CS 124
Austin Williams, Chris Pardee
Programming Asmt 1

Running Prim's Algorithm on Undirected Complete Graphs

For this project, we implemented Prim's Algorithm in Java and ran it on complete undirected graphs with the number of vertices mapped to increasing powers of two. We ran five trials for each number of vertices n and averaged the total weight of the five generated MSTs. We repeated this for increasing dimensions: 0, 2, 3, and 4 dimensions. The largest n we were able to run was 524288 vertices, which took over 20 hours to run five trials in each of the four dimensions.

Without any dimensions, our implementation of Prim's Algorithm generates $|V - 1|$ random edge weights per vertex on the range $[0, 1)$ using a uniformly distributed random number generator from java.util.random. We then thought that the average minimum weight edge that our algorithm would find would be near the value $\frac{1}{|V-1|}$. Our intuition stems from the fact that on each iteration, our algorithm chooses the minimum weight edge to a vertex not in *S* from a vertex in *S* from the set of edges previously generated that don't connect to a vertex in *S*. If the algorithm built an MST with all edge weights near $\frac{1}{|V-1|}$, the average weight of our MSTs would be around 1, regardless of the number of edges since a MST has $|V - 1|$ edges. However, we calculated the average weight of our MSTs with no dimensions to be very near 1.2, which led us to consider that even though we expect there to be $|V - 1|$ edges with minimum edge weights near $\frac{1}{|V-1|}$, our random number generator could generate a set of edges connected to an arbitrary vertex where the minimum weight edge is greater than or less than $\frac{1}{|V-1|}$. Why, on average, the minimum weight edge from the set of edges connecting to an arbitrary vertex is greater than $\frac{1}{|V-1|}$ is surprising to us.

The trends for two, three and four dimensions can be explained by the curse of dimensionality. The curse of dimensionality is a phenomenon that occurs when the number of dimensions increases. As the number of dimensions used to describe points increases, the volume of the space the points exist in also increases. This means that random points in this space will be further away from each other than in lower dimensions. Essentially, there's a lot more room for the points to exist in.

We see this effect start to happen in two dimensions. The average weight of the MST is no longer locked in around 1.2, but it actually increases with the number of vertices. We see that the amount of change between numbers of vertices increases a lot as we increase the number of dimensions. For example, at 524288 vertices in two dimensions the average weight of the MST is 469.023614, in three dimensions it's 4214.49072, and in four dimensions it's 13340.5318. The average weight of the MST at 524288 vertices increases by roughly a factor of 10 between dimensions 2 and 3, and by a factor of 3 between dimensions 3 and 4.

We find that the rate of increase with the number of vertices appears to follow a square root function. The growth rate is faster than a log function, but slower than a quadratic or exponential function. We have estimated the function $f(n)$ for two dimensions to be $f_2(n) = \frac{\sqrt{n}}{1.55}$, for three dimensions it is $f_3(n) = (\sqrt{n})^{1.26}$, and for four dimensions it is

$f_4(n) = (\sqrt{n})^{1.44}$. As the number of dimensions increases, we increase the exponent that the square root function is being raised to. It never gets too close to $f(n) = n$, since the maximum exponent is 1.44.

Our conclusions from analyzing this data are that we expect in the case with no dimensions that the minimum edge weight will be around $\frac{1}{|V-1|}$, and that in higher dimensions, the curse of dimensionality explains the increasing distances between points, which creates higher weight MSTs. We also found that the growth rate in each of the dimensions as the number of vertices increases follows a square root function. We were surprised that the average weight of the MSTs in the case with no dimensions was consistently close to 1.2 as opposed to 1.0. Our intuition would lead us to think that the average weight of the MSTs in this case should be 1.0, so we do not know how to account for the extra 0.2 weight.

# Dimensions	Guess for $f(n)$
0	$f_1(n) = 1.2$
2	$f_2(n) = \frac{\sqrt{n}}{1.55}$
3	$f_3(n) = (\sqrt{n})^{1.26}$
4	$f_4(n) = (\sqrt{n})^{1.44}$

# Dimensions	# Trials	# Vertices	Average weight
0	5	128	1.2031878
0	5	256	1.2701207
0	5	512	1.1974711
0	5	1024	1.2298102
0	5	2048	1.2121997
0	5	4096	1.1895994
0	5	8192	1.2032703
0	5	16384	1.200593
0	5	32768	1.1964904
0	5	65536	1.200593
0	5	131072	1.1970222
0	5	262144	1.1949738
0	5	524288	1.18656434

# Dimensions	# Trials	# Vertices	Average weight
2	5	128	7.4846277
2	5	256	10.689974
2	5	512	14.95201
2	5	1024	20.930115
2	5	2048	29.669987
2	5	4096	41.745735
2	5	8192	59.07405
2	5	16384	83.189514
2	5	32768	117.47069
2	5	65536	165.99219
2	5	131072	234.72235
2	5	262144	331.74515
2	5	524288	469.023614

# Dimensions	# Trials	# Vertices	Average weight
3	5	128	17.606438
3	5	256	27.542
3	5	512	42.990654
3	5	1024	68.37515
3	5	2048	107.417076
3	5	4096	169.59967
3	5	8192	266.90756
3	5	16384	422.23734
3	5	32768	668.16003
3	5	65536	1058.7845
3	5	131072	1677.1249
3	5	262144	2658.806
3	5	524288	4214.49072

# Dimensions	# Trials	# Vertices	Average weight
4	5	128	28.008259
4	5	256	47.17446
4	5	512	78.631226
4	5	1024	131.3377
4	5	2048	216.04749
4	5	4096	360.88583
4	5	8192	603.51685
4	5	16384	1008.5826
4	5	32768	1687.4877
4	5	65536	2827.9792
4	5	131072	4740.8613
4	5	262144	7947.922
4	5	524288	13340.5318