CS 124 Programming Assignment 1: Spring 2022

Your name(s) (up to two): Jackson Quick/Sean Roades

Collaborators: (You shouldn't have any collaborators but the up-to-two of you, but tell us if you did.)

No. of late days used on previous psets: Jackson Quick – 1, Sean Roades – 3
No. of late days used after including this pset: Jackson Quick – 2, Sean Roades – 4

Homework is due Wednesday at 11:59pm ET. You are allowed up to **twelve** (college)/**forty** (extension school) late days through the semester, but the number of late days you take on each assignment must be a nonnegative integer at most **two** (college)/**four** (extension school).

Results:

Table 1 gives the results for 5 iterations (except for the 3 smallest amounts of nodes, which were run on 50 for some better convergence to the mean) run on each of the dimension values $1, 2, 3, 4$ and the n values specified in the assignment handout. Dimension 1 here refers to the randomly generated weights for the uniform $[0, 1]$ distribution – that is, these are not points generated on the unit line. See Figure 1 for a visual representation of the table data. This data is also available in a CSV file in the submitted code.

Dimension	128(50)	256	512	1024(5)	2048	4096
1	1.1951	1.2054	1.2086	1.2157	1.2290	1.2066
2	7.58	10.64	14.99	21.13	29.50	41.89
3	17.60	27.50	43.18	68.20	107.75	169.33
4	28.53	47.09	78.19	129.65	215.58	360.60
Dimension	8192	16384	32768	65536	131072	262144
1	1.1994	1.1984	1.2020	1.2045	1.2036	1.2048
2	58.84	83.19	117.43	166.17	234.48	331.65
3	267.05	422.81	668.36	1057.93	1677.09	2658.80
4	602.84	1008.46	1687.84	2828.42	4740.50	7952.02

Table 1: Results for Various Dimension/Node Combinations

From the graph and data, it is clear that the family of each of the higher dimensions (other than 1, which is just constant) follows some sort of fractional polynomial function. Specifically, estimated values for each dimension:

- $f_1(n) = 1.2$
- $f_2(n) = n^{1/2.1496}$
- $f_3(n) = n^{1/1.5822}$
- $f_4(n) = n^{1/1.3892}$

Methodology/Discussion:

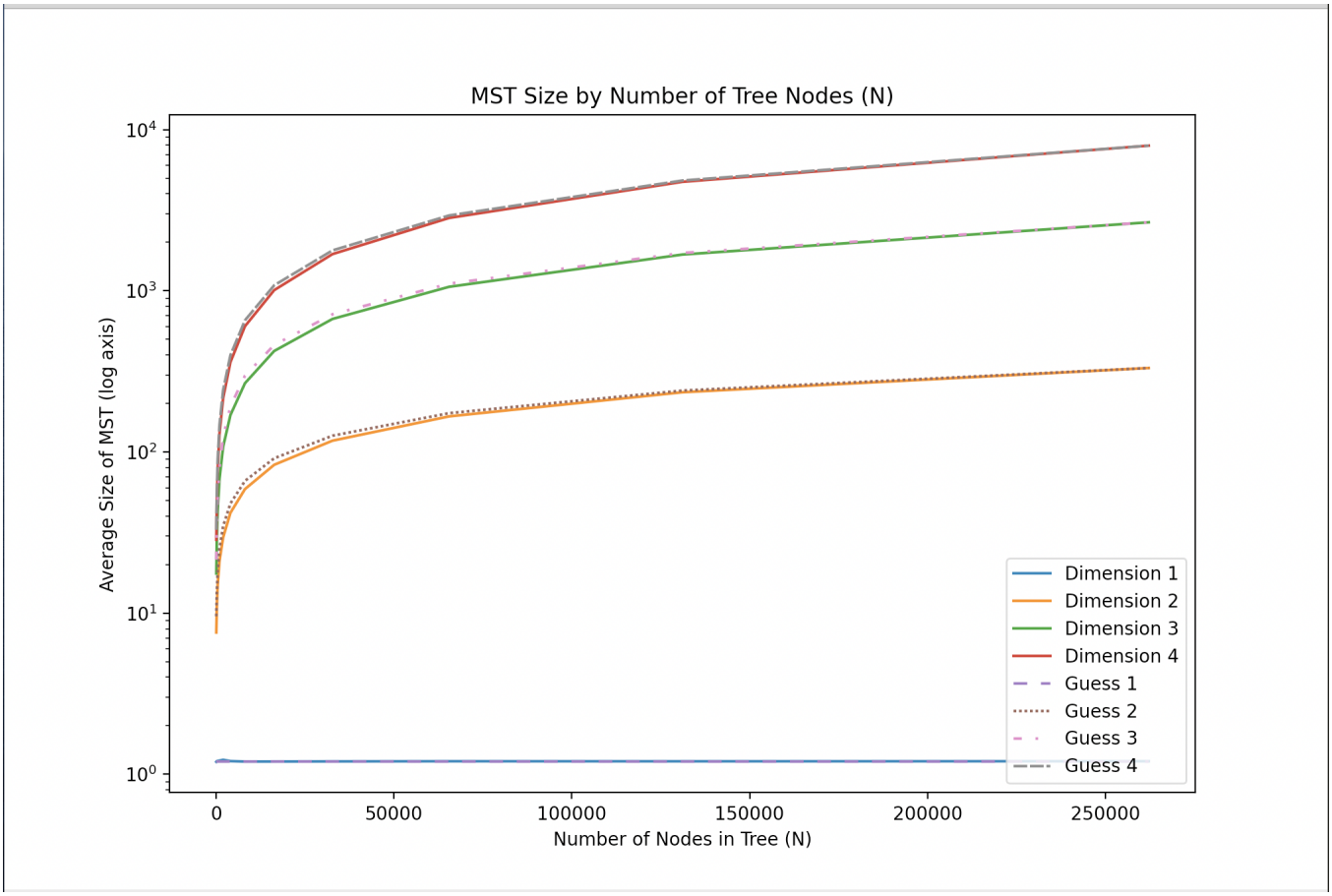

Figure 1: Graphical representation of the table data. Fractional polynomial relationship is clearly visible

We used Prim's algorithm, since the running time improves to $O(E + V \lg V)$ (an improvement over Kruskal) when using a Fibonacci heap, which we implemented. In total, the total runtime for calculating all values (reproducible using the included bash script) for the table took approximately 4 to 5 hours on OsX Big Sur on a 2.6GHz Intel i7 and 16 GB (only running one process). Surprisingly, random number generation/graph initialization took the most amount of time, particularly for larger values: Prim only took about 3 minutes to run in the largest worse case, while generating the weights in these cases took sometimes up to 9-10 minutes. As for the reliability of the randomness, using C's `rand()` would obviously not be an optimal choice for solutions where true randomness is demanded, but for our case here, pseudo-randomness (and a rather poor pseudo-randomness at that) will suffice.

We also made sure to implement cutoff values for each of the graph generations, both to improve speed and satisfy the memory constraints that plague the creation of a full graph at 2^{18}, i.e., over 34B edges. The precise cutoff values are given in table 2. Note that a check to ensure that no generation yielded an unsolvable MST Prim (i.e., at least one edge reached all vertices). This was done by asserting that Prim produced trees with $V - 1$ edges. None of the generations for our results with the given cutoff values produced such a case, although this of course is possible given enough iterations.

We were somewhat surprised by the constant f_1. An explanation for this might be the need to select

Dimension	128	256	512	1024	2048	4096
1	0.5	0.5	0.5	0.025	0.025	0.005
2	0.75	0.75	0.75	0.1	0.1	0.05
3	0.75	0.75	0.75	0.25	0.2	0.15
4	1	1	1	0.4	0.3	0.3
Dimension	8192	16384	32768	65536	131072	262144
1	0.005	0.0025	0.001	0.0005	0.0005	0.0005
2	0.03	0.03	0.02	0.02	0.01	0.01
3	0.15	0.1	0.1	0.075	0.075	0.05
4	0.2	0.175	0.175	0.125	0.1	0.1

Table 2: Cutoff values used for each graph generation, a function of n and dimension

approximately the n smallest weights, and since these are uniformly distributed on $[0, 1]$, the approximate value of each of these should be $\frac{n}{((n)(n-1)/2)} \approx \frac{1}{n}$ Thus the relationship of n weights of approximately $1/n$ is approximately $n \cdot \frac{1}{n} = 1$, giving the constant relationship seen. The added 1.2 factor probably comes from the $\frac{1}{n}$ being a slight underestimate for the average value of the included weights.

The other three dimensions each take the form of a polynomial with degree less than 1. This makes sense as there is an upper bound on how far apart an individual pair of points can be in this restricted area in 2, 3, or 4 space, meaning that the function can be at max linear/degree 1. With the growing number of points, however, the size of the MST increases more slowly, since the phenomenon of the unit square etc. "filling up" with points causes the distribution of points in the space to become more uniform, thus making it less likely that an "outlier" point not close to any of the others will need to be connected to with a large weight. The relationship among the 3 higher dimensions ($2 < 3 < 4$) can be explained by the typically larger distances between points in higher unit cubes in n-dimensions, thus taking longer for this "filling up" to take place.