PROGRAM STRUCTURES & ALGORITHMS

ASSIGNMENT-4 (PARALLEL SORTING)

NAME: SAI KUMAR GANGA
NUID : 002191288

TASK:

A cutoff (defaults to, say, 1000) which you will update according to the first argument in the command line when running. It's your job to experiment and come up with a good value for this cutoff. If there are fewer elements to sort than the cutoff, then you should use the system sort instead.

Recursion depth or the number of available threads. Using this determination, you might decide on an ideal number (*t*) of separate threads (stick to powers of 2) and arrange for that number of partitions to be parallelized (by preventing recursion after the depth of *lg t* is reached)

An appropriate combination of these.

OUTPUT OF THE ARRAY SIZE IS 1000000

HAVE ATTACHED THE SPREADSHEET

OBSERVARTION:

In execution of these tasks, I have observed that all threads are showing the similar pattern
The experiment has been done on the array size which is of 100,10000 and 100000.
In Conclusion, we can say that the degree of 6 would give a optimal case which is efficient.

OUTPUT GRAPH:

Datasize	1000000										
Sum of Time	**Column Labels**										
Row Labels	**1**	**2**	**4**	**8**	**16**	**32**	**64**	**128**	**256**	**512**	**1024**
510000	786	606	583	621	595	658	591	604	629	634	1319
520000	761	552	576	568	588	607	578	545	582	578	592
530000	810	564	560	596	542	544	623	641	570	560	573
540000	774	578	575	561	574	574	608	562	563	572	560
550000	797	561	691	560	547	557	647	560	559	604	575
560000	765	562	604	585	526	553	691	629	576	577	560
570000	782	575	634	566	544	566	698	687	561	546	549
580000	763	561	671	583	559	608	656	602	586	561	561
590000	783	565	609	561	575	580	628	558	578	567	540
600000	784	555	623	576	554	562	591	578	558	563	560
610000	766	573	637	631	561	544	574	545	576	563	574
620000	800	574	713	664	582	568	584	575	607	561	557
630000	796	588	609	604	554	607	569	554	586	558	547
640000	791	561	616	559	600	576	613	582	576	565	577
650000	796	575	635	576	620	530	681	655	561	564	604
660000	857	562	687	560	569	564	642	656	600	558	583
670000	787	567	615	577	546	571	637	694	751	563	589
680000	800	591	652	568	546	551	692	641	575	561	551
690000	832	559	626	582	579	591	666	665	561	578	572
700000	792	564	691	554	615	547	669	625	587	561	577
710000	802	585	699	594	561	560	654	684	590	544	573
720000	855	576	684	573	559	575	623	659	603	569	616
730000	786	577	604	574	546	562	634	693	609	548	640