Array Size	Merge	TMerge 2	TMerge 4	TMerge 16	TMerge 32
1	0	0	0	0	0
2	0	59	0	0	0
4	0	38	52	0	0
8	0	47	63	1	0
16	1	43	55	198	2
32	4	50	55	205	388
64	8	40	67	198	388
128	18	46	63	201	396
256	39	56	60	212	403
512	85	84	89	225	449
1024	181	138	142	260	476
2048	404	269	250	348	572
4096	826	489	516	496	742
8192	2003	987	1047	849	1028
16384	3735	2052	1610	1622	2042
32768	7934	4265	3271	3169	3592
65536	16791	9096	6435	6663	7211
131072	34810	18413	13130	13026	14303
262144	72910	38593	26892	26328	29179
524288	153746	82356	53679	52579	58401
1048576	321836	171643	152034	204117	164039

The speed ***improved*** for larger arrays and a certain number of threads.

The time taken to sort each subarray is significant for large arrays. By parallelizing this step, you reduce the overall sorting time substantially. Dividing the work across multiple threads reduces the computational load per thread. Thread creation and management introduce overhead. For small arrays, this overhead can dominate the total execution time, making threaded versions slower. With larger arrays, the sorting work becomes much more significant than the threading overhead. Sorting enormous arrays benefits from efficient memory access patterns.

The most challenging was ensuring optimal performance. The merge phase requires careful coordination. Merging sorted subarrays in parallel without overwriting shared data or causing race conditions is complex. The size of the subarrays to merge decreases as the algorithm

progresses, making it challenging to distribute work evenly among threads. Designing an algorithm that efficiently utilizes threads at each merging level is not intuitive.

The easiest part was parallelizing the initial sorting of the subarrays. Each thread works on a distinct portion of the array without communicating or synchronizing with other threads while sorting its subarray. Since threads do not interfere with each other's data, you do not have to implement complex synchronization mechanisms for this phase. I know this is not the most optimized way, but it was easy to implement to get something working quickly.