**INFO 6205**

**Program Structures & Algorithms**

**Fall 2020**

**Assignment No 5**

* **Task**

Your task is to implement a parallel sorting algorithm such that each partition of the array is sorted in parallel. You will consider two different schemes for deciding whether to sort in parallel.

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.

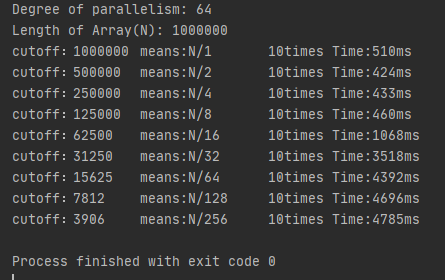
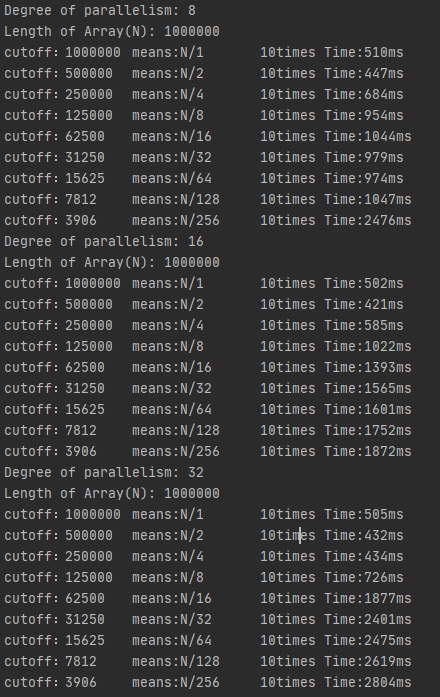
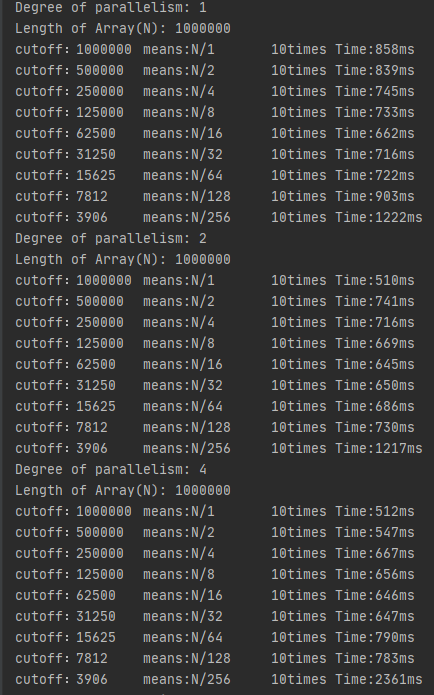
Recursion depth or the number of available threads. Using this determination, you might decide on an ideal number (t) of separate threads (stick to powers of 2) and arrange for that number of partitions to be parallelized (by preventing recursion after the depth of lg t is reached).

An appropriate combination of these.

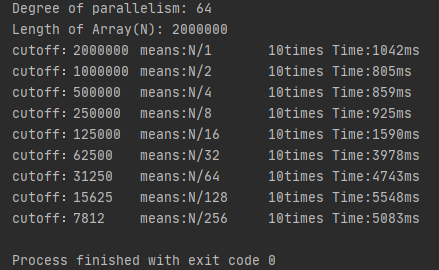
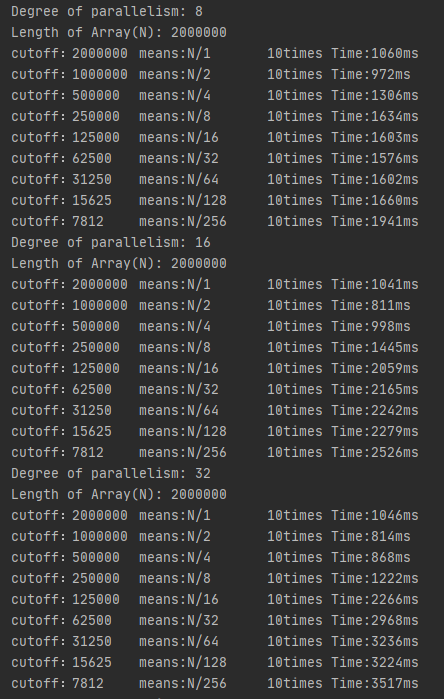
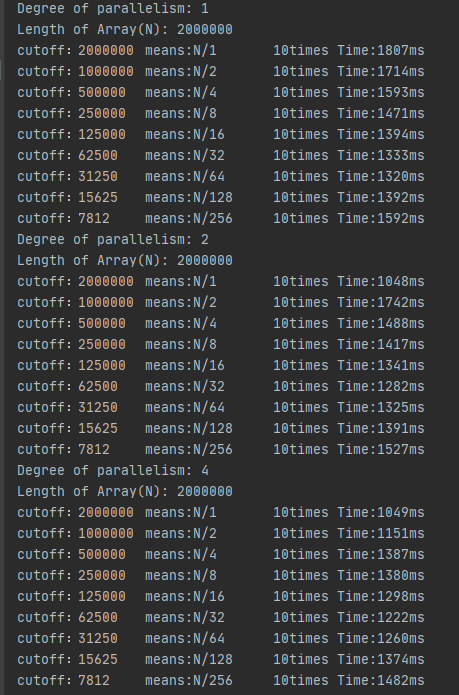
You must prepare a report that shows the results of your experiments and draws a conclusion (or more) about the efficacy of this method of parallelizing sort. Your experiments should involve sorting arrays of sufficient size for the parallel sort to make a difference. You should run with many different array sizes (they must be sufficiently large to make parallel sorting worthwhile, obviously) and different cutoff schemes.

* **Output** (few outputs to prove relationship)

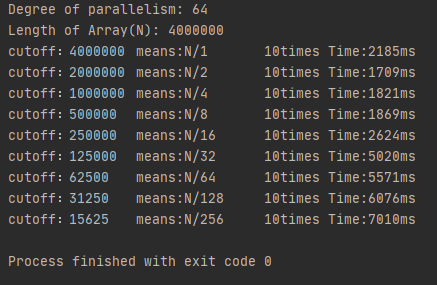
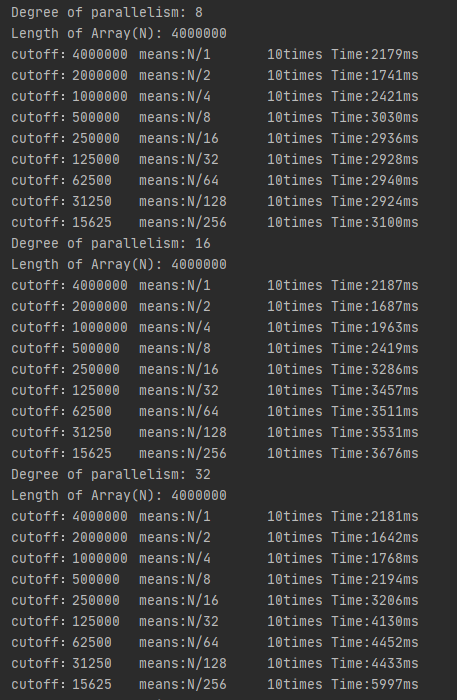
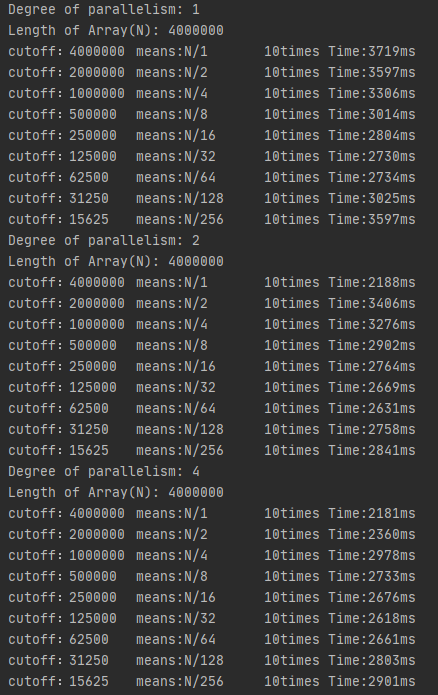
Length of Array (N):1000000



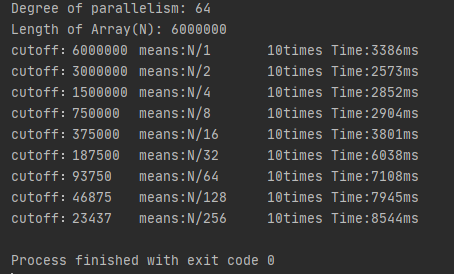
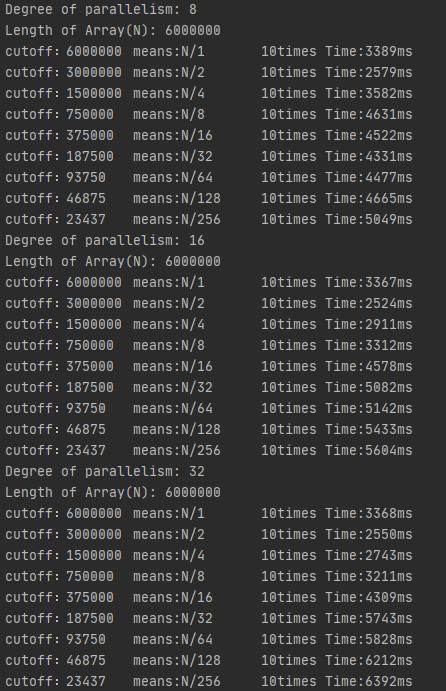
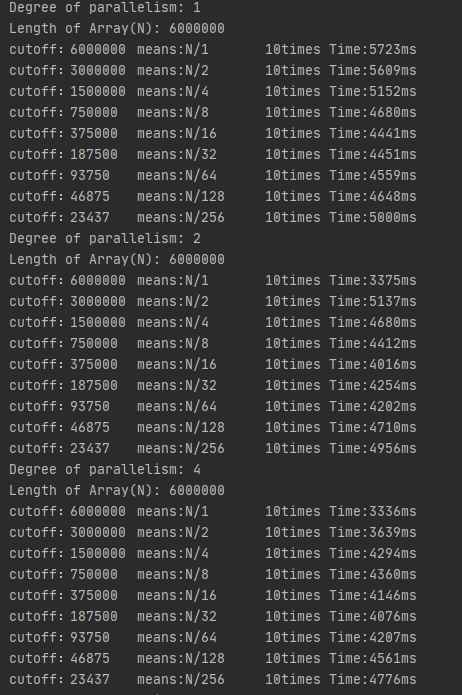
Length of Array (N):2000000



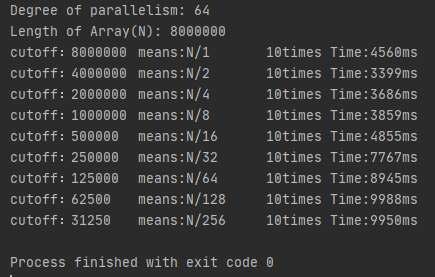
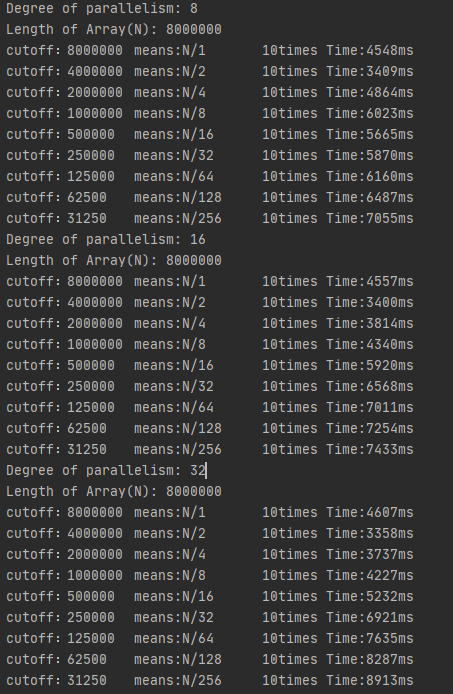
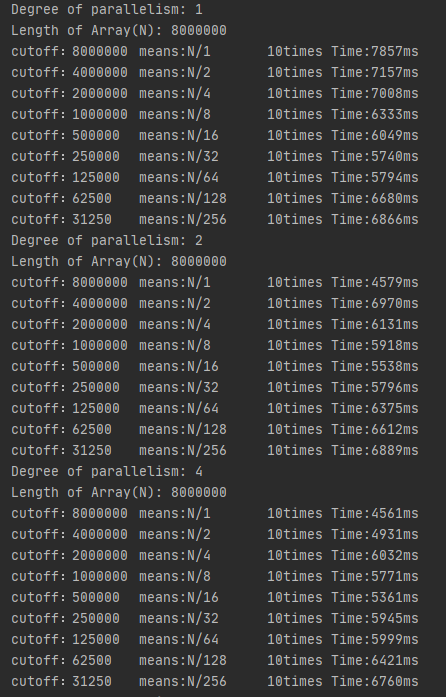
Length of Array (N):4000000



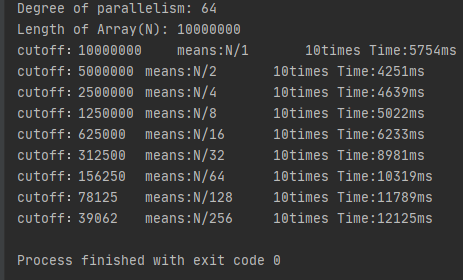
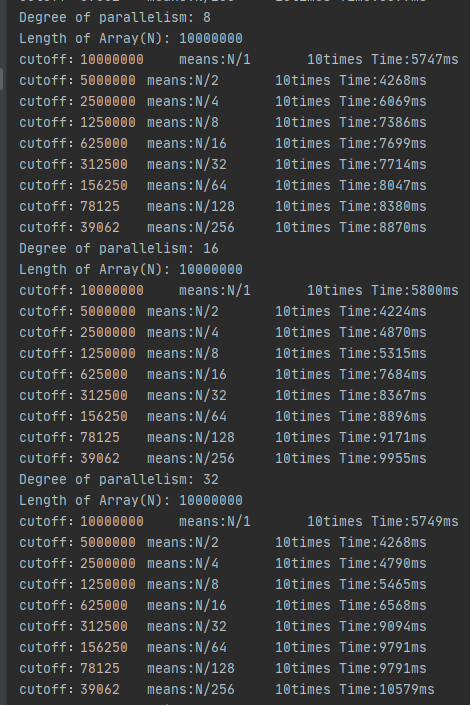
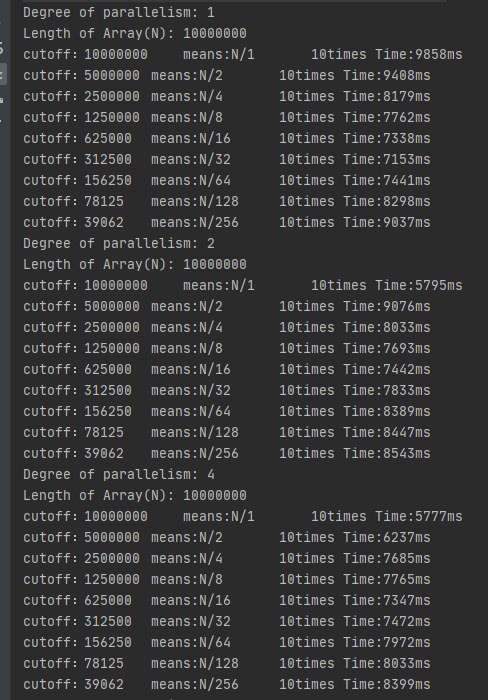
Length of Array (N):6000000



Length of Array (N):8000000



Length of Array (N):10000000



* **Relationship conclusion**

First, I think Each iteration of the Parsort will divide the length of the array(N) by 2 , So cutoff must relate to N divide by powers of 2( means cutoff=N/2^x);

Besides, the number of available threads also stick to powers of 2;

According to the results of experiment:

1.In my computer,the best number of available threads is 8.This is because when number of available threads less than 8, the time performance of Parsort worse than 8 threads;

when number of available threads bigger than 8,the best time performances are almost same with 8,but they would require more available threads.

2.The best number of cutoff should be N/2 ,under number of threads is 8 .

When N=1000000,2000000,4000000,8000000,10000000; They all reach best performance when Cutoff is N\2(N/2^x, x=1) and the number of threads not less than 8, so in order to save threads, 8 threads is the best for me.

In conclusion, I would select N/2 as the cutoff which N is the length of Array.Besides,the number of threads is 8.

* **Evidence to support relationship** (screen shot and/or graph and/or spreadsheet)

Number of Cutoff stick to N/(2^x), I set x=0,1,2,3,4,5,6,7,8.And I set x as the X-axis;

The time performances as the Y-axis;

Different threads as different lines;

Convert the output to a table as follow;

**when N=1000000,2000000,4000000,8000000,10000000; They all reach best performance when Cutoff is N\2 (N/2^x, x=1)and the number of threads not less than 8, so in order to save threads, 8 threads is best for me.**

* **Screenshot of Unit test passing**

**I can’t find unit test for this class, so I write class below to test Parsort.**

