Alex Blair and Aleisha Smith

Algorithms

14 December 2018

Final Project Analysis

Aleisha and I chose to implement both Prim’s and Kruskal’s Algorithms for minimum spanning trees on an adjacency matrix for a graph of five nodes as well as ten. Initially we split up the work half and half with Alex implementing Prim’s algorithm and Aleisha implementing Kruskal’s algorithm. Alex chose to use C++ for his implementation and Aleisha chose Python. We communicated effectively and met multiple times to discuss implementation techniques and challenges together. The main problem we faced was with implementing union find in the context of disjoint sets for Kruskal’s algorithm and we were still unable to completely implement this algorithm due to the complexity and our lack of understanding of how to utilize union find successfully. However, we did successfully implement Prim’s algorithm which can take a matrix of a given size and it will output the edges in the minimum spanning tree as well as the weights associated with those edges. We mainly utilized greedy algorithm techniques for both implementations because this technique fits naturally with both Prim’s and Kruskal’s algorithms. In the original problem description, the data set required was to be a one thousand node graph which after further discussion and attempted implementation we did not find or create a data set this large. Doing so would have required about a 32X32 matrix. Therefore, we settled with two smaller examples, one which is five nodes and one which is ten. Each of these matrices is run through our algorithms and the findings are reported below.

We ran each of the algorithms side by side, and they behaved similarly. Based on our research, Prim’s algorithm and Kruskal’s algorithm both run at O (n log n), although our implementation of Kruskal’s algorithm runs a little bit slower than that since we had to leave out the union-find method. We compared the results from a 5x5 matrix and a 10x10 matrix. We are unsure about the accuracy of the results from Kruskal’s algorithm because we had issues with the implementation. Also, while we may not have reached exactly O (n log n) efficiency it seems to be decently close. We were also unable to test our code on a sizable matrix to see any drastic differences. Overall if we were to do this project over again, we may pour more resources into implementing Kruskal’s algorithm and understanding the union-find method.

While Aleisha and I both understand each algorithm and the concepts behind them, implementing them proved to be a whole different challenge. Many of the concepts that we knew did not directly translate or figuring out how to translate them eluded us. However, we did learn a lot from this whole process such as how each algorithm works on an adjacency matrix, and how with Kruskal’s algorithm each set is disjoint until finally coming together at the end to form a tree. We also learned how to properly represent a graph in code and how to read this input and perform some sort of operation on it. This project cemented the theory learned in both Advanced Algorithm Design and Analysis as well as Graph Theory to help us better understand how graphical representations in code can be important in real world applications.