CS 124 Programming Assignment 1: Spring 2021

Your name(s) (up to two): Yumi Yi

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: 2
No. of late days used after including this pset: 4

dimension\n	128	256	512	1024	2048	4096	8192	16384	32768	65536	131072	262144
0	1.661872	1.834337	1.942195	2.151831	2.418605	2.514846	2.776090	2.930284	3.133021			
2	8.454415	11.804102	16.980932	24.114902	34.158459	48.658951	69.330978	97.996933	139.608002			
3	19.124027	29.733328	47.628330	75.029991	118.324707	189.685654	298.890045	477.151215	757.451294			
4	29.616308	49.823723	83.557365	140.527908	233.205521	391.604797	659.594421	1101.382202	1850.682251			

The table for the average tree size for different values of n for each graph type is given above. 5 trials were done for each value of n. The entries for $n = 65536$, 131072, and 262144 are left blank because my program didn't return results quickly enough.

Functions:

My guesses for $f(n)$ are as follows:

dimension 0: $f(n) = \frac{\log n}{\sqrt{22}}$
dimension 2: $f(n) = 0.76\sqrt{n}$
dimension 3: $f(n) = \frac{\sqrt{n}\log n}{4.1}$
dimension 4: $f(n) = \sqrt{n\log n}$

Discussion:

There was a lot I learned from this assignment, not only about graph algorithms and heap data structures, but also about general coding practices and making good decisions in regards to programming open-ended projects like this one.

I decided to use Prim's algorithm instead of Kruskal's, because all the graphs in this assignment have every possible edge. Considering each algorithm's runtime in terms of the number of vertices n and edges m, Prim's is faster in cases where the graph has many edges (Prim's runtime is $O(m \log_{m/n} n)$, as opposed to Kruskal's $O(m \log m)$).

I also find it interesting how the various growth rates $f(n)$ look. I might be incorrect with my guesses, but in general, it appears that the growth rates increase as the dimension increases. I find this interesting because, for each value of n, the number of edges will be identical regardless of dimension, and yet the growth rates clearly seem to differ across dimensions. And along the same vein, my program also took noticeably longer for higher dimensions, even for the same value of n. The only explanation I can think of is the added work the program has to do as the dimension of the graph increases, such as calculating the weights of each edge. In particular, this would explain why dimension 0 has an especially slow growth rate, since edge weights are assigned at random and no calculations are necessary.

Finally, while it may not be directly related to the goals of the assignment, I found it extremely rewarding to work through this assignment. Even though I couldn't find all the data I needed, I learned a lot just trying to figure out working implementations of binary heaps and Prim's algorithm, and making various decisions while doing so. For example, I chose to index a majority of the arrays from 1 because I found it more intuitive to associate array indeces with vertices that way. I also struggled to figure out how to store key/value pairs in a binary heap in C at first. Ultimately, I decided to store only the vertices in the heap, and keep track of the distance values using the dist[] array I already had. Finally, I also decided I would keep track of which vertices were in the set S by keeping an array inS[] of 0s and 1s, where inS[v]=1 if the vertex v is in S, and inS[v]=0 if not. I figured these decisions would be the simplest for my purposes, though I'm sure there are more effective ones out there.