GIT Department of Computer Engineering
CSE 344 – Spring 2022

Homework 5 #Report

Okan Torun
1801042662

How did I handle the problem?

First of all, I read the data in the two input files into two different arrays as much as the desired number of characters. If the file content is missing from the desired number, I ended the program by giving an error. Then I created the desired number of threads. I kept the information of the threads in a struct and sent them into the function they are running.First, I calculated the product of two square matrices and recorded the result in the C matrix. I did this by dividing the number of threads I created. Then, I made the **pthread_cond_wait** command wait for the finished matrices. After all the matrices came, I proceeded to the second operation.Here, by using the desired DFT formula, I ensured that all matrices do equal work. I finished this process by calculating the same number of rows of each matrix DFT matrix. In the meantime, I made the desired print operations and calculated the elapsed times. After calculating the DFT matrix, I printed the DFT result to the file with the main process in the main function.

Matrix Multiplication

I used all the cells of the 1st matrix I had in matrix multiplication. In the other matrix, I calculated the number of threads and the number of columns in the matrix and how many columns it should deal with. And I sent the start column (start) and end column values for each matrix (end) as parameters.

```
for(int i = start ; i < end; i++){
    for(int j = 0 ; j < matrixEdge; j++){
        for(int k = 0 ; k < matrixEdge; k++){
            result += fileValues1[j][k] * fileValues2[k][i];
        }
        matrixC[j][i] = result;
        result = 0;
    }
}
```

Threads waiting for each other

All incoming threads are put on hold by **pthread_cond_wait** operation, respectively. The last incoming thread releases all threads with **pthread_cond_broadcast** and **pthread_mutex_unlock** operations, and they all start the second operation at the same time.

```
pthread_mutex_lock(&mutex);
++arrived;
while(arrived < mValue)
{
    pthread_cond_wait(&cond, &mutex);
}
pthread_cond_broadcast(&cond);
pthread_mutex_unlock(&mutex);
```

DFT Calculation

I made 4 nested for loops for the DFT calculation. For the first two, I shared the task of the subjects. For the other two, I went through all the cells of the C matrix and calculated a single cell of the DFT matrix. I created two different DFT arrays for this. I saved the imaginer value in one of them and the real value in the other.

```
for(int k=start;k<end;k++)
{
    for(int l=0;l<totalCell;l++)
    {
        double reTemp=0;
        double imTemp=0;
        for(int m=0;m<totalCell;m++)
        {
            for(int n=0;n<totalCell;n++)
            {

                double x=-2.0*PI*k*m/(double)totalCell;
                double y=-2.0*PI*l*n/(double)totalCell;
                reTemp+=matrixC[m][n]*cos(x+y);
                imTemp+=matrixC[m][n]*1.0*sin(x+y);
            }
        }
        DFTRe[k][l]=reTemp;
        DFTIm[k][l]=imTemp;
    }
}
```

Signal Handling

Here I closed all open files and freed all allocated resources.

Thread Number

If the number of threads is more than the number of columns, I create as many threads as the number of columns. If the number of columns is not divided by the number of threads, I rounded the number of columns to the number of threads to a lower number divided by the number of threads. Each thread did as much as the number of rounded columns, and the last thread completed the remaining number of columns.

Test Cases

Errors:

```
okan@okan-ABRA-A5-V16-4:~/Desktop$ ./hw5 -i inputFile1 -j inputFile2 -o output.csv -n 5 -m 8
Your inputfile1 does not have enough characters !!!
```

```
okan@okan-ABRA-A5-V16-4:~/Desktop$ ./hw5 -i inputFile1 -j inputFile2 -o output.csv -n 5 -m 0
The value of m must be greater than 1.
```

```
okan@okan-ABRA-A5-V16-4:~/Desktop$ ./hw5 -i inputFile1 -j inputFile2 -o output.csv -n 1 -m 8
The value of n must be greater than 2.
```

```
okan@okan-ABRA-A5-V16-4:~/Desktop$ ./hw5 -i file1.txt -j file2.txt -o output.csv -n 3 -m 4
Sun May 22 23:08:20 2022 Two matrices of size 8x8 have been read. The number of threads is 4
Sun May 22 23:08:20 2022 Thread 0 has reached the rendezvous point in 0.000006 seconds.
Sun May 22 23:08:20 2022 Thread 1 has reached the rendezvous point in 0.000008 seconds.
Sun May 22 23:08:20 2022 Thread 2 has reached the rendezvous point in 0.000007 seconds.
Sun May 22 23:08:20 2022 Thread 3 has reached the rendezvous point in 0.000185 seconds.
Sun May 22 23:08:20 2022 Thread 3 is advancing to the second part
Sun May 22 23:08:20 2022 Thread 0 is advancing to the second part
Sun May 22 23:08:20 2022 Thread 2 is advancing to the second part
Sun May 22 23:08:20 2022 Thread 1 is advancing to the second part
Sun May 22 23:08:20 2022 Thread 0 has has finished the second part in 0.000459 seconds.
Sun May 22 23:08:20 2022 Thread 2 has has finished the second part in 0.000876 seconds.
Sun May 22 23:08:20 2022 Thread 3 has has finished the second part in 0.000912 seconds.
Sun May 22 23:08:20 2022 Thread 1 has has finished the second part in 0.001002 seconds.
Sun May 22 23:08:20 2022 The process has written the output file. The total time spent is 0.003754 seconds.
```

1	4105035.000000 + j(0.000000)	125759.052855 + j(16662.423463)	-250793.000000 + j(15128.000000)	-32333.052855 + j(-149745.576537)	71323.000000 + j(-0.000000)	-32333.052855 + j(149745.576537)	-250793.000000 + j(-15128.000000)	125759.052855 + j(-16662.423463)
2	33485.071673 + j(-402189.901045)	3693.440902 + j(-26856.247456)	-1894.188309 + j(13964.730076)	-10609.121425 + j(-717.922656)	3241.670991 + j(1331.277995)	13322.640031 + j(9583.987448)	5009.433760 + j(10694.583038)	3755.110606 + j(-13715.656260)
3	149771.000000 + j(27778.000000)	20003.371840 + j(6061.671140)	-8557.000000 + j(2542.000000)	952.581961 + j(-2711.662474)	-1269.000000 + j(5562.000000)	-13581.371840 + j(5490.328860)	-3573.000000 + j(1814.000000)	-874.581961 + j(9399.662474)
4	-34267.071673 + j(-98421.901045)	-5848.878575 + j(2166.077344)	-4867.433760 + j(-509.416962)	-6463.440902 + j(-2644.247456)	-4055.670991 + j(-2708.722005)	4170.889394 + j(-13199.656260)	1924.188309 + j(2776.730076)	-5828.640031 + j(-12452.012552)
5	175707.000000 + j(-0.000000)	3528.867966 + j(2486.156421)	-6849.000000 + j(-6816.000000)	3953.132034 + j(-1897.843579)	-6293.000000 + j(-0.000000)	3953.132034 + j(1897.843579)	-6849.000000 + j(6816.000000)	3528.867966 + j(-2486.156421)
6	-34267.071673 + j(98421.901045)	-5828.640031 + j(12452.012552)	1924.188309 + j(-2776.730076)	4170.889394 + j(13199.656260)	-4055.670991 + j(2708.722005)	-6463.440902 + j(2644.247456)	-4867.433760 + j(509.416962)	-5848.878575 + j(-2166.077344)
7	149771.000000 + j(-27778.000000)	-874.581961 + j(-9399.662474)	-3573.000000 + j(-1814.000000)	-13581.371840 + j(-5490.328860)	-1269.000000 + j(-5562.000000)	952.581961 + j(2711.662474)	-8557.000000 + j(-2542.000000)	20003.371840 + j(-6061.671140)
8	33485.071673 + j(402189.901045)	3755.110606 + j(13715.656260)	5009.433760 + j(-10694.583038)	13322.640031 + j(-9583.987448)	3241.670991 + j(-1331.277995)	-10609.121425 + j(717.922656)	-1894.188309 + j(-13964.730076)	3693.440902 + j(26856.247456)