Srividya Burra(002985163)

**Program Structures & Algorithms**

**Fall 2021**

**Assignment No. 4**

* **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).

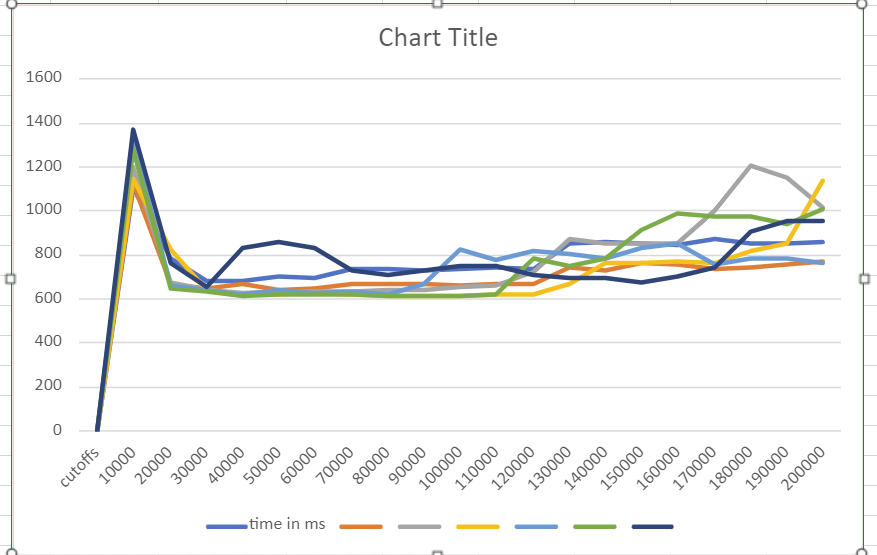
An appropriate combination of these.

* Implemented parallel sort and running with threads ranging in 2 powers (1,2,4,8,16,32,64).
* Taken cut-off values as 10,000 which increments 10,000 each time for 20 iterations.
* Plotted graph for all the above results.
* **Relationship Conclusion:**

**🡪** For the initial cut-offs we observe the time taken to be the maximum**.**

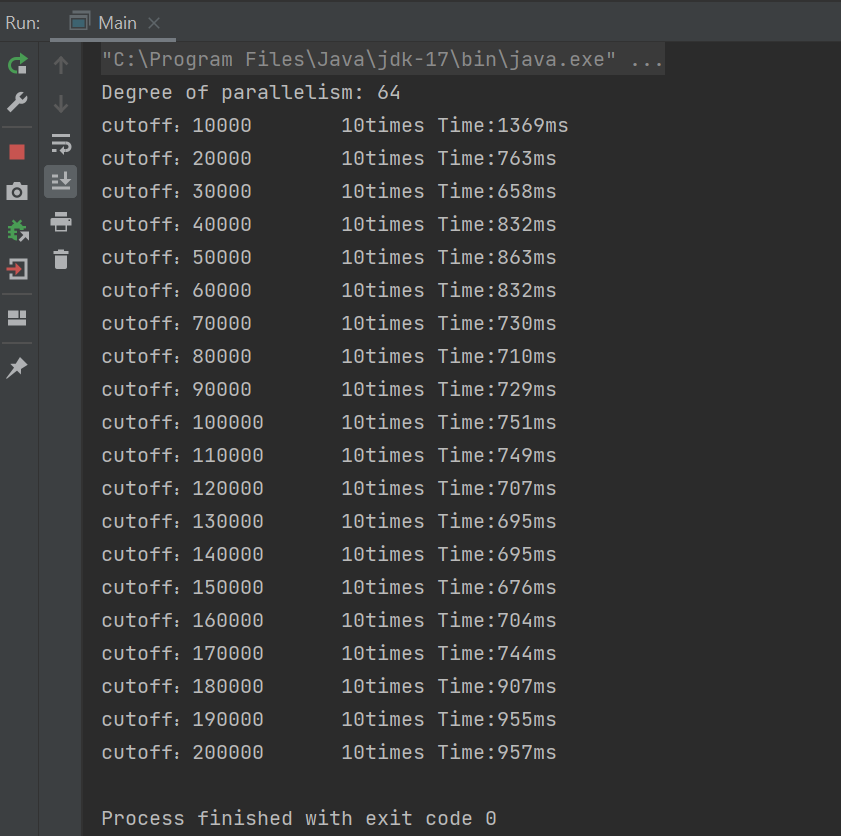
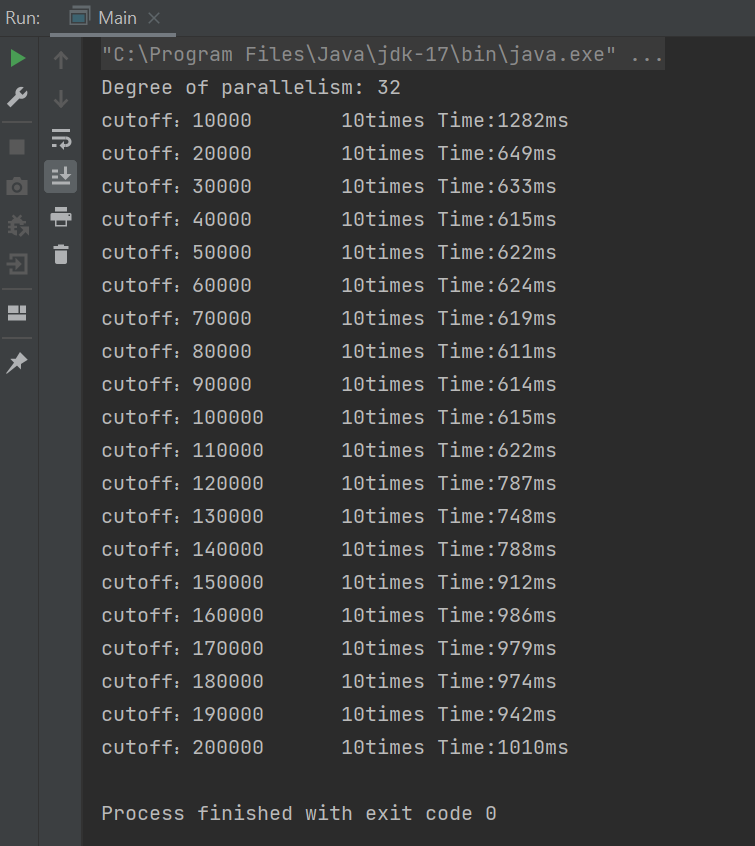
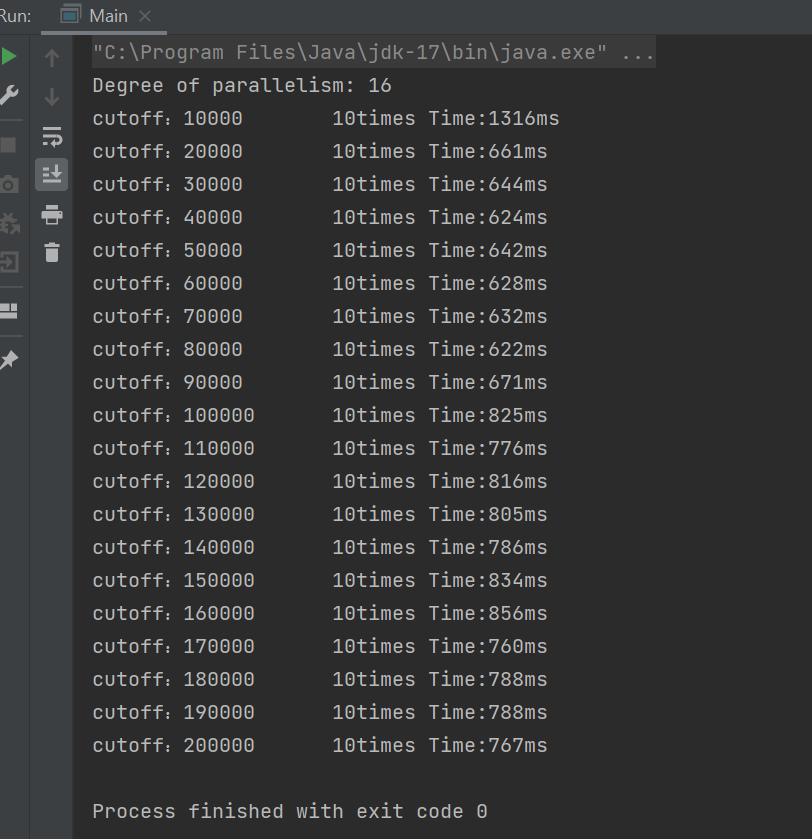
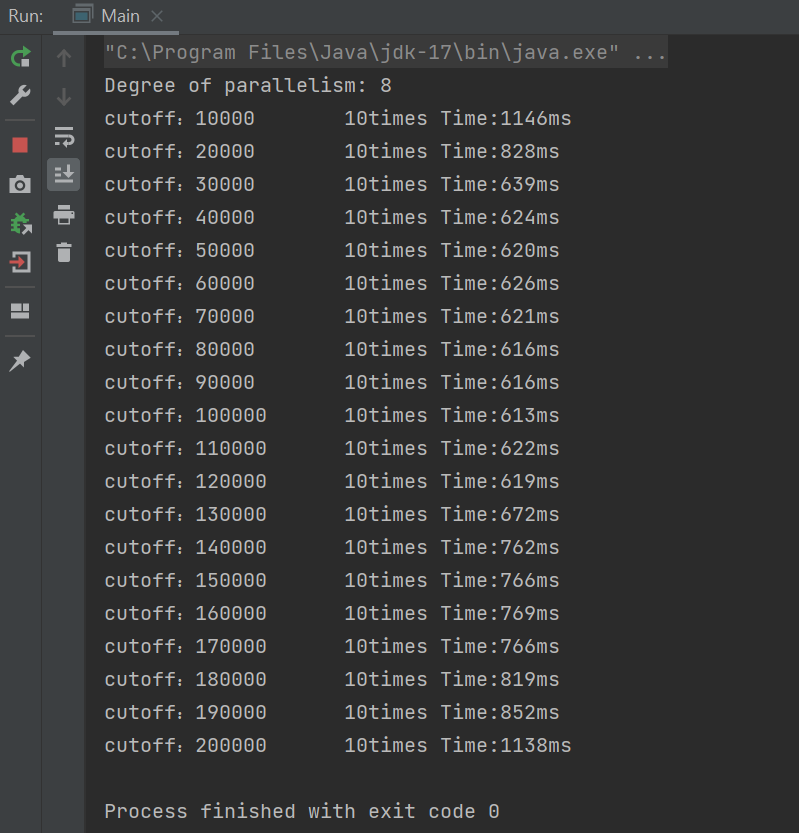
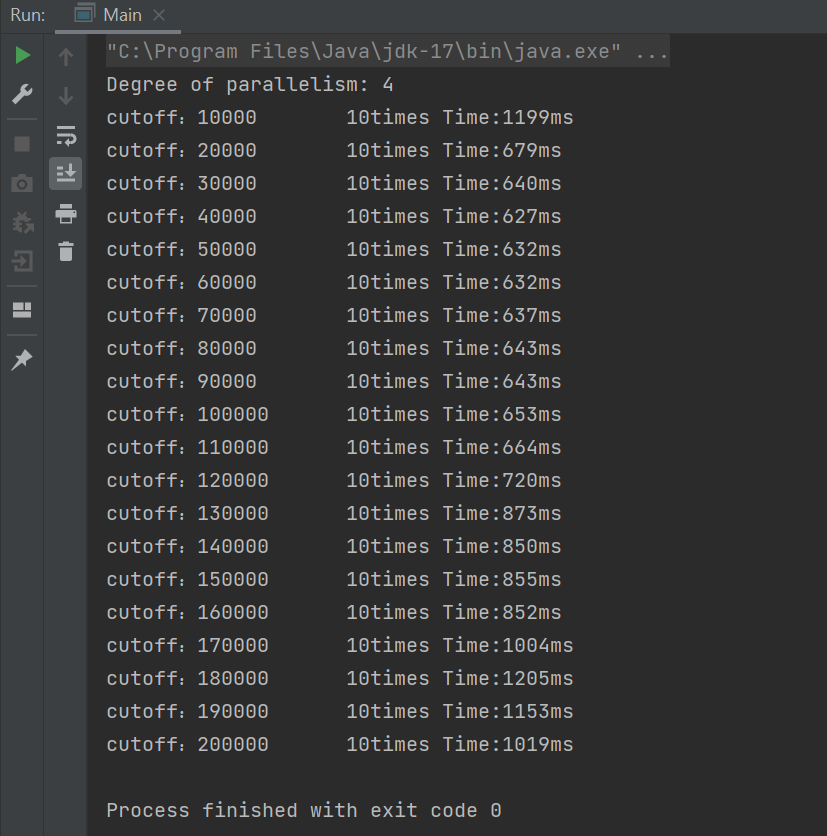
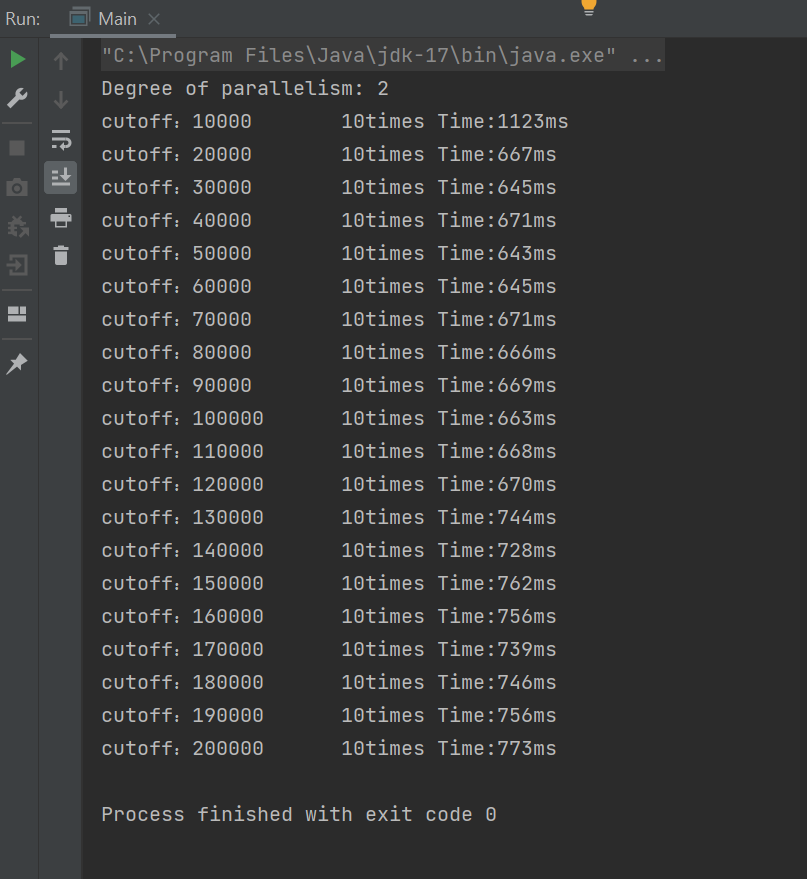
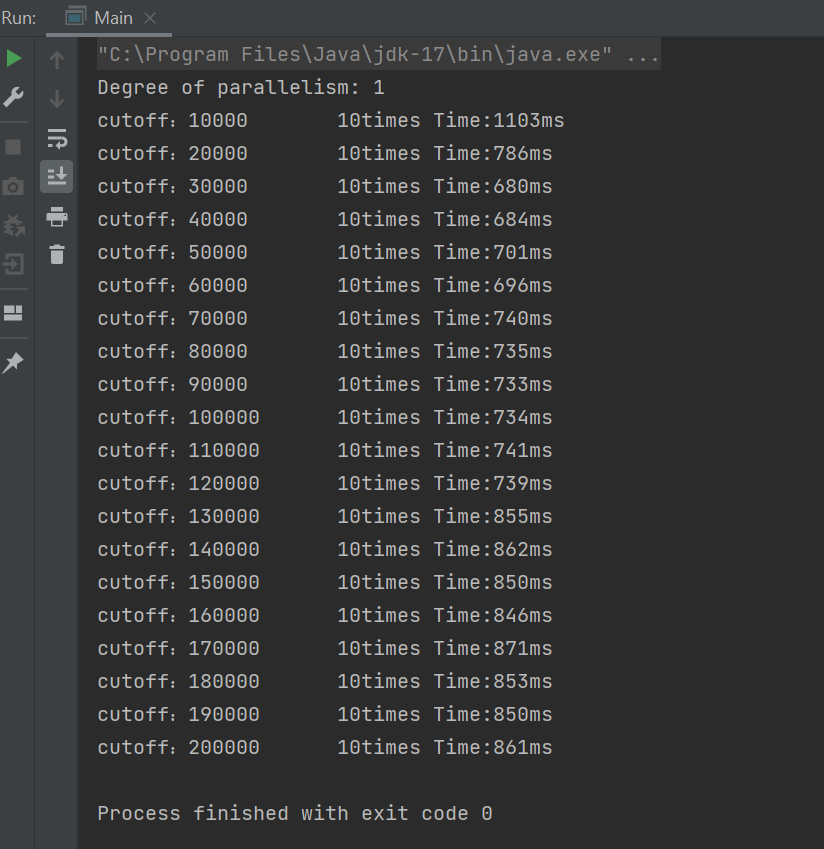
**🡪** As the cut-offs increase, the time taken reduces reaching its minimum values from 30,000 to 1,00,000, and later as the cutoffs increase further we see that time taken increases again.

* We can conclude that the parallel sort algorithm has worse performance for very low cut-offs as well as very high. It gives a better performance in between.
* **Evidence to support the conclusion:**

****

X-axis : cut-offs

Y-axis : time taken in milliseconds

**Output (Snapshot of Code output in the terminal)**

1. **Graphical Representation(Observations from experiments should be tabulated and analyzed by plotting graphs(usually in excel) to arrive on the relationship conclusion)**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | time in milli-seconds |  |  |  |  |  |  |
| cutoffs | 1 thread | 2 threads | 4 threads | 8 threads | 16 threads | 32 threads | 64 threads |
| 10000 | 1103 | 1123 | 1199 | 1146 | 1316 | 1282 | 1369 |
| 20000 | 786 | 667 | 679 | 828 | 661 | 649 | 763 |
| 30000 | 680 | 645 | 640 | 639 | 644 | 633 | 658 |
| 40000 | 684 | 671 | 627 | 624 | 624 | 615 | 832 |
| 50000 | 701 | 643 | 632 | 620 | 642 | 622 | 863 |
| 60000 | 696 | 645 | 632 | 626 | 628 | 624 | 832 |
| 70000 | 740 | 671 | 637 | 621 | 632 | 619 | 730 |
| 80000 | 735 | 666 | 643 | 616 | 622 | 611 | 710 |
| 90000 | 733 | 669 | 643 | 616 | 671 | 614 | 729 |
| 100000 | 734 | 663 | 653 | 613 | 825 | 615 | 751 |
| 110000 | 741 | 668 | 664 | 622 | 776 | 622 | 749 |
| 120000 | 739 | 670 | 720 | 619 | 816 | 787 | 707 |
| 130000 | 855 | 744 | 873 | 672 | 805 | 748 | 695 |
| 140000 | 862 | 728 | 850 | 762 | 786 | 788 | 695 |
| 150000 | 850 | 762 | 855 | 766 | 834 | 912 | 676 |
| 160000 | 846 | 756 | 852 | 769 | 856 | 986 | 704 |
| 170000 | 871 | 739 | 1004 | 766 | 760 | 979 | 744 |
| 180000 | 853 | 746 | 1205 | 819 | 788 | 974 | 907 |
| 190000 | 850 | 756 | 1153 | 852 | 788 | 942 | 955 |
| 200000 | 861 | 773 | 1019 | 1138 | 767 | 1010 | 957 |