Julian Duodu

My Priority Queue was implemented by making an ArrayList of PQNodes that were parameterized to be available for almost any data type. The PQNode class held a value, a key, and a Handle. Each Handle contained an integer index that pointed to the spot in the priority queue where the Handle’s associated data pair was located. All exchanges of information in my class were done through a swap method that switched the PQNode locations and updated the Handles to point towards the new indices. The decreaseKey method was implemented by checking the value of the new key against the value of the existing key. As long as the handle that was passed through the parameter into the method was contained in the priority queue, then the method would push the key and its corresponding node up through the heap until the min-Heap property was satisfied. MinHeap was implemented by checking the key of a parent index against its children in order to find the smallest child. From there, if the smallest child was smaller than the parent, then the two of them would be switched and the child would be bubbled up in the heap until the min-heap property is satisfied. The insert method functioned by increasing the length of the ArrayList by 1, adding the element to the priority queue, and then re-establishing the min-heap property through utilization of the swap method. The ExtractMin method was implemented by recording the value of the priority queue’s minimum value, then switching it with the min-heap’s last value before reestablishing min-heap properties.

I had some trouble implementing Djikstra’s method. My code was modeled after the provided code in the handout and from lecture, albeit with a few modifications to accommodate the handles. However I still get a null-pointer error and was never able to finish the returnPath method as a result. I believe that if I were able to resolve the nullPointer error, then the code would be able to function. However, the error seems to stem form the way that I am trying to utilize an array of distances and handles to maintain the priority queue without needlessly re-inserting elements. The ShortestPaths Constructor functions almost exactly like Djikstra’s. It starts by populating a distance array with values of infinity at all indices except 0, where the startId is stored. From there, the method is supposed to extract the minimum value from the priority queue, iterate though all of the edges adjacent to the extracted Vertex, and then navigate to the vertices that have not yet been visited and have a key that can be decreased. This method continues to cycle, filling up and extracting from the priority queue, until an array of edges in between all of the “shortestPath” vertices has been populated. This array of edges is then passed on to returnPath, where it is supposed to be used to navigate the shortestPath in reverse by iterating through the array of edges in reverse. The IDs of these edges are stored in another array that is then returned through the method to provide a path to the program. However, I never managed to get that far on the code, so that is mostly theory. I do apologize for not managing to finish this.