**INFO 6205**

**Program Structures & Algorithms**

**Fall 2020**

**Assignment 5**

* **Task**

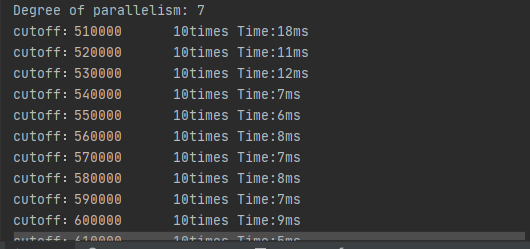
The goal of this assignment is to implement a parallel sorting algorithm derived from merge sort such that each partition of the array is sorted in parallel. The program needs to decide whether to sort in parallel or in sequential to based on efficiency. The experiment requires a suitable value for cutoff to better compare the runtime efficiency of the parallel sort.

* **Program Output**

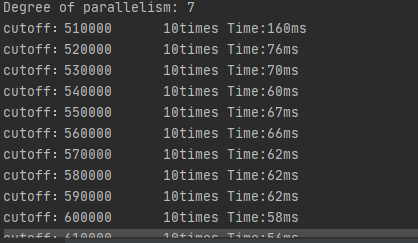
N is the size of the array

Cutoff = 1000

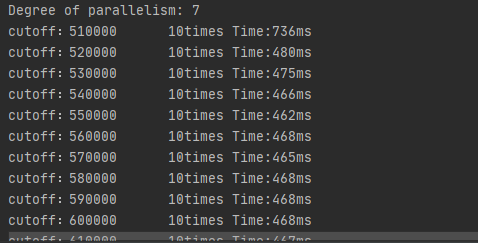
Case 1: n = 10000



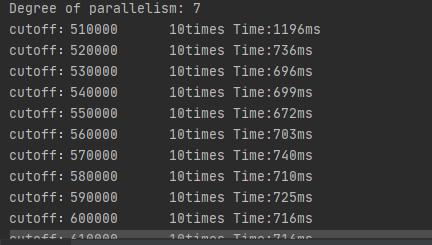
Case 2: n = 100000



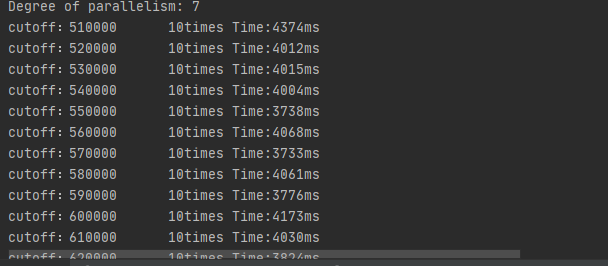
Case 3: n = 1000000



Case 4: n = 2000000

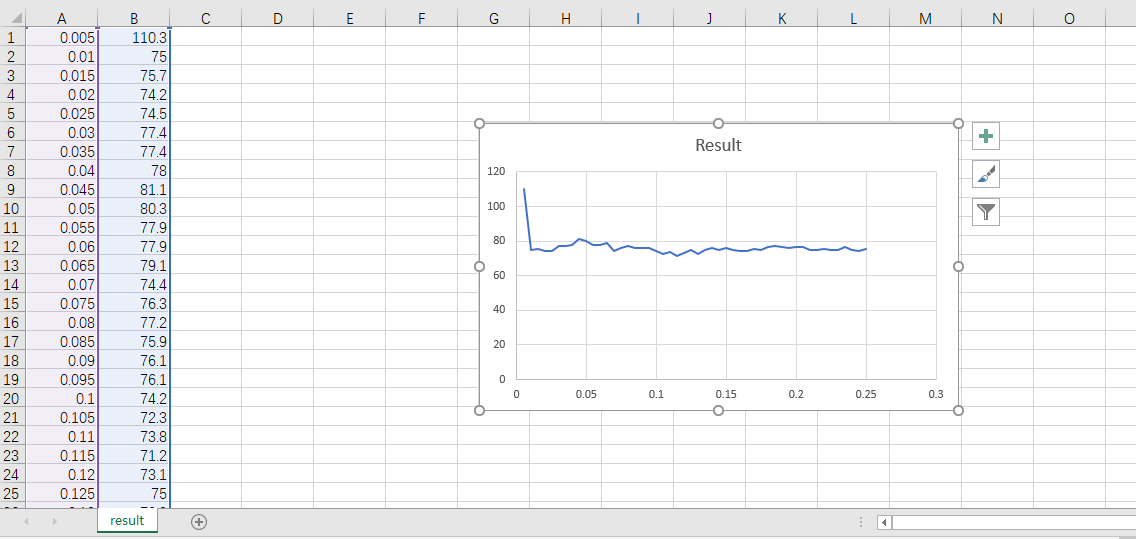


Case 5: n = 10000000



When I chose different cutoff, runtime becomes slower.

* **Evidence**

****

I use Excel to make a scatter plot which shows the relationship between cutoffs and runtime. We can observe that despite at the beginning of running the program, different cutoffs have similar processing time.

* **Conclusion**

Parallel sorting isn’t always the better way because we have a bunch of tasks consisting of simply sorting two integers which makes the parallel sorting slower than the sequential sorting. It’s way faster to have one thread performing two times a simple job rather than having two threads synchronizing each other to split and share this job. Therefore, we need to find a limit regarding the number of elements to handle. If we are above this limit, we trigger the parallel execution. If we are below, we just run the sequential method. **ForkJoinPool** is a powerful tool for implementing recursive and parallel algorithm.