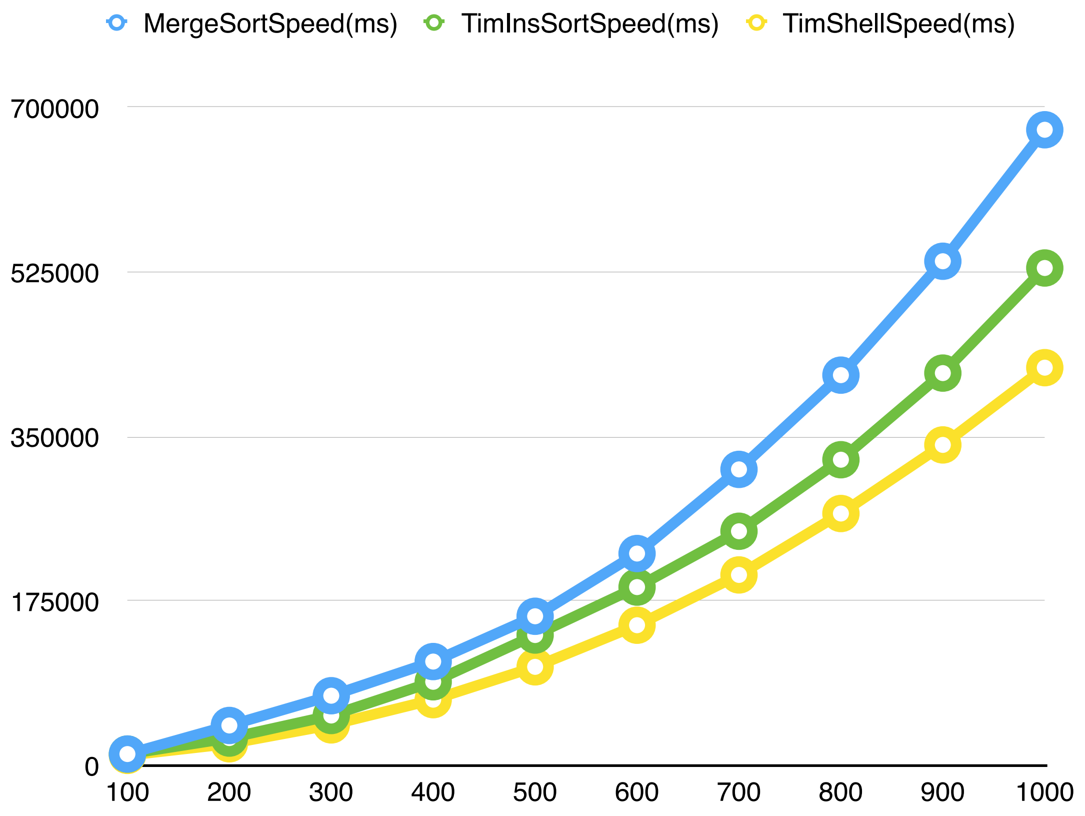
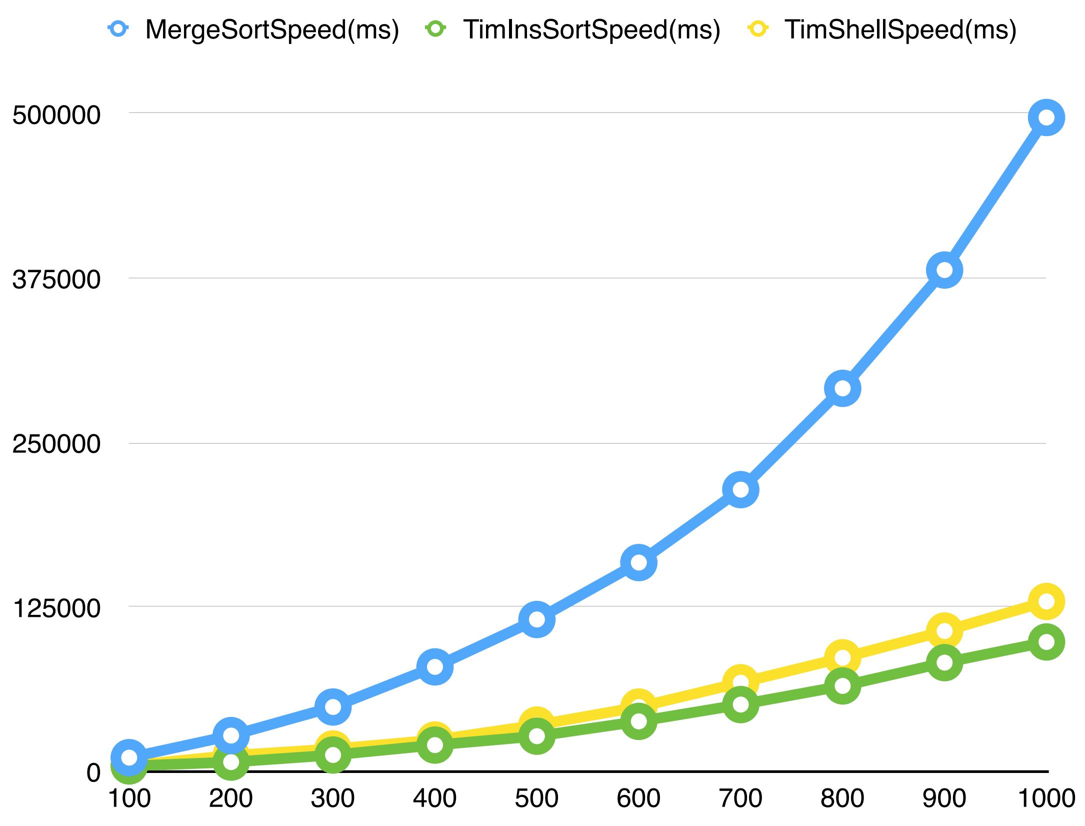
I created a test that creates a list of [100,200,…1000] random array and run each 1000 times. Time used was logged. Difference in running time is not big, Mergesort terms to be a little bit slower than Timsorts

SIZE(Random)*run1Ktimes	MergeSortSpeed(ms)	TimInsSortSpeed(ms)	TimShellSpeed(ms)
100	12324	12320	10920
200	42744	29328	23244
300	74412	53352	43368
400	110760	89232	70200
500	158964	138996	105144
600	225732	190008	149448
700	315276	249444	202956
800	415584	325572	268320
900	536796	417768	341328
1000	676728	529620	423540



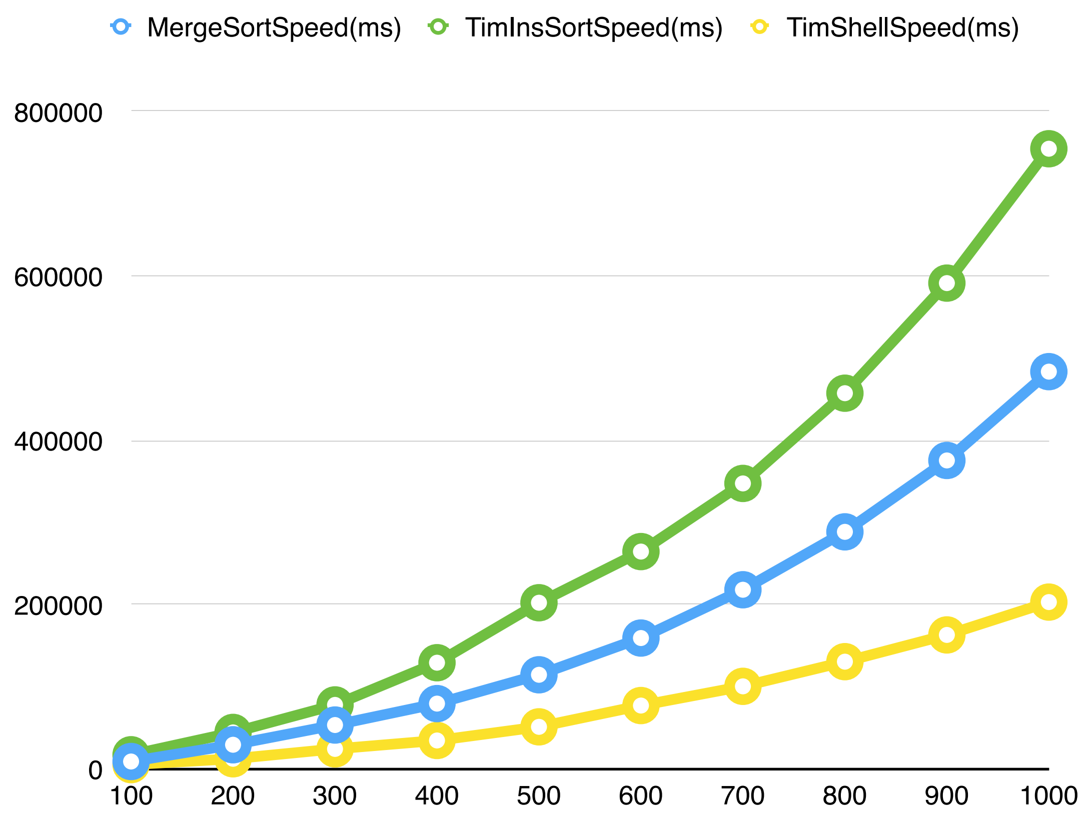
I created a test that creates a list of [100,200,…1000] random array in sorted (ascend order) and run each 1000 times. Time used was logged. This time both Timsorts has a much better performance than Mergesort because when subarray gets under 64 it utilizes a sorting algorithm friendly to sorted elements.

SIZE(Ascend)*run1Ktimes	MergeSortSpeed(ms)	TimInsSortSpeed(ms)	TimShellSpeed(ms)
100	10626	4312	4620
200	27412	7238	12320
300	49126	12628	17248
400	79618	20174	24024
500	115654	26950	35420
600	158928	38192	49126
700	214368	50974	67606
800	291368	65142	86548
900	381304	82852	106876
1000	497266	98560	129360



I created a test that creates a list of [100,200,…1000] random array in reverse sorted (descend order) and run each 1000 times. Time used was logged. This time Shell-based timsort is better than Mergesort but Insertion-based is much worse. This is probably due to regular insertion sort is very bad at sorting an array in reverse order.

SIZE(Descend)*run1Ktimes	MergeSortSpeed(ms)	TimInsSortSpeed(ms)	TimShellSpeed(ms)
100	9322	17538	5530
200	29704	44714	12640
300	53720	78210	24648
400	79790	129560	34918
500	114866	202556	51508
600	159580	264808	77420
700	218356	347758	100804
800	288824	457568	130824
900	375724	591552	163372
1000	483796	755082	203346



How did I implement timsort?

For both Timsort, I make them subclass of Mergesort and override the sort() method.

Instead of partitioning right-away, I use index to detect the length of the subarray. If it is smaller than 64, I implement shell/insertion sort instead. Otherwise, I do partition as usual.