Eduard Klimenko

343 HW5

Report

To begin the problem, I first read the chapter 9.2 about Kruskal’s Algorithm. I took very detailed notes because I knew I would have to apply it in HW5 part A and even more importantly I knew I would have to implement it in part B. After taking notes and understanding the chapter is started applying the algorithm to the problem instance in part A. I verified the solution I obtained by using brute force as the problem instance was fairly small.

I was not sure how to start part B so I read the entire pdf again. I coded small parts of the program which I knew I could do like getting input from the keyboard. I then we into creating the graphs based on the input from the keyboard which was simple.

Starting the algorithm, I knew I had to use a heap as it was specified in the assignment. Remembering back from my data structures course I recalled that it can be implemented as a priority queue. This is much faster and efficient over implementing a heap from scratch including all of its operations. Unfortunately, I had a compiler error which stated that the Edge class did not have a valid compareTo() method. A simple work around I to pass in a comparator class as an argument along with the edges. The compare method simply accessed the data fields as doubles and returned -1, 0, or 1.

From there it was pretty straight forward. The code was very close to the pseudo code so there was no problem converting it to Java. I had previous code from trees as arrays so I used that to my advantage when making the find and union functions.

Printing out the different pieces of information such as the data and the two vertices were a hassle. I had to cast everything to doubles or it wouldn’t compile and I also had to review string format to have a nice-looking output.

Thinking of test cases was difficult for me because it was hard to imagine what the graph would look like on paper. Graph1 was similar to a linked list as it should just add on as it goes because the weights are increasing. Graph2 was for testing random cases like having the same edge between two vertices but having different values. This should grab the smaller edge and not add the same edge again as it is not part of the same disjoint set (which it did). It also tested several other things. Graph3 was built from part A and is was what I used as input for most of the time because I knew what the solution was supposed to look like.

Overall, the assignment was pretty difficult because you needed a firm understanding of the graph, heaps, find, union, sets, input, and other concepts. Understanding Kruskal’s algorithm was essential in completing this assignment.