Report

Assignment-2

Vivek Kumar(B18092)

Bhumanyu Goyal(B18012)

Harish Jaglan(B18115)

**Q3 Multithreaded In-place merge sort**

**Idea**

Convert the threads to the nearest no. of threads suitable to form a perfect tree, then use the normal function of merge sort with following changes:-

1. If the level in perfect tree(mergeSort tree) is greater than or equal to last level of perfect tree use mergeSort function without any threads
2. Else form 2 threads, for [left, mid] and [mid+1, hi] mergesort
3. Use *pthread\_join* to wait for both subarrays to get sorted
4. Merge the sorted subarrays in the same thread and return

**Note**:-

1. For measuring time of execution of mergeSort with threading use Wall clock time not CPU clock time.
2. Benefit of multithreading is observed for arrays of very large size, most likely because of overhead in thread creation

**Graphs**

X-axis → #threads, Y-axis → time

Fig1. Array size=1e2, thread\_step= +1

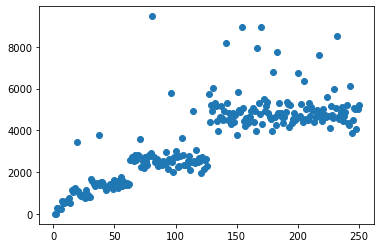


Fig2. Array size=1e4, thread\_step= +1

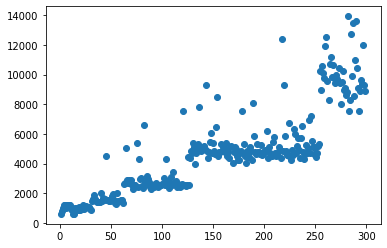


Fig3. Array size=1e7, thread\_step= +1

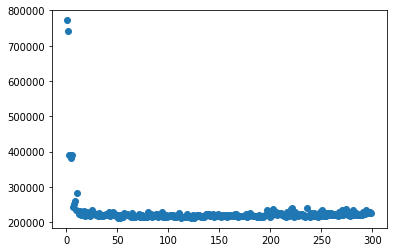
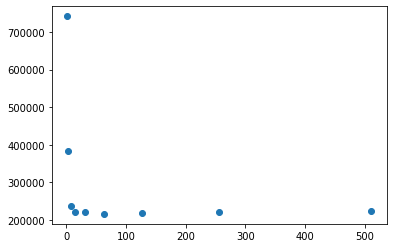


Fig4. Array size=1e7, thread\_step= x2



**Conclusion**

As observed from graphs, multithreading decreases execution time for arrays of size 1e7(or large array) while for others multithreading has more time than 1 thread execution.