My approach to this problem was to first familiarize myself with the graph ADT code and understand how it works. I also read and understood the Kruskal’s algorithm and how it works. Once I knew the basics, I started writing code for the algorithm. I created a hash table that maps labels to vertices and used this to store the graph data. I implemented the heap data structure to store the edges in a priority queue. I then wrote code for the Kruskal’s algorithm and tested it on my own data sets. To ensure that the code was working correctly, I also tested it on various other data sets. After I was satisfied that the code was working as expected, I wrote a program to generate the minimum spanning tree and output the total cost. Finally, I wrote a program to read in the graph data from a file and output the set of edges in the minimum spanning tree.

When I was assigned this task, I was intimidated by the sheer amount of information that I had to process. I was unfamiliar with the graph ADT, Kruskal's algorithm, and heap implementation. I had to take the time to understand how each of these elements worked and how to combine them to form a solution to the problem.

The first step I took was to read and understand the code that was provided to me. I read through the code samples and tried to understand how they worked. I also took the time to read up on the graph ADT, Kruskal's algorithm, and heap implementation. I read through the descriptions and examples of each to understand how they worked and how they could be used to solve the problem.

Once I had a good understanding of the problem, I began to write code. I started by implementing Kruskal's algorithm, which would be used to find the minimum spanning tree. I then wrote code to read in the graph data and store it in a hash table. After that, I wrote code to calculate the total cost of the minimum spanning tree.