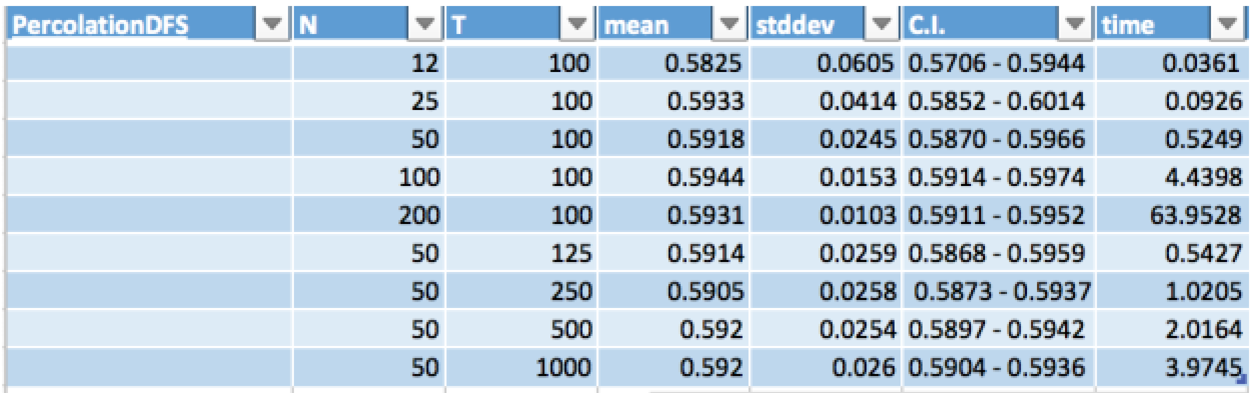
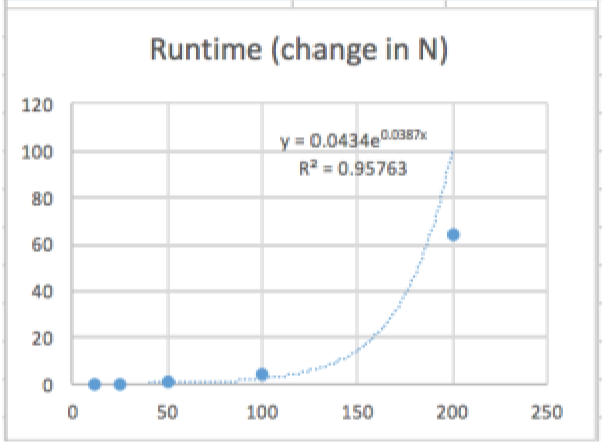
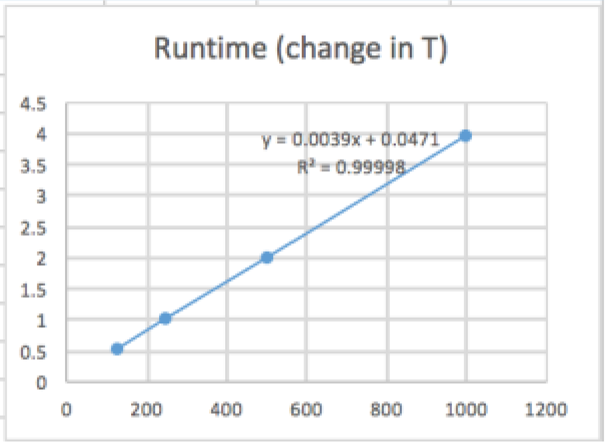
Percolation Analysis
Joel Mire

Percolation DFS



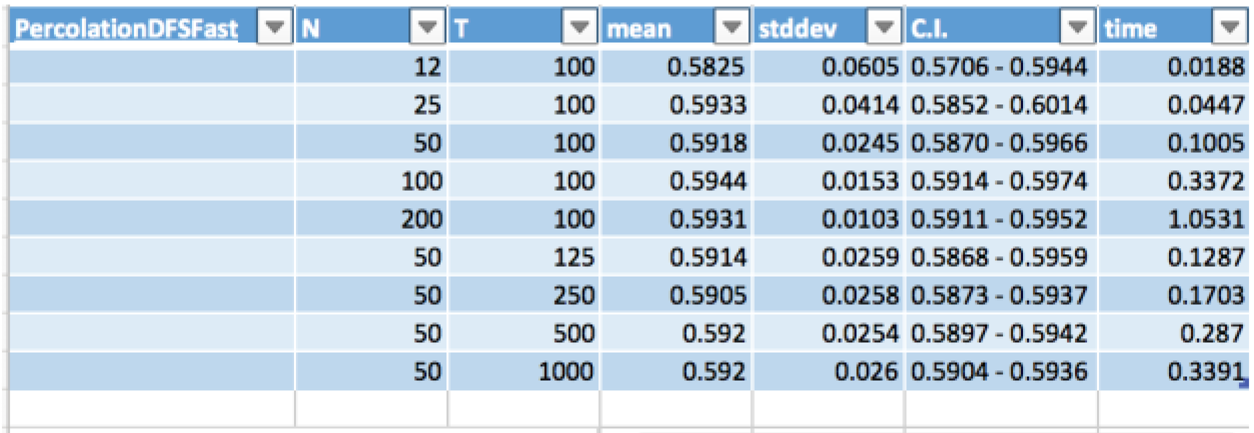
PercolationDFS	N	T	mean	stddev	C.I.	time
	12	100	0.5825	0.0605	0.5706 - 0.5944	0.0361
	25	100	0.5933	0.0414	0.5852 - 0.6014	0.0926
	50	100	0.5918	0.0245	0.5870 - 0.5966	0.5249
	100	100	0.5944	0.0153	0.5914 - 0.5974	4.4398
	200	100	0.5931	0.0103	0.5911 - 0.5952	63.9528
	50	125	0.5914	0.0259	0.5868 - 0.5959	0.5427
	50	250	0.5905	0.0258	0.5873 - 0.5937	1.0205
	50	500	0.592	0.0254	0.5897 - 0.5942	2.0164
	50	1000	0.592	0.026	0.5904 - 0.5936	3.9745



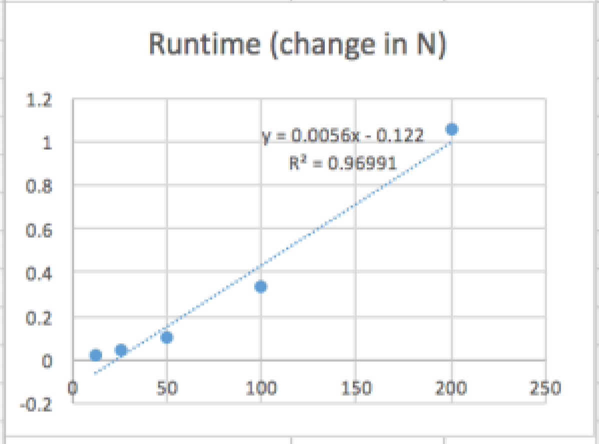


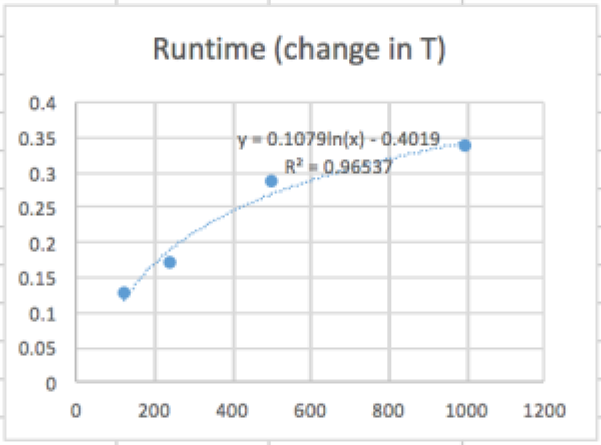
1. Based on these data points, runtime drastically increases as N doubles, suggesting that the runtime may increase by a power of 2.
2. As T doubles, runtime doubles as well. The graph and data points in the appended table and graph support this claim closely. For example, from T = 250 to T = 500, runtime increases from 1.0205 to 2.0164.
3. O(T * (N(N-1))^2) →O((N^4) * T)…….O(TN^4). DFS checks site in the grid (row, then column), so it is quite inefficient.
4. If N = 200 takes 64 seconds, and there are 86400 seconds in a day; then 86400 = 0.0434e^0.0387x. X = 374.781. Therefore, N = 374 size grid.
5. 4N^2 + 4

Percolation DFS Fast



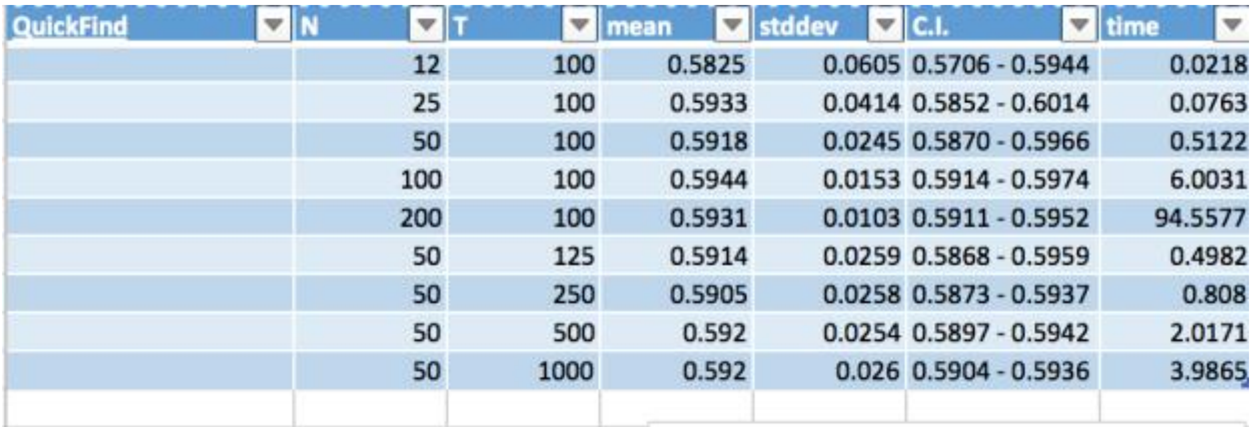
PercolationDFSFast	N	T	mean	stddev	C.I.	time
	12	100	0.5825	0.0605	0.5706 - 0.5944	0.0188
	25	100	0.5933	0.0414	0.5852 - 0.6014	0.0447
	50	100	0.5918	0.0245	0.5870 - 0.5966	0.1005
	100	100	0.5944	0.0153	0.5914 - 0.5974	0.3372
	200	100	0.5931	0.0103	0.5911 - 0.5952	1.0531
	50	125	0.5914	0.0259	0.5868 - 0.5959	0.1287
	50	250	0.5905	0.0258	0.5873 - 0.5937	0.1703
	50	500	0.592	0.0254	0.5897 - 0.5942	0.287
	50	1000	0.592	0.026	0.5904 - 0.5936	0.3391



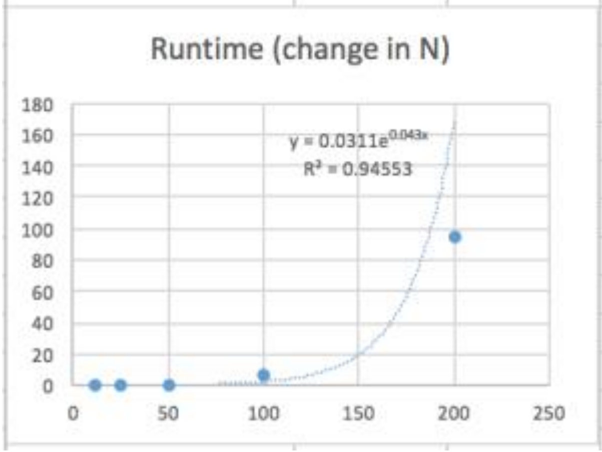


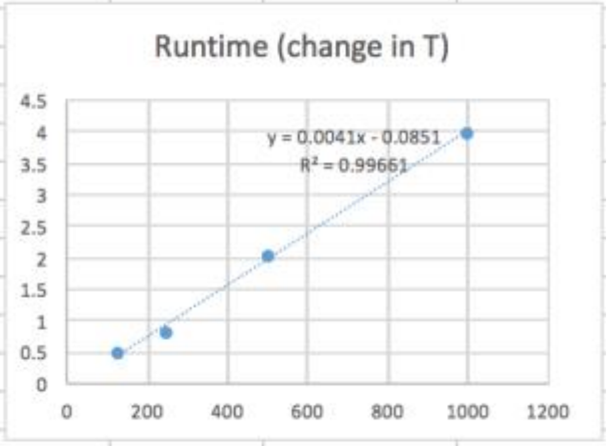
1. Based on the table above, doubling the grid size (N) causes the runtime to roughly triple. As N increases from 50 to 100, for example, the runtime increases from 0.1005 to 0.3372, which is an approximately 3x longer runtime.
2. Doubling the number of simulations (T) results in a runtime lengthened by a constant of roughly 0.5. For example, as N doubles twice, runtime takes two steps from 0.1287 to 0.1703 to 0.287, which is a total increase of 1 for two doubles. The appended graph supports that the relationship is linear (even though the line of best fit appears to be logarithmic…we would not expect that for T).
3. O(0.5T * 3N) → O(TN)
4. 86400 = 0.0056x – 0.122. X = 15428593.
   Therefore, N = 15428593 size grid.
5. 4N^2 + 4

Percolation UF QuickFind



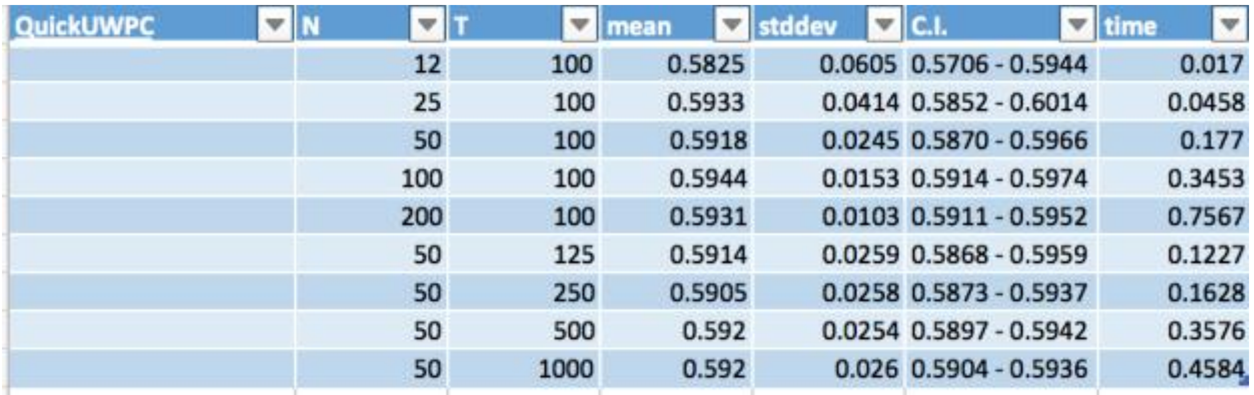
QuickFind	N	T	mean	stddev	C.I.	time
	12	100	0.5825	0.0605	0.5706 - 0.5944	0.0218
	25	100	0.5933	0.0414	0.5852 - 0.6014	0.0763
	50	100	0.5918	0.0245	0.5870 - 0.5966	0.5122
	100	100	0.5944	0.0153	0.5914 - 0.5974	6.0031
	200	100	0.5931	0.0103	0.5911 - 0.5952	94.5577
	50	125	0.5914	0.0259	0.5868 - 0.5959	0.4982
	50	250	0.5905	0.0258	0.5873 - 0.5937	0.808
	50	500	0.592	0.0254	0.5897 - 0.5942	2.0171
	50	1000	0.592	0.026	0.5904 - 0.5936	3.9865



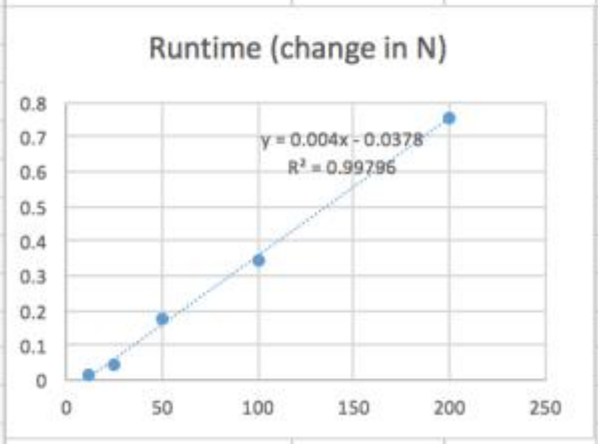


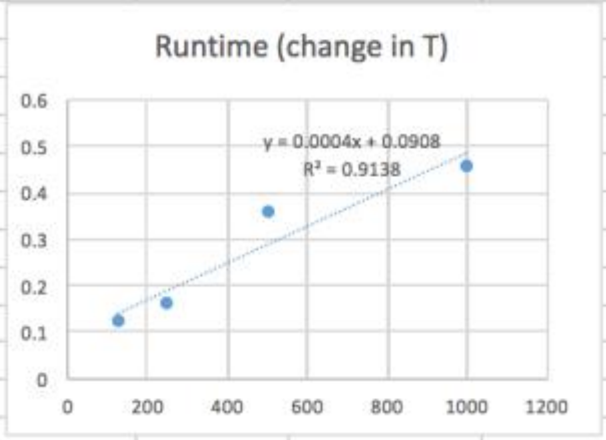
1. T QuickFind graph (above) appears to be exponential. For example, from N = 25 to N = 100, N has doubled twice while runtime has roughly doubled twice as well, from 0.0763 to 6.0031. However, it may show upon closer algorithmic analysis that my estimations are not even close to correct as N increases beyond what is reasonable to test for empirically. While these data points do not give me great confidence to predict the rate of increase as the program scales, 3N^2 provides a reasonable estimate as well.
2. Based on the table above, doubling T while keeping N constant causes runtime to double as well. For example, from T = 500 to T = 1000, runtime increases from 2.0171 to 3.9865, suggesting a linear relationship.
3. O(T * 3N^2) → O(TN^2)
4. 86400 = 0.0311e^0.043x. X = 291.5 . Therefore, size N = 291 grid.
5. (N^2 + 12) + (4N + 4) = N^2 + 4N + 16 (instantaneous variables from PercolationUF and QuickFind instantaneous variables as well)

Percolation UF QuickUWPC



QuickUWPC	N	T	mean	stddev	C.I.	time
	12	100	0.5825	0.0605	0.5706 - 0.5944	0.017
	25	100	0.5933	0.0414	0.5852 - 0.6014	0.0458
	50	100	0.5918	0.0245	0.5870 - 0.5966	0.177
	100	100	0.5944	0.0153	0.5914 - 0.5974	0.3453
	200	100	0.5931	0.0103	0.5911 - 0.5952	0.7567
	50	125	0.5914	0.0259	0.5868 - 0.5959	0.1227
	50	250	0.5905	0.0258	0.5873 - 0.5937	0.1628
	50	500	0.592	0.0254	0.5897 - 0.5942	0.3576
	50	1000	0.592	0.026	0.5904 - 0.5936	0.4584





1. As N doubles, runtime appears to double as well. For example, from N = 50 to N = 100, runtime approximately doubles from 0.177 to 0.3453.
2. Runtime increases at a rate of approximately (3/2)x. Based on the line of best fit superimposed on the graph above, from T = 200 to T = 800, we would expect runtime to nearly double, suggesting that 3/2 estimates result on runtime for a single doubling of T.
3. O(NT)
4. 86400 = 0.004x – 0.0378
   X = 21600095
   Therefore, size N = 21600095 grid.
5. (N^2 + 12) + (4N + 8) = N^2 + 4N + 20