

Programming Assignment 4

ANLY550

Norman Hong, Yingjie(Chelsea) Wang

Contents

[Part I – Results 1](#_Toc6075180)

[Part 2 – Discussion 4](#_Toc6075181)

# Part I – Results

The following are plots that contain the average weight of the minimum spanning tree. The average weight was calculated with 5 random trials.

The following table contains the averaged weight of the MST for graphs with different number of vertices. Furthermore, different graph dimensions were also tested.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Dimensions  Vertices | 0 | 2 | 3 | 4 |
| 16 | 0.940416823 | 2.680091 | 4.76975 | 6.194497 |
| 32 | 1.108691889 | 3.945498 | 7.15807 | 10.03887 |
| 64 | 1.280300712 | 5.590526 | 11.43279 | 17.50037 |
| 128 | 1.296532699 | 7.562562 | 17.55878 | 28.48571 |
| 256 | 1.166660605 | 10.5945 | 27.52708 | 47.18461 |
| 512 | 1.247803301 | 15.07284 | 43.2394 | 78.15761 |
| 1024 | 1.194949514 | 21.03212 | 68.27522 | 129.8046 |
| 2048 | 1.211521988 | 29.79775 | 107.3335 | 216.4143 |
| 4096 | 1.202861927 | 41.81634 | 169.1149 | 362.053 |
| 8192 | 1.199598397 | 58.85173 | 267.3135 | 603.8539 |
| 16384 | 1.204204615 | 83.10667 | 421.9562 | 1009.899 |

# Part 2 – Discussion

Since the graphs are dense, Prim’s algorithm was chosen over Kruskal’s algorithm to compute an MST. The implementation utilized an adjacency matrix to initialize the graph. For a low n, the adjacency matrix works fine. However, when n got large, I noticed a lot of memory was being used to store all possible edges. The space complexity of an adjacency matrix is O(|V|2). It would be more efficient to utilize an adjacency list because the space complexity is O(|E| + |V|).

A list object was used to implement Prim’s algorithm. Therefore, the runtime is O(|V|2). A list object was used because of its simplicity. A binary heap implementation of Prim’s algorithm has time complexity of O(|E| + |V|log|V|). For a dense graph, the number of edges dominate over the number of vertices squared. Therefore, this means that using a binary heap implementation is slower than the list implementation of Prim’s algorithm.

Another possible method to optimize the creation of an MST is to create a sparse graph by pruning edges that are unlikely to be contained in the MST. The following figure shows that the graph with a dimension of 4 has MST that contains the highest edge weights when compared to dimension 3, 2 and 0. This implies that it is possible to simplify the graph by removing edges that are greater than the maximum edge weight found in an MST for a dimension 4 graph. The following figure shows the maximum MST edge weight with respect to the number of vertices in a graph. The equation y = 1.1611x-0.223 can be used to determine which edges to remove from the graph. Since a sparse graph is created, it would be best to implement Kruskal’s algorithm instead of Prim’s algorithm.

For dimension of 0, the average MST weight is roughly 1.18. The slope is essentially 0, which implies a horizontal line. This means that the average MST weight is constant. For dimension of 2, the equation was determined to be y = 0.7043x0.4911. For dimension of 3, the equation was determined to be y = 0.7629x0.6493, and the equation was determined to be y = 0.8039x0.7346 for dimension of 4. It would seem that the exponent is ½ for dimension of 2, 2/3 for dimension of 3, and ¾ for dimension of 4. However, it is possible that this is a coincidence.

The following figure shows the runtime for graphs of different dimensions.

The figure clearly shows that the runtime increases exponentially as n increases for graphs of the same dimension. The runtimes are the same for graphs of different dimensions. The reason it’s the same is because the same Prim’s algorithm is being ran. There would be differences in runtime if we had measured the runtime of initializing each graph as well.