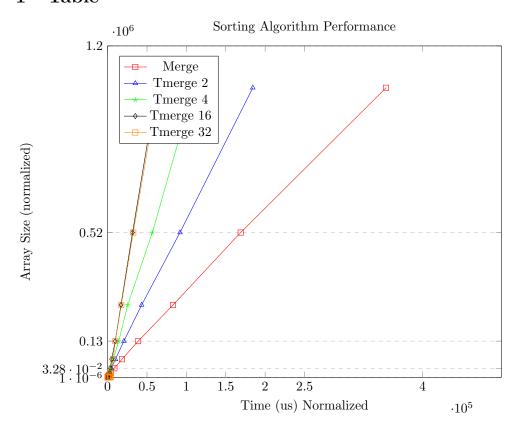
# CS361 Homework 4

#### Nick Pelletier

5/5/2023

### 1 Table



# 2 Images and answer to question 2

Notice that In the plot we can only see performance at large values. These are the values you'd see at a data center or in general when using large data. As you can see Tmerge32 and Tmerge16 are quite close to each other in my test (this can be partially attributed to the logic I used to determine the correct

number of threads to use).

Below I will show a couple screenshots of table examples from running my code:

Delow I will bllow			те ехатргев п	om rammig n	iy code.					
Passed 10 out										
Time in Microseconds shown. Time to create arrays is not counted. Only sorting.										
				·						
Array Size	Merge	TMerge 2	TMerge 4	TMerge 16	TMerge 32					
1	Θ	Θ	Θ	Θ	0	l				
2	12	267	Θ	Θ	Θ	I				
4	Θ	98	99	Θ	Θ	I				
8	1	180	131	1	Θ					
16	2	135	378	321	2	İ				
32	4	123	334	875	729	İ				
64	j 9 j	139	330	1764	1586	İ				
128	20	113	434	1627	3364	İ				
256	41	121	273	1388	3279	İ				
512	92	209	348	1624	3914	į				
1024	195	268	352	1565	3323	i				
2048	439	353	465	1859	3653	İ				
4096	875	589	914	2006	3661	į				
8192	1933	1124	987	1871	2897	į				
16384	4036	2336	1835	2282	3793	i				
32768	8591	4762	3703	3408	4926	İ				
65536	18207	9858	7008	5045	6950	ĺ				
131072	38769	20652	14123	9630	9700	İ				
262144	83108	43179	25491	16959	17377	İ				
524288	168895	91850	56800	31599	32552	İ				
1048576	353565	184351	109175	62201	64191	İ				
nwp28@tux2 hw	4>									

The above screenshot is the current plot. Seeing this in more detail, we can tell that my method wasn't as good as standard merge sort between the start and 16384 array size. It was only after that (at array size 32768) that my Thread merge sorting method outshone the regular merge sorting method. Of course you can see through these that my usage of threads was quite optimized as the more threads that were given resulted in faster run times.

The next couple screenshots are 2 other results that prove the above statement.

	seconds show					
Time to creat	e arrays is	not counted.	Only sorting	g.		
Array Size	Merge	TMerge 2	TMerge 4	TMerge 16	TMerge 32	
1	Θ	Θ	Θ	Θ	Θ	
2	1	198	Θ	Θ	Θ	
4	0	249	132	0	Θ .	ĺ
8	1	116	336	Θ	Θ .	
16	1	111	328	248	1	İ
32	3	109	264	1550	708	ĺ
64	9	145	287	1350	2788	j l
128	19	175	286	1290	2703	j l
256	41	149	294	1418	3059	j l
512	105	177	295	1420	3239	j
1024	194	266	392	1546	3255	j l
2048	409	402	536	1772	3688	j
4096	881	772	646	1918	3852	j
8192	1910	1402	951	5267	3776	j
16384	4137	2288	1637	2501	4177	Ī
32768	8910	4918	3058	3404	5187	j
65536	17576	9964	6124	5126	5761	
131072	37011	20761	12352	8280	9680	j l
262144	78370	42631	25472	15193	16804	j l
524288	163914	88976	52164	30347	30341	j l
1048576	342177	184927	107708	59000	60520	j l
nwp28@tux2 hw	4>					
Passed 10 out	of 10 Tosts	,,,,,				
Passed 10 out						
Time in Micros	seconds shown	١.	Only sorting			
Time in Micros	seconds shown arrays is n	n. not counted.			TMerne 32	
Time in Micros Time to create   Array Size	seconds shown arrays is n Merge	n. not counted. TMerge 2	TMerge 4	TMerge 16	TMerge 32	
Time in Micros Time to create   Array Size     1	seconds shown a arrays is r Merge   0	n. not counted. TMerge 2   0	TMerge 4   0	TMerge 16   0	<b>΄</b> Θ	
Time in Micros Time to create   Array Size     1     2	seconds showr e arrays is r Merge   0   1	n. not counted. TMerge 2   0   443	TMerge 4   0   0	TMerge 16   0   0	9   9	
Time in Micros Time to create   Array Size     1     2     4	seconds showr e arrays is r Merge   0   1   0	n. not counted. TMerge 2   0   443   307	TMerge 4   0   0   104	TMerge 16   0   0   0	0   0   0	
Time in Micros Time to create   Array Size     2     4     8	seconds showr e arrays is r Merge   0   1   0   0	n. not counted. TMerge 2   0   443   307   137	TMerge 4   0   0   104   358	TMerge 16   0   0   0   1	9   9   9   1	
Time in Micros Time to create   Array Size     2     4     8	seconds shown e arrays is r Merge   0   1   0   0   2	n. not counted. TMerge 2   0   443   307   137   129	TMerge 4   0   0   104   358   327	TMerge 16   0   0   0   1   236	0   0   0   1   2	
Time in Micros Time to create Array Size 1 2 4 8 16 32	seconds shown e arrays is r Merge   0   1   0   0   2   4	n. not counted. TMerge 2   0   443   307   137   129   125	TMerge 4   0   0   104   358   327   246	TMerge 16   0   0   0   1   236   1531	0   0   0   1   2   692	
Time in Micros Time to create   Array Size     2	seconds shown e arrays is r Merge   0   1   0   0   2   4	n. not counted. TMerge 2   0   443   307   137   129   125   155	TMerge 4   0   0   0   0   104   358   327   246   268	TMerge 16   0   0   0   0   0   1   1   236   1531   1300	0   0   0   1   2   692   2979	
Time in Micros Time to create Array Size 1 2 4 8 16 32 64 128	seconds shown e arrays is r Merge   0   1   0   2   4   10	n. not counted. TMerge 2   0   443   307   137   129   125   155	TMerge 4   0   0   0   104   358   327   246   268   405	TMerge 16   0   0   0   0   1   1   236   1531   1300   1483	0   0   0   1   2   692   2979   2975	
Time in Micros Time to create   Array Size     1	seconds shown e arrays is r Merge   0   1   0   2   4   10   19   42	n. not counted. TMerge 2   0   443   307   137   129   125   155   174	TMerge 4   0   0   0   104   358   327   246   268   405   266	TMerge 16   0   0   0   0   1   1   236   1531   1300   1483   1686	0   0   0   1   2   692   2979   2975   2910	
Time in Micros Time to create   Array Size     1	seconds shown e arrays is r Merge   0   1   0   2   4   10   19   42   89	n. not counted. TMerge 2   0   443   307   137   129   125   155   174   143	TMerge 4   0   0   0   104   358   327   246   268   405   266   254	TMerge 16   0   0   0   0   1   1   236   1531   1300   1483   1686   1477	0   0   0   1   2   692   2979   2975   2910	
Time in Micros Time to create   Array Size     1	seconds shown e arrays is r Merge   0   1   0   2   4   10   19   42   89	n. not counted. TMerge 2   0   443   307   137   129   125   155   174   143   183   226	TMerge 4   0   0   0   104   358   327   246   268   405   266   254   296	TMerge 16   0   0   0   0   0   1   236   1531   1300   1483   1686   1477   1276	0   0   0   1   2   692   2979   2975   2910   2909   2603	
Time in Micros Time to create   Array Size     1	seconds shown e arrays is r Merge   0   1   0   2   4   10   42   89   189	n. not counted. TMerge 2   0   443   307   137   129   125   155   174   143   183   226   347	TMerge 4   0   0   0   104   358   327   246   268   405   266   254   296   414	TMerge 16   0   0   0   0   1   1   236   1531   1300   1483   1686   1477   1276   1312	0   0   0   1   2   692   2979   2975   2910   2909   2603   2798	
Time in Micros Time to create   Array Size     1	seconds shown e arrays is r Merge   0   1   0   2   4   10   19   42   89   189   407   876	n. not counted. TMerge 2   0   443   307   137   129   125   155   174   143   226   347   602	TMerge 4   0   0   0   104   358   327   246   268   405   254   296   414   592	TMerge 16   0   0   0   0   1   236   1531   1300   1483   1686   1477   1276   1312   1367	0   0   0   1   2   692   2979   2975   2910   2909   2603   2798   2969	
Time in Micros Time to create   Array Size     1	seconds shown e arrays is r Merge   0   1   0   2   4   10   19   42   89   189   407   876	n. not counted. TMerge 2   0   443   307   137   129   125   155   174   143   226   347   602   1132	TMerge 4   0   0   0   104   358   327   246   266   254   296   414   592   918	TMerge 16   0   0   0   0   1   236   1531   1300   1483   1686   1477   1276   1312   1367   1743	0   0   0   1   2   692   2979   2975   2910   2909   2603   2798   2969   3064	
Time in Micros Time to create   Array Size     1	seconds shown e arrays is r Merge   0   1   0   2   4   10   19   42   89   189   407   876   1880	n. not counted. TMerge 2   0   443   307   137   129   125   155   174   143   226   347   602   1132   2476	TMerge 4   0   0   0   104   358   327   246   268   405   266   254   414   592   918   1702	TMerge 16   0   0   0   0   1   236   1531   1300   1483   1686   1477   1276   1312   1367   1743   2277	0   0   0   1   2   692   2979   2975   2910   2909   2603   2798   2969   3064   3595	
Time in Micros Time to create   Array Size     1	seconds shown e arrays is r Merge   0   1   0   2   4   10   19   42   89   407   876   1880   4004   8355	n. not counted. TMerge 2   0   443   307   137   129   125   155   174   143   226   347   602   1132   2476   5148	TMerge 4   0   0   0   104   358   327   246   268   405   266   254   414   592   918   1702   3112	TMerge 16   0   0   0   0   1   236   1531   1300   1483   1686   1477   1276   1312   1367   1743   2277   3379	0   0   0   1   2   692   2979   2975   2910   2909   2603   2798   2969   3064   3595   4975	
Time in Micros Time to create   Array Size     1	seconds shown e arrays is r Merge   0   1   0   2   4   10   19   42   89   407   876   1880   4004   8355	n. not counted. TMerge 2   0   443   307   137   129   125   155   174   143   226   347   602   1132   2476   5148   10095	TMerge 4   0   0   0   104   358   327   246   268   405   266   254   296   414   592   918   1702   3112   6053	TMerge 16   0   0   0   0   0   1   236   1531   1300   1483   1686   1477   1276   1312   1367   1743   2277   3379   6043	0   0   0   0   1   2   1   2   692   2979   2975   2910   2909   2603   2798   2969   3064   3595   4975   6362	
Time in Micros Time to create   Array Size     1	seconds shown a arrays is reference   Merge   0   1   0   0   2   4   10   19   42   89   189   407   876   1880   4004   8355   18023   38185	n. not counted. TMerge 2   0   443   307   137   129   125   155   174   143   183   226   347   602   1132   2476   5148   10095   20960	TMerge 4   0   0   0   104   358   327   246   268   405   266   254   296   414   592   918   1702   3112   6053   12341	TMerge 16   0   0   0   0   1   236   1531   1300   1483   1686   1477   1276   1312   1367   1743   2277   3379   6043   8150	0   0   0   0   0   1   2   1   2   692   2979   2975   2910   2909   2603   2798   2969   3064   3595   4975   6362   9394	
Time in Micros Time to create   Array Size     1	seconds shown a arrays is referenced and seconds shown a arrays is referenced and seconds are seconds and seconds and seconds are seconds and seconds and seconds are seconds and seconds and seconds are seconds and seconds and seconds are seconds and seconds and seconds are seconds and seconds and seconds are seconds and seconds are seconds and seconds are seconds and seconds are seconds and seconds are seconds and seconds are seconds and seconds are seconds are seconds and seconds are seconds and seconds are seconds and seconds are seconds	n. not counted. TMerge 2   0   443   307   137   129   125   155   174   143   183   226   347   602   1132   2476   5148   10095   20960   41265	TMerge 4   0   0   0   104   358   327   246   268   405   266   254   296   414   592   918   1702   3112   6053   12341   24257	TMerge 16   0   0   0   0   1   236   1531   1300   1483   1686   1477   1276   1312   1367   1743   2277   3379   6043   8150   15561	0   0   0   0   0   1   2   1   2   692   2979   2975   2910   2909   2603   2798   2969   3064   3595   4975   6362   9394   17483	
Time in Micros Time to create   Array Size     1	seconds shown a arrays is reference   Merge   0   1   0   0   2   4   10   19   42   89   189   407   876   1880   4004   8355   18023   38185	n. not counted. TMerge 2   0   443   307   137   129   125   155   174   143   183   226   347   602   1132   2476   5148   10095   20960	TMerge 4   0   0   0   104   358   327   246   268   405   266   254   296   414   592   918   1702   3112   6053   12341	TMerge 16   0   0   0   0   1   236   1531   1300   1483   1686   1477   1276   1312   1367   1743   2277   3379   6043   8150	0   0   0   0   0   1   2   1   2   692   2979   2975   2910   2909   2603   2798   2969   3064   3595   4975   6362   9394	

Now where did this improvement come from? When the array size got larger, instead of more recursion with a linear wait-time the thread method split up the recursion into N threads that did their own recursion and resulted in a reduction of time (not completely /n due to laws/algorithms in relation to threading that were explained in the early weeks of class) that can be plainly seen.

# 3 What was the most difficult portion

Funnily enough, the part that took the most time was the bug fixing and other mistakes like accidentally dividing to create 0 when optimizing thread usage at low array sizes.

### 4 What was the easiest portion

The easiest portion was compiling my thoughts into something feasible. I took about 1h of thinking on how to make my tmergesort and make it optimal. I got a result I was expecting, but not fully expecting at lower array size values.