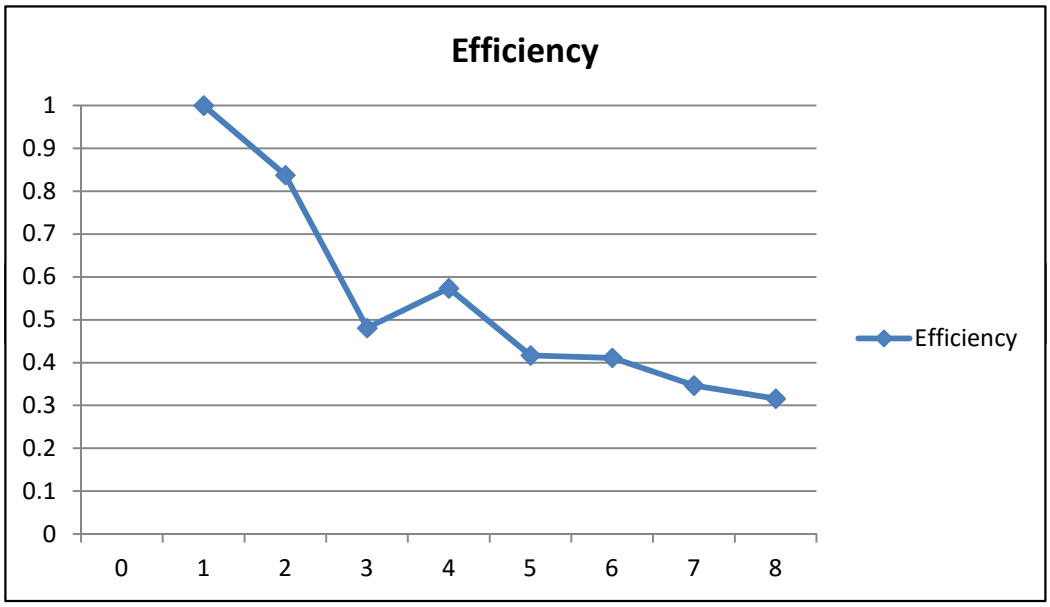
Matt Baker
A01536520

Homework #2
Report

	run 1		run 2		Average of r1 and r2			Speed up	Efficiency
	Average	Std. Dev	Average	Std. Dev	Average	Std. Dev			
1 Thread	1181	528	975	167	1078	347.5	**t1**	1	1
2 Threads	596	250	691	285	643.5	267.5	**t2**	1.68	0.84
3 Threads	647	290	848	371	747.5	330.5	**t3**	1.44	0.48
4 Threads	478	197	462	163	470	180	**t4**	2.29	0.57
5 Threads	527	188	507	205	517	196.5	**t5**	2.09	0.42
6 Threads	433	172	441	163	437	167.5	**t6**	2.47	0.41
7 Threads	435	135	453	168	444	151.5	**t7**	2.43	0.35
8 Threads	418	138	435	131	426.5	134.5	**t8**	2.53	0.32

Time is in the measure of: Milliseconds
Each run was the average of 30 executions of a mandelbrot .cpp

The following graph illustrates how adding more threads decreases efficiency:



My findings:
The use of threads greatly increased the run time of the program that I wrote. For example, only running one thread took roughly one second to make a Mandelbrot set while the use of two threads reduced the time to around two-thirds of a second.
I was also surprised to notice how little adding more processors effected the run time after roughly four threads. As we can see in the Efficiency column above, with every addition of a new thread we lose on efficiency, making the additional threads counter-productive.