Alex Childers

PA #2 Reflection Essay

02/25/2019

I rate my work on this assignment a 6.5/10. I’m pretty ecstatic that I got the program to work at a Tier 2 level, but there are also some aspects of my code that I wish I had cleaned up, and bonus features/tiers that I wish I had implemented.

There were two particular hurdles that I struggled with during this assignment: translating alphabetical node names into building abbreviations and names (e.g. P2 represents BSS, or Behavioral and Social Sciences), and the Tier 2 tasks of indicating which buildings one is likely to pass while going from one building to another.

I think there were two main reasons that I struggled with translating alphabetical node names into building names: comprehending the programmatic difference between full node names and individual building names (e.g. Canyon Complex is a node, but Alder and Chinquapin are buildings in the Canyon Complex), and trying to save space when loading this information into the program. I spent a while diagramming out the structure of each building’s nomenclature. I was able to finally get over this hurdle by accepting that I was going to have to use memory to store node/building names effectively, as well as drawing from what I learned in my CS 325 (Database Design) class: careful data duplication can be a good thing, and may eliminate errors having to do with foreign keys, consistency, and such. This particular aspect of the assignment was an interesting exercise in the idea that being more time-efficient often leads to being less space-efficient. If I had programmed a way to load and save shortest paths to and from a file, as I wanted to, this too would have been an exercise in saving time at the cost of memory.

Even in last semester’s PA having to do with graphs, I struggled with building a path from one node to another. I looked to my code from last semester for guidance on how to record a path between nodes, but I don’t think my code was very good last semester, so I spent a while trying to implement path-recording by adding new member functions to the GraphNode class. I had considered using C++ tuples to make the process of recording a path in CampusGraph::computeShortestPath() easier, but I was hesitant to use a C++ feature I had never used before. However, in the end, I wasn’t able to record a path correctly within GraphNode objects, so I converted all the pairs in computeShortestPath() to tuples and learned the syntax as I went. In the end, it wasn’t as difficult as I thought it would be, and now I’ve learned a new C++ tool.

There are some aspects of the assignment that could be improved. While I understand that our class needed data to be collected before starting the assignment, I wish that I had more notice that I needed to walk around campus and collect data so that I could plan accordingly. I also think that this assignment could produce more accurate data if there were a way to standardize the weights among various nodes—I found that the time it took me to walk from building to building was affected by how tired I got as I walked the entire campus. I also noticed that some data in the given distances.csv looked odd—I find it hard to believe that one could walk from T2 (Schatz Energy Research Center) to Z (Science A) in 30 seconds when it also takes 38 seconds to walk from U (Telonicher House) to Y (Gist Hall). If it were possible to measure distance and/or topography between nodes, that might make for a more informative campus graph.

(By the way, I recorded the path I walked around campus, and I thought it looked interesting.)

