a single wire) and `ArrayList<Integer> prices` (price of wires between this student and his/her neighbors).

- `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 `ArrayList students`. You can modify `testRun()` for testing purposes, but we will use our own `Driver` to test your code.

You need to implement the `Heap` and `Program2` classes. `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 minimum cost to connect yourself to UT. Give the runtime of your algorithm and justify.

(b) Give the pseudocode for your implementation of an algorithm to find set of wires to be purchased in order to connect you and all your classmates to UT using the minimum total price. Give the runtime of your algorithm and justify.

For both (a) and (b), make sure to give enough detail in your pseudocode so that the runtime can be accurately determined. (For example, when following a wire to update the student's `minCost`, how do you "find" the corresponding `Student` in the heap?)

Part 2: Implement a Heap [20 points]

Complete `Heap` by implementing the following methods:

- `void buildHeap(ArrayList<Student> students)`
  Given an ArrayList of students, build a minimum heap based on each student's `minCost` in either $O(n \log(n))$ or $O(n)$ time.

- `void insertNode(Student in)`
  Insert `Student in` in $O(\log(n))$ time.

- `Student findMin()`
  Return minimum in $O(1)$ time.

- `Student extractMin()`
  Return minimum and delete from heap in $O(\log(n))$ time.
- `void delete(int index)`
  Delete the Student at `int index` in $O(\log(n))$ time.
- `void changeKey(Student s, int newCost)`
  Change `minCost` of `Student s` to `newCost` and update the heap in $O(\log(n))$ time.

Break any ties by student name.
For example if Student 0 and Student 1 both have `minCost = 4`, choose Student 0 to be "smaller".

Part 3: Connect Yourself to UT [30 points]

Implement an algorithm to find the minimum cost to connect yourself to UT. You will be heavily penalized for a non-polynomial time (in terms of the number of students and wires) algorithm. The specific inputs provided and outputs expected are provided below.

Input(s):

- You, given by `Student start`
- UT, given by `Student dest` (this means that we are treating UT as a student with a unique id number)

Output:

- A `cost` value where `cost` represents the minimum total price required to connect Student `start` to Student `dest` (i.e., UT)

Method Signature:

- `int findMinimumStudentCost(Student start, Student dest);`

Part 4: Connect Entire Class to UT [30 points]

Implement an algorithm to find the minimum total price needed to connect all students to UT, given a list of students and the price of wires between them. You will be heavily penalized for a non-polynomial time algorithm. The specific inputs provided and outputs expected are provided below.

Input(s): none (notice that here you don't really need to know UT's id)
Output:

- A `cost` value where `cost` represents the minimum total price required to connect all students together

Method Signature:

- `int findMinimumClassCost();`

Of the files we have provided, `Student`, `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. **The files you turn in MUST compile and run with the provided, unmodified `Driver` to ensure it runs with the `Driver` used in grading.**

Input File Format

The first line of file is the total number of students (one of them represents UT) and the total number of wires between them. The first number on each even line is the id of a student, and every number that follows it is a student that is connected to it by a wire. The first number on each odd line is the id of a student, and every number that follows it is the price of the wire between the student and the adjacent student 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 students, 2 wires.
Student 0 has a wire to Student 1 at price 9.
Student 1 has a wire to Student 0 at price 9 and Student 2 at price 8.
Student 2 has a wire to Student 1 at 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 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 and our grader.** If you have doubts, email a TA or post your question on Piazza.
- **Do not add any package statements to your code.** 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 before turning in the assignment.
- There are several `.java` files, but you only need to make modifications to `Heap` and `Program2`. 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. It currently prints out your Graph and Heap. Modify `testRun()` to suit your liking. A main portion of the lab is debugging—be sure to leave time for that.
- 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).

NOTE: To avoid receiving a 0 for the coded portion of this assignment, you MUST ensure that your code correctly compiles with the original, unmodified `Driver.java` on Java 1.8. Do not modify the signatures of or remove existing methods in `Student.java`, `Heap.java`, and `Program2.java`. Do not add package statements. Do not add extra imports. You must zip your code using the exact format described below with no spaces. We recommend testing compilation of your code using the ECE LRC Linux machines (using "javac *.java" and "java Driver inputfile.txt") after redownloading `Driver.java` from Canvas. We will not be allowing regrades on this assignment, so please be careful and double check that your final submission is correct.

What To Submit (please read carefully)

You should submit to Canvas a single ZIP file titled `pa2_eid_lastname_firstname.zip` that contains all of your java files (`Student`, `Heap`, and `Program2`) and any extra `.java` files you added. 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). Your ZIP file name MUST have the exact format: `pa2_eid_lastname_firstname.zip`. Be certain that there are no spaces in your zip file name. Failure to follow this naming format will result in a penalty of up to 5 points.

Your PDF report should be typed or legibly scanned and submitted to Gradescope. Both your zipped code and PDF report must be submitted BY 11:59pm on Sunday, July 25, 2021. If you are unable to complete the assignment by this time, you may submit the assignment late until Tuesday, July 27, 2021 at 11:59pm for a 20 point penalty.