Assignment 7

Inderpreet Singh Chera - 160101035

2 Polynomial Multiplication:

This multiplication of polynomials is carried out by 3 methods:

1. School Method
2. Karatsuba's Algorithm
3. FFT divide and conquer

System Settings

All tests were done on **Intel(R) Core(TM) i3-5005U CPU @ 2.00GHz** processor. This computer is **dual core** where each core has **2 threads**. Also during each experiment it was insured that **no other applications** were running in the background, so that we don't have any biased readings.

Implementations

We have 2 polynomials, say a and b with n-1 max power, i.e., the total number of elements in each polynomial is n.

School Method

This is a naive way of multiplying 2 polynomials. We multiply each element of b with all elements of a and get the output. This takes O(n) time complexity.

To do the same in parallel and to distribute almost the same amount of work to each processor, we tell each processor to get the answer of each mod p, which has value equal to processor id, coefficient of final aray. If p divides n, then it can be shown that the same amount of work is done by all the threads.

So, say we have 4 processors, then to get the coefficients of resultant array, processors will be allocated as $P_0P_1P_2P_3P_0P_1$... and so on.

Karatsuba's Algorithm

This is a divide and conquer algorithm. So, here doing completely parallel work, we will require to have exponential number of threads($O(3^n)$), as to get the result of each subpart we have to make a new thread. But, the system didn't allow to make exponential number of threads.

So, partial parallel algorithm is implemented. Upto a certain level, all the work is distributed parallely and then serial work is done. By this, if level is 0, then 1 thread is used. If the level is 1, then 4 threads. If it is 2, then 13 threads, and so on. Then after the certain max level all the work is done serially using the karatsuba algorithm.

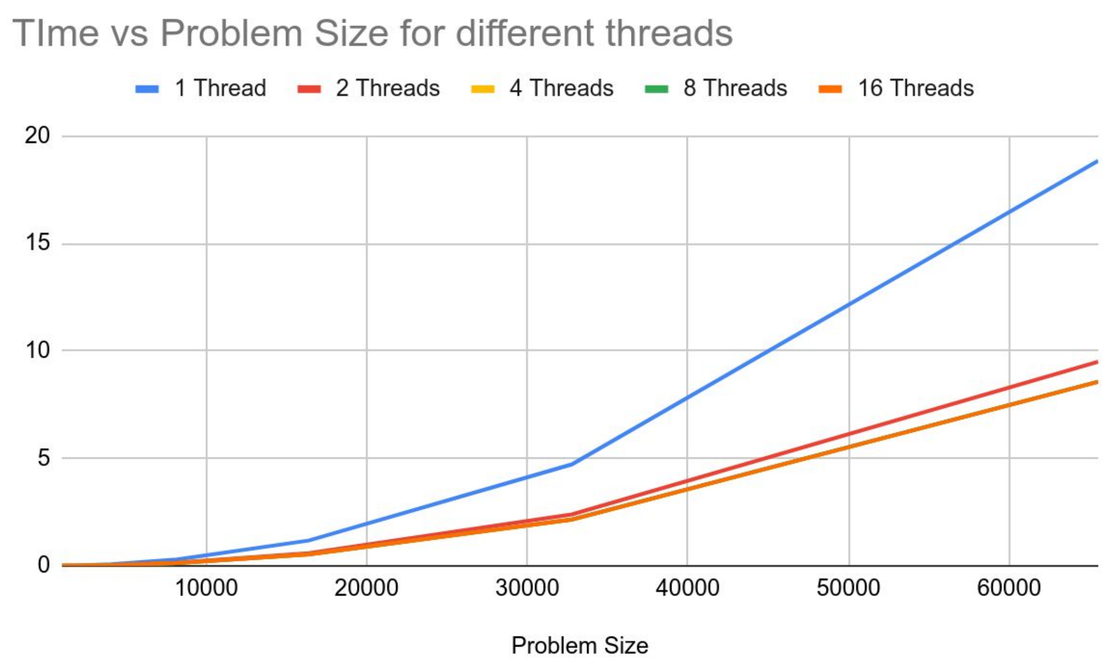
FFT Divide and Conquer(to be done)

Observations

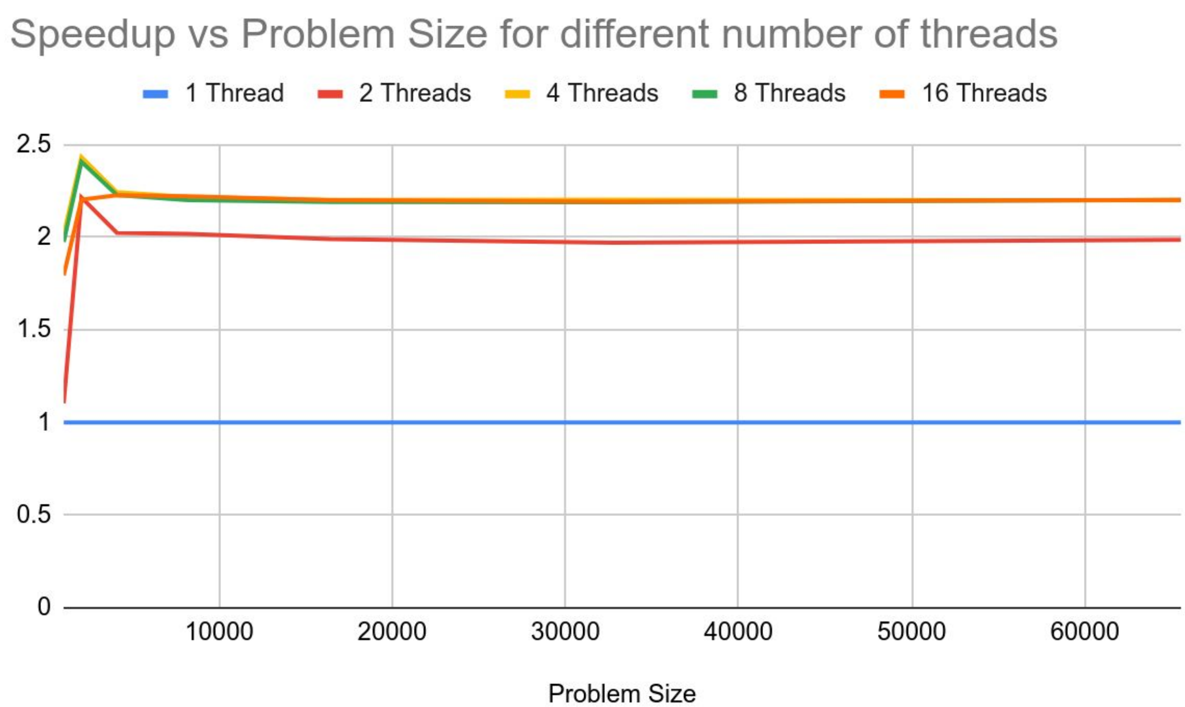
School Method:

1. **TIme (and speedup) vs Size for different number of threads:**

Problem Size	1 Thread	2 Threads	4 Threads	8 Threads	16 Threads
1024	0.00467371	0.0042328	0.00231602	0.00236647	0.00260466
2048	0.0216977	0.00979527	0.00891673	0.00901242	0.00984955
4096	0.075046	0.0370683	0.0334506	0.0336668	0.0336938
8192	0.297658	0.14748	0.134205	0.135282	0.133997
16384	1.17742	0.591356	0.534705	0.537322	0.53486
32768	4.7213	2.39507	2.14306	2.15579	2.15348
65536	18.8549	9.49411	8.57282	8.56366	8.56261



Problem Size	1 Thread	2 Threads	4 Threads	8 Threads	16 Threads
1024	1	1.104165092	2.017992073	1.97497116	1.794364716
2048	1	2.215120155	2.433369632	2.40753316	2.202912823
4096	1	2.024533092	2.243487411	2.229080281	2.227294042
8192	1	2.018294006	2.217935248	2.200277938	2.221378091
16384	1	1.991051076	2.201999233	2.191274506	2.201361104
32768	1	1.971257625	2.203064777	2.190055618	2.192404852
65536	1	1.98595761	2.1993813	2.201733838	2.202003828



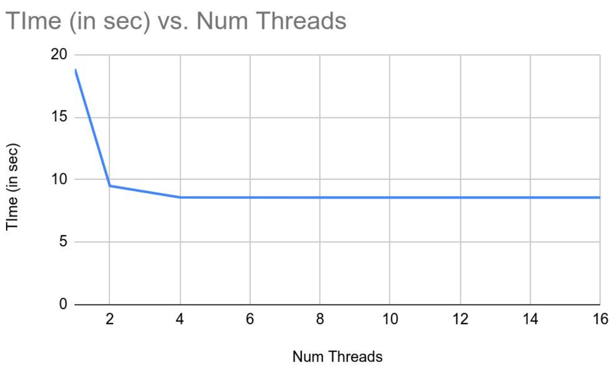
Observations:

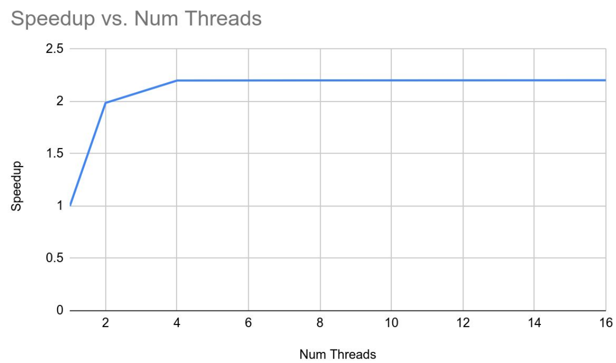
Time taken by a single threaded program is more than a multi threaded program. Time taken reduces till 4 threads, and then remains almost constant because my pc can handle a maximum of 4 threads that can run parallel.

Speedup is observed maximum for problem size 2048 and after that it reduces and remains almost constant.

2. **Time(and speedup) vs number of threads for problem size = 2^16.**

Num Threads	TIme (in sec)	Num Threads	Speedup
1	18.8549	1	1
2	9.49411	2	1.98595761
4	8.57282	4	2.1993813
8	8.56366	8	2.201733838
16	8.56261	16	2.202003828





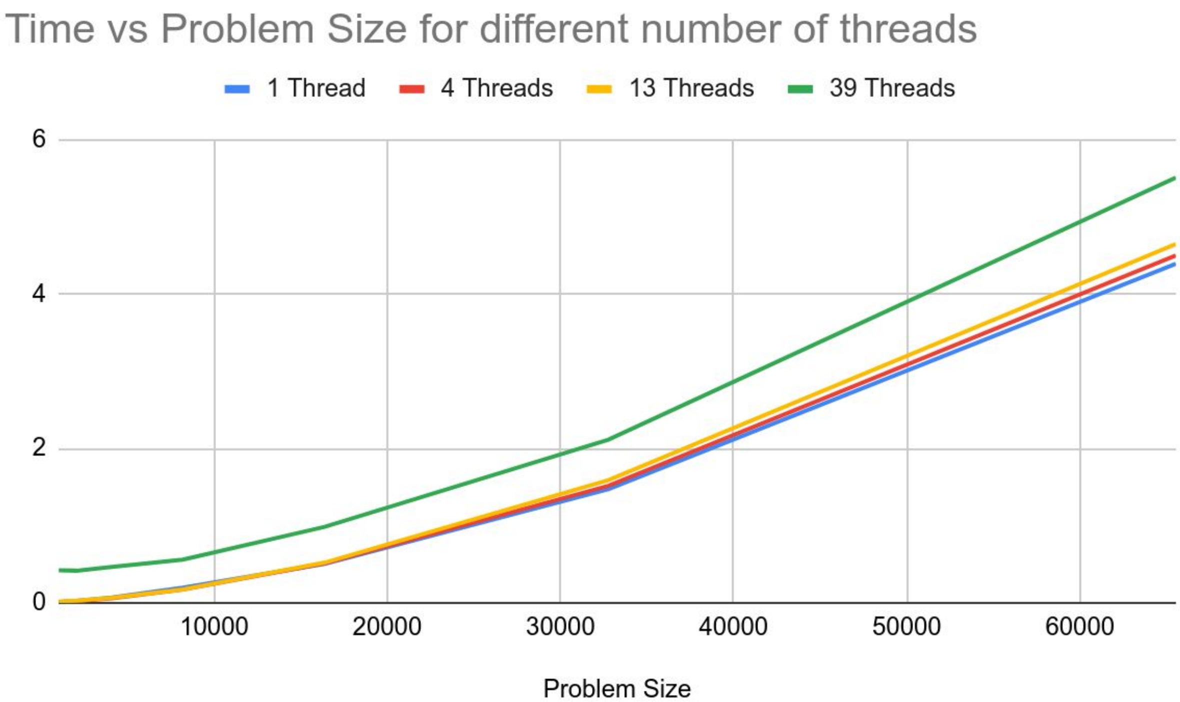
Observations:

We observe that time decreases upto 4 threads and then remain constant. Similarly, speedup increases till 4 threads and then remains constant.

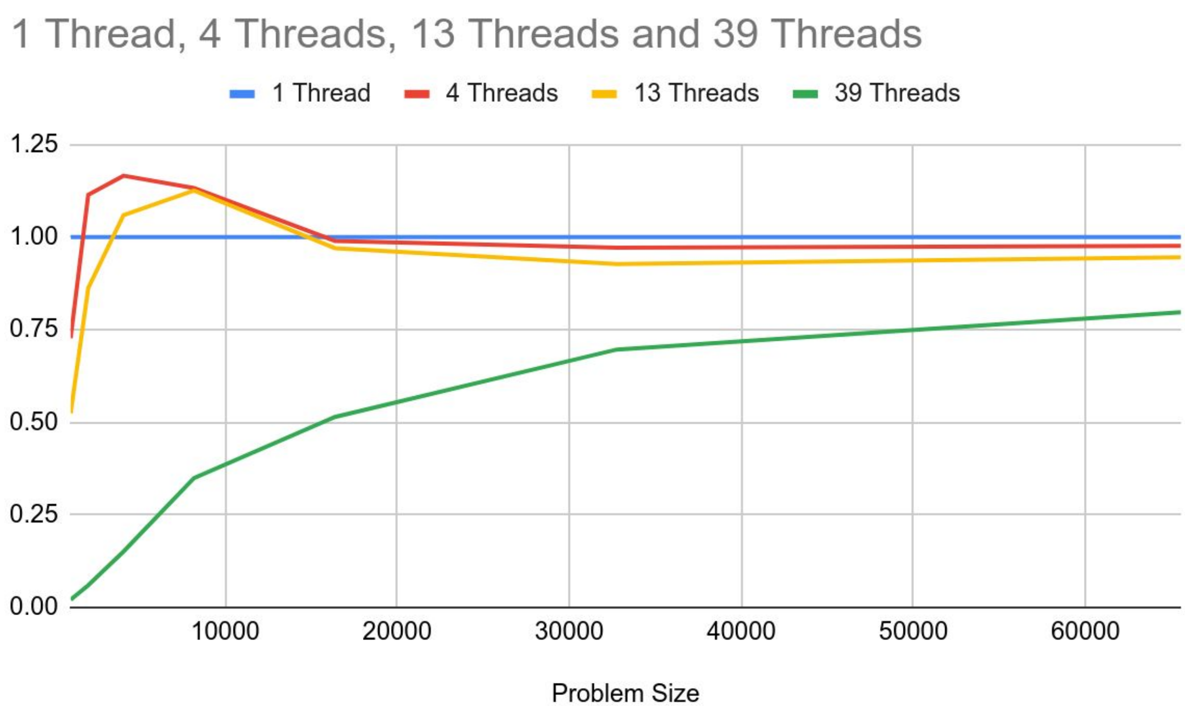
Karatsuba Algorithm:

1. **TIme (and speedup) vs Size for different number of threads:**

Problem Size	1 Thread	4 Threads	13 Threads	39 Threads
1024	0.00869528	0.011958	0.0165868	0.420957
2048	0.0250423	0.0224731	0.0290358	0.417112
4096	0.0703828	0.0603751	0.0664368	0.466027
8192	0.195823	0.172887	0.173897	0.560788
16384	0.506459	0.511564	0.522004	0.984732
32768	1.47302	1.51661	1.58884	2.1143
65536	4.39404	4.49971	4.64765	5.50956



Problem Size	1 Thread	4 Threads	13 Threads	39 Threads
1024	1	0.7271516976	0.5242289049	0.02065598149
2048	1	1.114323347	0.8624628906	0.06003735208
4096	1	1.165758732	1.059394793	0.151027301
8192	1	1.132664688	1.126086131	0.3491925647
16384	1	0.990020799	0.9702205347	0.5143115081
32768	1	0.9712582668	0.9271040508	0.6966939413
65536	1	0.9765162644	0.945432638	0.7975301113



Observations:

Speedup is not coming greater than 1, except for when we are having 4 threads. It is possible as major work as the size of the problem increases is serial.