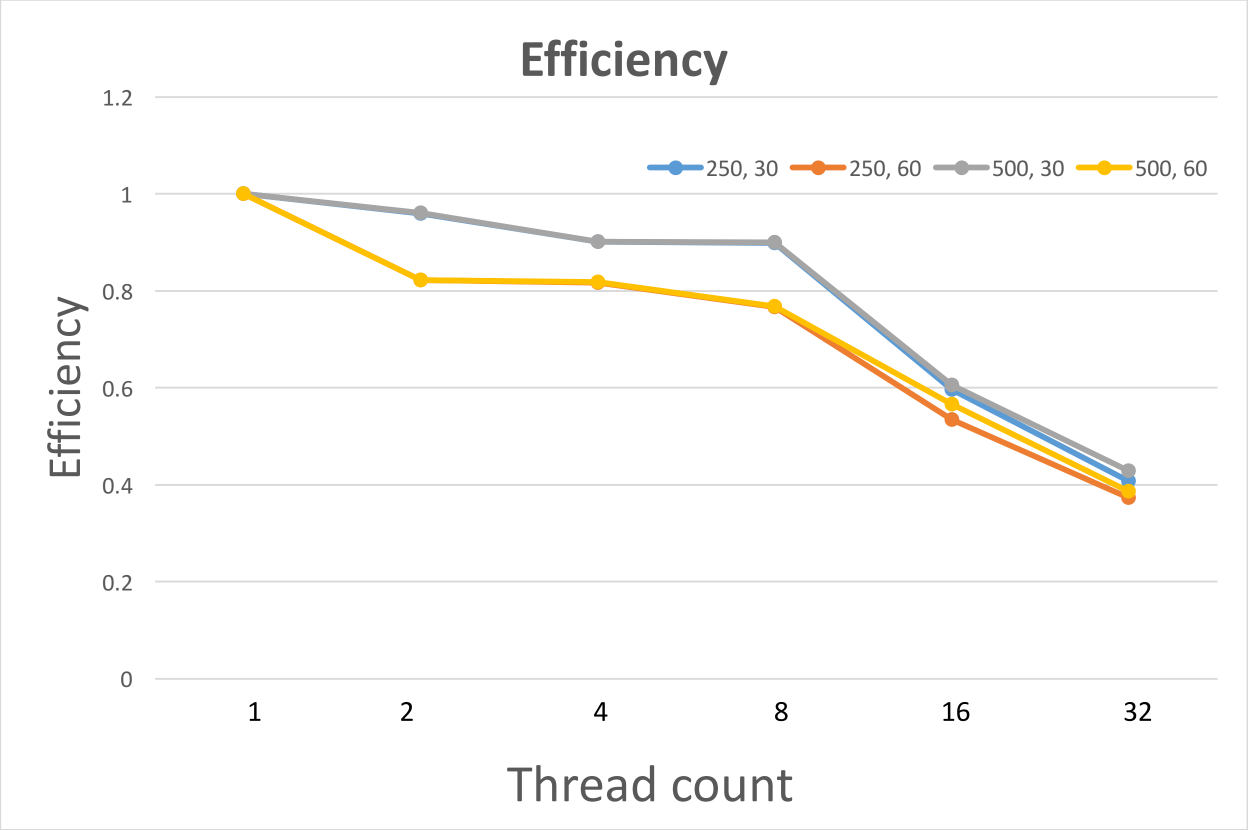
Project 3

Diego Loya
Jacqueline Moad



Creating and terminating threads takes a considerable amount of time. Since only part of the code is parallelized, there is more overhead and the efficiency is affected significantly as the thread count increases.

Shorter runtimes at 32 occur because each thread has its own chunk of data to compute and it writes back to its own spot it so there are no data races. For this reason, the code is also easily parallelized, and results in shorter runtime even with 32 threads.

Fractal Table:

		Number of threads					
Pixels	**Frames**	**1**	**2**	**4**	**8**	**16**	**32**
250x250	**30**	0.997s	0.519s	0.277s	0.139s	0.104s	0.076s
250x250	**60**	1.698s	1.033s	0.519s	0.277s	0.199s	0.142s
500x500	**30**	3.984s	2.074s	1.105s	0.553s	0.411s	0.290s
500x500	**60**	6.789s	4.129s	2.076s	1.104s	0.749s	0.548s

		Number of threads					
Pixels	**Frames**	**1**	**2**	**4**	**8**	**16**	**32**
250x250	**30**	1	0.960	0.901	0.898	0.597	0.409
250x250	**60**	1	0.822	0.817	0.766	0.535	0.373
500x500	**30**	1	0.961	0.902	0.901	0.606	0.429
500x500	**60**	1	0.822	0.818	0.768	0.566	0.387

TSP Table:

	Number of threads				
# of cities	**1**	**2**	**4**	**8**	**16**
1291	2.9446s	1.5134s	0.7737s	0.4028s	0.4250s
1379	7.6193s	3.9055s	1.9611s	1.0333s	1.0880s
1400	6.7002s	3.4230s	1.7248s	0.9068s	0.9497s

For 1291 and 1400 cities the times scale until you get to 16 cores where you lose efficiency. This could mean load imbalance because the more threads, the less scalable. Also, as the threads increase and cache is saturated, there is a bottleneck caused by the mutex.