Table 1: Results for Values of n With $numtrials = 5, dimension = 4$

n	average $f(n)$	total time (seconds)
128	28.627	0.0165
256	46.509	0.075
512	76.25	0.300
1024	130.42	1.259
2048	216.420	5.378
4096	360.030	23.28
8192	602.38	97.43
16384	1009.21	403.647
32768	1686.93	1512.621

Our implementation used Kruskal's algorithm. We chose this over Prim's algorithm because it only requires tracking a list of all the edges in the tree as candidates, rather than updating the list of candidates on each iteration, which we found easier to implement. We also learned some optimizing tricks for Kruskal's algorithm in class, such as path compression and union by rank, both of which were implemented in our code.

Times for each run are listed in the table above. Each doubling of n corresponded to approximately a 4x increase in time. This makes sense because the number of edges that must be generated, sorted, and checked is quadratic in n, so it is reasonable for the time to increase quadratically as well.

At no point did we find the random number generator to create any aberrant behavior. We seeded with the current time on each iteration, so while the numbers were generated pseudorandomly, the sequence was different each time.