**CSCI E-124 - Minimum Spanning Trees in Random Graphs**

**By: Tofik Mussa**

I was tasked with collecting data points to determine a formula for the average weight of a minimum spanning tree for a graph with randomly generated weighted edges in dimensions 0 and 2 through 4.

I used java for the implementation. I initially drew a complete graph and because of having to store the edges in memory, I quickly ran out of heap space. The maximum graph I could generate within a reasonable amount of time was only 2^13 vertices. I tried to come up with a regression line to predict the maximum weight for a given number of vertices in the two-dimensional case. I incremented the number of vertices by 100 to get a better approximation for finding out about the maximum weight. I run 20 trials for each number of vertices and computed the average for the trials.

I got this best fitting line running the experiment once for 27 data points for the zero-dimensional case starting from 2 up to 2702 incremented by 100.

Shape

Description automatically generated

I got this best fitting line running the experiment once again for 27 data points for the zero-dimensional case starting from 2 up to 2702 incremented by 100.

Shape

Description automatically generated

I got this best fitting line running the experiment once for 26 data points for the two-dimensional case starting from 2 up to 2602 incremented by 100.

Chart, scatter chart

Description automatically generated

I got this best fitting line running the experiment once again for 26 data points for the two dimensional case starting from 2 up to 2602 incremented by 100.

Chart, scatter chart

Description automatically generated

I got this best fitting line running the experiment once for 23 data points for the three-dimensional case starting from 2 up to 2302 incremented by 100.

Chart, scatter chart

Description automatically generated

I got this best fitting line running the experiment once again for 23 data points for the three-dimensional case starting from 2 up to 2302 incremented by 100.

Chart, scatter chart

Description automatically generated

I got this best fitting line running the experiment once for 23 data points for the four-dimensional case starting from 2 up to 2302 incremented by 100.

Chart, scatter chart

Description automatically generated

I got this best fitting line running the experiment once again for 23 data points for the four-dimensional case starting from 2 up to 2302 incremented by 100.

Chart, scatter chart

Description automatically generated

I then compared if throwing edges have any effect. I compared the data points I had before throwing out the edges and after to check if there is any significant difference between the weight of the spanning trees.

Seeding

Discuss runtime of Kruskal

How to run it

I created modular classes for object orientation.

I had to force the garbage collector manually for running out of heap space. It did slow down the runtime. Simply used arrays instead of hash sets to save heap space. It also made the algorithm run faster. I had to also increase the heap size in my IDE.

It turned out to be the degree of the best fit for the polynomial regression is related to the dimension.

I achieved better results to estimate max weight when running twenty thousand trials for smaller number of vertices than I had while running smaller number of trials for large number of vertices.

I ran into issues using hash sets in Java, so I switched to Arrays

The max weight numbers for d3 and d4 are pretty close

I had to sort through multiple bugs that were offsetting my numbers

Generating the graph took so much time

I initially tried increments of 5 to determine max weight but later realized that the rate of growth as a function of number of vertices can’t be predicted like that. A smaller slope will be profound when the number of vertices grows

My heuristic was to take the log of the variable vertices that I obtained using polynomial fitting so that it grows proportionally as we double the number of vertices. The rate of growth for the smaller will effectively translate when doubling the vertices (it will grow by the same function of change) as I double

Why I chose Kruskal

I was surprised to find out that a graph with more number of edges can have less weight. This has to do more with the pseudo random generator in Java. It does have an uptrend when sampled enough times though. There will also be more edges with small weight that are up for grabs

I came up with piece wise function for throwing out edges

I had to have number of vertices in the top because I want it to grow and I had to trim it down by dividing

From too many observations, no matter what function I used to throw out edges, the total weight was very similar