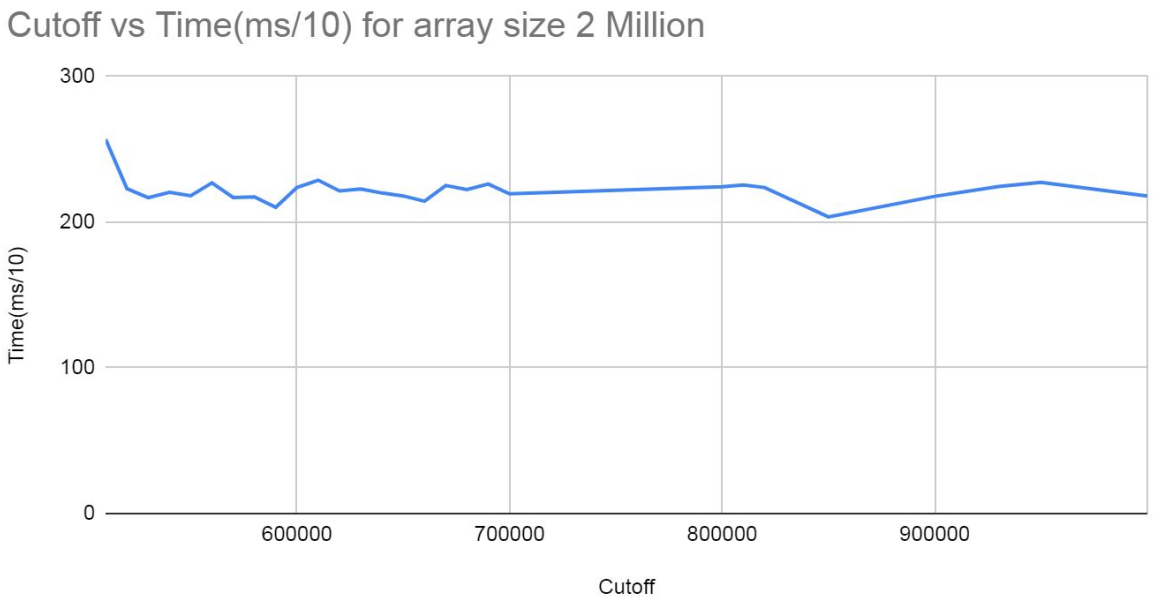
NUID: 001567040 (Suhani Shah)

Assignment 5

1. Parallelsort for array size 2 Million

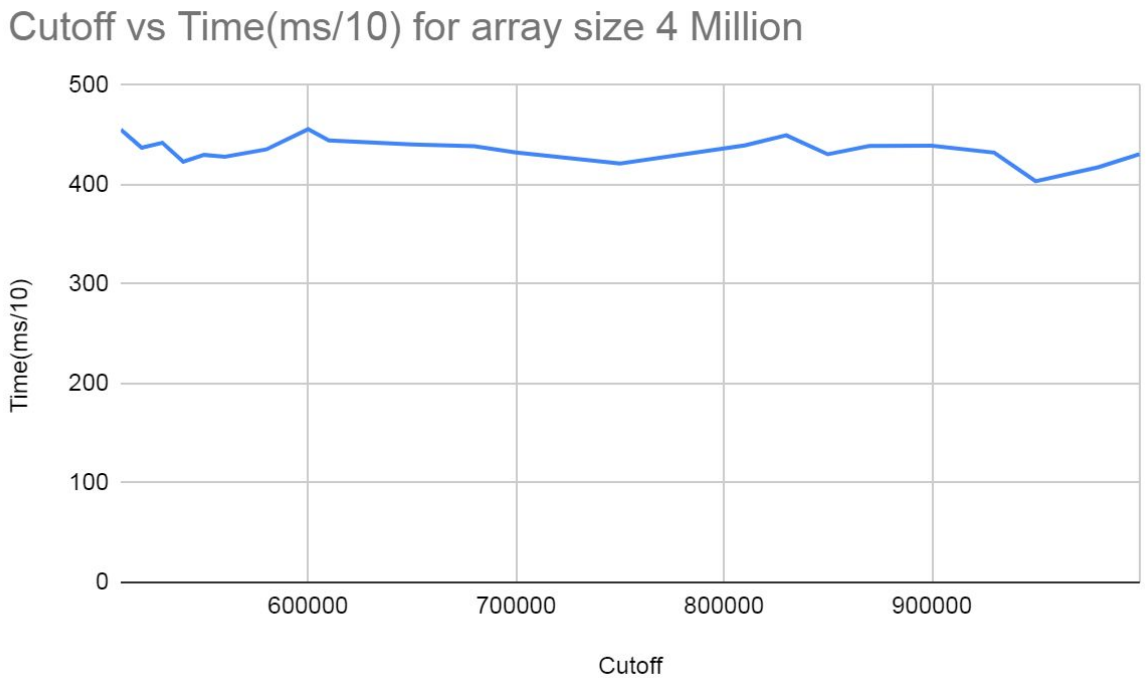
```
<terminated> Main [Java Application] C:\Program Files\Java\jre1.8.0_
Degree of parallelism: 3
cutoff:510000           10times Time:2464ms
cutoff:520000           10times Time:2357ms
cutoff:530000           10times Time:2252ms
cutoff:540000           10times Time:2327ms
cutoff:550000           10times Time:2120ms
cutoff:560000           10times Time:2436ms
cutoff:570000           10times Time:2322ms
cutoff:580000           10times Time:2363ms
cutoff:590000           10times Time:2380ms
cutoff:600000           10times Time:2580ms
cutoff:610000           10times Time:2331ms
cutoff:620000           10times Time:2531ms
cutoff:630000           10times Time:2404ms
cutoff:640000           10times Time:2343ms
cutoff:650000           10times Time:2371ms
cutoff:660000           10times Time:2133ms
cutoff:670000           10times Time:2341ms
cutoff:680000           10times Time:2338ms
cutoff:690000           10times Time:2299ms
cutoff:700000           10times Time:2169ms
cutoff:710000           10times Time:2374ms
cutoff:720000           10times Time:2134ms
cutoff:730000           10times Time:2367ms
cutoff:740000           10times Time:2216ms
cutoff:750000           10times Time:2228ms
cutoff:760000           10times Time:2477ms
cutoff:770000           10times Time:2276ms
cutoff:780000           10times Time:2330ms
cutoff:790000           10times Time:2301ms
cutoff:800000           10times Time:2297ms
cutoff:810000           10times Time:2276ms
cutoff:820000           10times Time:2220ms
cutoff:830000           10times Time:2265ms
cutoff:840000           10times Time:2255ms
cutoff:850000           10times Time:2232ms
cutoff:860000           10times Time:2421ms
```

NUID: 001567040 (Suhani Shah)



2. Parallel sort for array size 4 Million

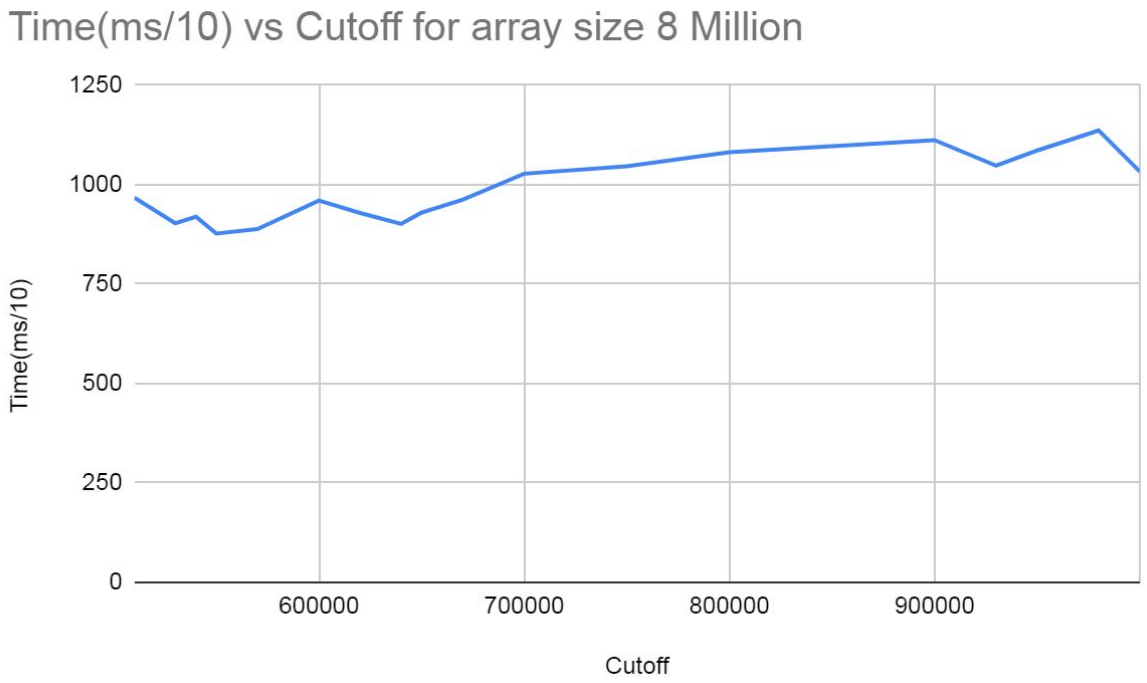
```
cutoff:510000           10times Time:4891ms
cutoff:520000           10times Time:4091ms
cutoff:530000           10times Time:4108ms
cutoff:540000           10times Time:3937ms
cutoff:550000           10times Time:4176ms
cutoff:560000           10times Time:4083ms
cutoff:570000           10times Time:3980ms
cutoff:580000           10times Time:4117ms
cutoff:590000           10times Time:3937ms
cutoff:600000           10times Time:4214ms
cutoff:610000           10times Time:4059ms
cutoff:620000           10times Time:3971ms
cutoff:630000           10times Time:4096ms
cutoff:640000           10times Time:3981ms
cutoff:650000           10times Time:3942ms
cutoff:660000           10times Time:4169ms
cutoff:670000           10times Time:4269ms
cutoff:680000           10times Time:4210ms
cutoff:690000           10times Time:4083ms
cutoff:700000           10times Time:4164ms
cutoff:710000           10times Time:4146ms
cutoff:720000           10times Time:4081ms
cutoff:730000           10times Time:4157ms
cutoff:740000           10times Time:4127ms
cutoff:750000           10times Time:4119ms
cutoff:760000           10times Time:4203ms
cutoff:770000           10times Time:4232ms
```



3. Parallelsort for array size 8 Million

8 Million array size	
Cutoff	Time(ms/10)
510000	967.5
530000	903
540000	919.5
550000	877.3
570000	888.6
600000	960.3
620000	928.9
640000	901.2
650000	929.7
670000	962.1
700000	1027.7
750000	1046.6
800000	1081.6
900000	1112
930000	1048.2
950000	1086.1
980000	1136.6
1000000	1033.3

NUID: 001567040 (Suhani Shah)



Conclusion:

- For small size array, the size less than the cutoff, Arrays.sort() method is used to sort elements as for small primitive arrays, Arrays.sort() method internally calls Quick sort.

- But for large array sizes, like 2 Million or more, Parallel merge sort is used. There are different cutoffs which define the smallest part of the array and every time when array size is halved, it is compared with cutoff to sort the array with Arrays.sort() method.

- For small arrays, Arrays.sort() is called which internally invokes quick sort because quick sort's performance is better for small primitive arrays than merge sort.

- In parallel merge sort, **Thread** is used for each divide part and after sorting of each sub array, conquer is called. So parallel merge sort is also a divide and conquer algorithm.