Steven Friedman
steven.friedman 431474
Lab 4 (paths)

PRIORITY QUEUE

My priority queue class passes all tests. The priority queue is an ArrayList of Nodes, a new object I created. The constructor, isEmpty, min, handleGetKey, handleGetValue, and toString are all pretty straightforward. Insert works similarly to the class priority queue pseudo code, except it handles increasing the size of the queue with a special case within the while loop and outside the loop in case the loop doesn't run. ExtractMin works very similarly to the class pseudo code, calling heapify. I had some issues with indices here, but I worked them out. Heapify finds the smaller child of the input by assuming the first one and creating a special case for the second one. Heapify also calls a swap method. Swap isn't too complicated, but I kept trying to set handles to a new handle object with the new index and realized that I had to get the handle and set the index. DecreaseKey changes the key if it should be changed and moves it up the queue.

HANDLE

Handle just has an index and a getter and setter for it.

NODE

These are the elements of the priority queue. It has a key, a value (of type T), and a handle.

SHORTEST PATHS

Shortest paths implements Dijkstra's algorithm using a priority queue of vertices and an array of objects called VertexInfo, which stores information about the vertices. The constructor is pretty similar to the class pseudo code for Dijkstra, but it keeps track of the edge of the path that brought you to each given vertex. ReturnPath returns an empty array if there is no path. Otherwise, it counts up the number of edges in the path to construct the array and then constructs it and fills the array with those edges. I tried using an ArrayList and then using the toArray method, but I was having trouble with types, since an ArrayList can only be of Integers, not ints, but then the toArray would make an array of objects, not ints.

VERTEX INFO
This keeps track of the information about the vertices in the priority queue. It has a handle, a parent vertex, and a parent edge ID number.