sameera Oblig 3 report

Introduction:

This report is for oblig 3 “Prime Numbers” in IN3030 2024.
I have tested my code on a laptop with an AMD Ryzen™ 7 4700U-prosessor 2.00 GHz with 8 cores.

User guide:

The program takes two parameters, one is required, and one is optional.

1. The required parameter n, that is used to find the all the prime numbers less than n. n is also used to find the factors for the 100 numbers less than n^2. n must be larger than 16.
2. The second parameter k is the number of threads that the program will use. If k is 0 or left empty, the program uses the number of cores on the machine.

Example:

java Oblig3 n

Here the program uses the number of cores on the machine. The program executes all 4 methods for Sieve of Eratosthenes and factorization, and reports the time taken in milliseconds.

Parallel Sieve of Eratosthenes:

First, I sequentially find primes up to the square root of n and store them in an array. This step is important for the parallel phase, where the task of marking numbers is distributed across multiple threads. Each thread is responsible for a specific segment of the oddNumbers array, working independently to mark non-primes. To ensure all threads have completed their marking before proceeding, a CyclicBarrier is employed for synchronization. After synchronization, the algorithm compiles the list of primes by scanning the oddNumbers array for positions that remain unmarked, indicating prime numbers.

Parallel factorization of a large number:

I divide the list of prime numbers among multiple threads, with each thread independently working on its assigned part of primes to identify and record the factors of a large number. To ensure all threads have completed their factorization tasks, I employ a CyclicBarrier for synchronization. Once all threads reach this synchronization point, the algorithm merges the factors identified by each thread. If any part of the number remains after this process, it means that there is a prime factor larger than those in the provided list, which is then added to the final list of factors.

Implementation:

The program takes user input for the number n and the thread count k, it uses all available cores if k is empty or zero. The program executes the Sieve of Eratosthenes in both sequential and parallel, timing each run and testing resulting prime lists are equal. It then times the factorization of the 100 largest numbers below n^2, by using the prime list from the Sieve of Eratosthenes. This process is repeated seven times and the output is median time, to get an accurate performance for both the sequential and parallel methods.

Measurements:

All times are in ms

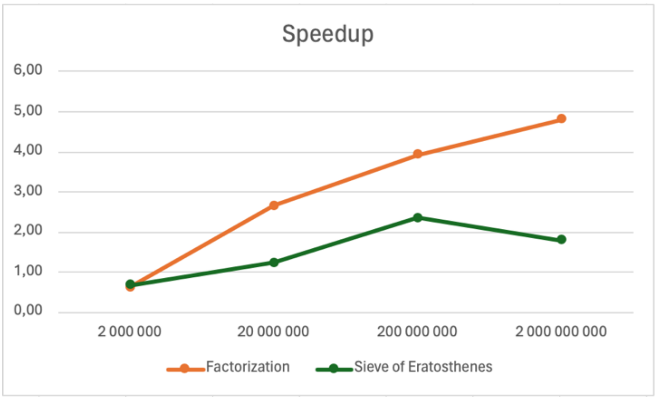
Sieve of Eratosthenes			
n	Sequentia	Parallel	Speedup
2 000 000	12	18	0,67
20 000 000	117	95	1,23
200 000 000	1237	525	2,36
2 000 000 000	14953	8316	1,80

As we can see the parallel implementation is slower when n is at 2 million. But As n increases, parallel processing gives more advantage, outperforming sequential with speedups of 1.23, 2.36, and 1.80. The most significant speedup occurs at n=200 million, rather than n=2 billion, potentially due to the increased overhead at larger scales.

Factorization			
n	Sequentia	Parallel	Speedup
2 000 000	113	181	0,62
20 000 000	966	363	2,66
200 000 000	5102	1300	3,92
2 000 000 000	46400	9671	4,80

Like the Sieve of Eratosthenes, the sequential factorization is faster when n is at 2 million, as indicated by a speedup of 0.62. However, as n increases, the parallel method gains an advantage, achieving speedups of 2.66, 3.92, and 4.80. This shows the parallel factorization method's escalating efficiency with larger numbers, with its best speedup at n=2 billion.

Graph showing speedup:



The speedup from the factorization is much bigger than the speedup gotten from the Sieve of Eratosthenes. While the Sieve of Eratosthenes shows varying speedup, the factorization speedup consistently increases with the value of n.

Conclusion:

Both the parallel versions of Sieve of Eratosthenes and factorization shows a good speed up when n is big. But the speedup of the factorization is better and is not varying, as compared to the Sieve of Eratosthenes.