Parallel and Distributed Programming
Lab 5

In this lab our task was to perform both trivial and using the Karatsuba algorithm multiplications of polynomials of arbitrary sizes. I will firstly present the main idea of the algorithms and then show a benchmark of comparative running time analysis.

Algorithms:

1. **Trivial Sequential**

This is a secondary school elementary algorithm that just iterates through each combination of coefficients and sums them up in order to obtain the result. This is what we intuitively do when we multiply polynomials by hand.

2. **Trivial Parallel**

In this I enhanced the algorithm described at 1. by parallelizing using an ExecutorService with a variable number of threads one of the loops described above. That is, I parallely did the calculations for each of the coefficients of the first polynomial and after all I just added up the results in a sequential manner.

3. **Karatsuba Sequential**

Here I implemented the divide-and-conquer algorithm developed by Karatsuba and described here: https://en.wikipedia.org/wiki/Karatsuba_algorithm. This starts from the idea that it’s faster to perform a little bit more additions than do many multiplications, and, subsequently, does the following: splits each polynomial in 3 parts, the first half, the second half and one overlapping part, that sums up the 2 mentioned before. If we are in the base case when both polynomials are of degree one, do the trivial computation. Else, do recursively down the tree the same operation and then, having obtained the results, we combine them in the following manner: shift the high part to its place by adding the needed number of zeroes, remove what is above the correct result (the overlapping part) and then add the low part.

4. **Karatsuba Parallel**

We parallelize the algorithm described at 3. by using an ExecutorService for the recursion, so that each is done parallely. The catch is that we need a new Executor for each iteration since the threads go one way and the recursion in the other. In order to buy efficiency, when the polynomials are small enough, we go to sequential again, and also when the depth is over a given threshold. The rest of the algorithm is the same.

Benchmark:

Algorithm	Size	Time
Trivial Sequential	10	0.0019s
Trivial Sequential	50	0.0047s
Trivial Sequential	100	0.009s
Trivial Parallel	10	0.24s
Trivial Parallel	50	0.22s
Trivial Parallel	100	0.03s
Karatsuba Sequential	10	0.01s
Karatsuba Sequential	50	0.03s
Karatsuba Sequential	100	0.049s
Karatsuba Parallel	10	0.067s
Karatsuba Parallel	50	0.91s
Karatsuba Parallel	100	1.98s