I&C SCI 46 Lecture Fall 2021 Project 5: My Favorite Trees

Due **December 3 at 9:59 AM**

Introduction

I think minimum spanning trees and priority queues are neat. This isn't why they're taught in ICS 46, though: they both have a variety of applications.

Getting Started

Before you begin work on this project, there are a couple of chores you'll need to complete on your ICS 46 VM to get it set up to proceed.

Refreshing your ICS 46 VM environment

Even if you previously downloaded your ICS 46 VM, you will probably need to refresh its environment before proceeding with this project. Log into your VM and issue the command ics46 version to see what version of the ICS 46 environment you currently have stored on your VM. Note, in particular, the timestamp; if you see a version with a timestamp older than the one listed below, you'll want to refresh your environment by running the command ics46 refresh to download the latest one before you proceed with this project.

If you're unable to get outgoing network access to work on the ICS 46 VM — something that afflicts a handful of students each quarter — then the ics46 refresh command won't work, but an alternative approach is to download the latest environment from the link below, then to upload the file on to your ICS 46 VM using SCP. (See the Project #0 write-up for more details on using SCP.) Once the file is on your VM, you can run the command **ics46 refresh_local NAME_OF_ENVIRONMENT_FILE**, replacing **NAME_OF_ENVIRONMENT_FILE** with the name of the file you uploaded; note that you'd need to be in the same directory where the file is when you run the command.

Creating your project directory on your ICS 46 VM

A project template has been created specifically for this project, containing a similar structure to the basic template you saw in Project #0.

Decide on a name for your project directory, then issue the command **ics46 start YOUR_CHOSEN_PROJECT_NAME project5** to create your new project directory using the project5 template. (For example, if you wanted to call your project directory proj5, you would issue the command `ics46 start proj5 project5` to create it.) Now you're ready to proceed!

Choosing a project partner

You *have the **option*** to work with a second person for this assignment. If you do so, I expect you to work via pair programming. That is, you may **not** split the assignment, such as by having one person implement the priority queue and the other person write code for the MST.

Similarly, any academic dishonesty arising from a group will be treated as an offense by both partners.

To declare a partnership, **both partners** need to fill out the following form by **November 29** at 9:59AM. *There will be no exceptions granted. Be sure you fill it out correctly and that you know what your UCINetID is and how it is not your UCI ID number. Be sure your partner has submitted it.*

https://docs.google.com/forms/d/e/1FAIpQLSePJnI_moaAv5BwpzEpAOBLAuCo7EZHLiAqyaq1dZ9M_HdjCw/viewform

Requirements

- Fill in MyPriorityQueue.hpp
    - Your implementation must fit the interface given.
    - Your implementation must be templated as provided.
    - Your implementation must be a binary min heap. If you don't know why I used the modifier "binary" here, don't worry. It's the min heap from class.
    - You do not need to write a copy constructor or an assignment operator, but knowing how to do so is generally a good thing.
    - All functions that need to be implemented have been started in the provided .hpp
    - For comparing keys, use the "natural" comparison offered by <.
      Any test cases provided will have something for the key that has this defined.
    - If you want to define your own < for use in the second part, the Edge.hpp class will show you how. If it isn't clear, please ask!
    - Do not hard code for assumptions about the tree that are used in the second part of the program.

- Write the function to compute the MST in `proj5.cpp`
    - You must use your PriorityQueue from the first part as appropriate.
    - Your implementation must be based on Prim's algorithm, as shown in class.
    - You may assume the graph is *complete*: for all pairs of vertices *i,j* with *i != j*, there is a positive edge weight.
    - You may assume the graph is *simple*: there are no parallel edges, and no self-loops. The "edge" weight from i to i is set as 0 for all test cases.

    - Your implementation does not have to be the most efficient thing ever, but it cannot be "too slow." In general, any test case that takes over two minutes on the grader's computer may be deemed a wrong answer, even if it will later return a correct one.

You are explicitly allowed to use `std::vector` in your solution. In addition, you may use parts of the C++ standard library that solve small/trivial parts of the assignment (ex. `std::max`, `std::swap`) that you could implement yourself if you needed to. You may NOT use parts of the standard library that would do most of the work of the assignment for you (ex. `std::priority_queue`). If you are unsure if something is considered a trivial part of the assignment, please ask.

Deliverables

After using the gather script in your project directory to gather up your C++ source and header files into a single **project5.tar.gz** file (as you did in previous assignments), submit that file (and only that file) to Checkmate. Refer back to Project #0 if you need instructions on how to do that.

You will submit your project via Checkmate. Keep in mind that you're responsible for submitting the version of the project that you want graded. We won't regrade a project simply because you submitted the wrong version accidentally. (It's not a bad idea to look at the contents of your tarball before submitting it; see Project #0 for instructions on how to do that.)

Can I submit after the deadline?
Yes, it is possible, subject to the late work policy for this course, which is described in the section titled Late work in the course reference and syllabus.

Grading
Your grade for this project will be graded on correctness: I will run some number of test cases using Google test. Each is worth some number of points and is graded based on whether or not your code correctly determines if the puzzle has a solution, and if so, what it is. If it is determined that your program does not make an attempt to solve the problem at hand, you will not get these points, regardless of the result from testing. The tests will look a lot like the tests in your Google Test starting directory for this assignment; if you pass those, you're off to a good start, but it's not a guarantee.

The same caveats that applied to previous assignments apply here as well. Make sure you test your project extensively, including at least one test for every function in your priority queue. For best results, write Google Tests instead of relying on main.