The ParSort Class is modified from:

if (to - from < cutoff) Arrays.sort(array, from, to);

to:

if (to - from <= cutoff) Arrays.sort(array, from, to);

to distinguish (to-from)=cutoff from (to-from)>cutoff.

The program is test with 2000000 elements with cutoff from 10000 to 2000000 at the step of 10000.

A screenshot of a computer

Description automatically generated with low confidence

The program is running under 11 threads.

Text

Description automatically generated

Which means, theoretically, there can be at most 11 sub-arrays sorting at O(nlogn) asynchronously.

Notably, when cutoff grows from 490000 to 500000:

Graphical user interface

Description automatically generated with low confidence

There is a significant enhancement in the running time.

Similarly, when cutoff grows from 990000 to 1000000:

Graphical user interface

Description automatically generated with low confidence

There is also a significant enhancement in the running time.

Same for cutoff from 1990000 to 2000000.

Text

Description automatically generated with low confidence

The reason behind the result set forth is the relationship between cutoff and the number of sub-arrays:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Cutoff | 2m | 1.99-1m | 990k-500k | 490k-250k | … |
| Sub-arrays | 1 | 2 | 4 | 8 | … |

The more sub-arrays are sorting asynchronously, there is less total running time *per se*.

However, when cutoff grows from 240000 to 250000:

A picture containing text

Description automatically generated

Or from 120000 to 130000:

Graphical user interface

Description automatically generated with low confidence

There is no significant enhancement in running time.

Also, when the cutoff grows from 2000 to 30000 at the step of 1000:

Table

Description automatically generated

Total running time decreases as the cutoff grows on the grounds that copying time grows as the cutoff decreases.

(Graph in the next page)

All in all, the program meets its peak performance with the cutoff allows more threads to run asynchronously but does not induce redundant array copy.