ICS 353
Programming Assignment Report: Strassen's Algorithm

by

Group 16

Mujtaba Al-Mohsin

201480180

Abdulaziz Al-Amer

201235200

Prepared for

Dr. Wasfi Al-Khatib

(60 points) Write a report (word document or pdf-generated file from an editor) that contains the following information

a. (5 points) How to compile and run the code, with any implementation details worth of mentioning.

1- Run the program, then You will be asked to enter N

of the size of the matrix $2^n \times 2^n$.

```
Enter a value of n where your matrices are of size 2^n X 2^n: 8
```

2- After that, enter the path of the file text that you want to test.

```
Enter a value of n where your matrices are of size 2^n X 2^n: 8

Enter input file path:
C:\Users\Muj\Desktop\matrix_08-13\matrix_08.txt
```

3- Choose the number of the algorithm to run.

```
Enter a value of n where your matrices are of size 2^n X 2^n: 8

Enter input file path:
C:\Users\Muj\Desktop\matrix_08-13\matrix_08.txt

Enter multiplication algorithm number:
1- Iterative
2- Divide and conquer recursive
3- Strassen’s algorithm  with base case of n = 1
4- Strassen's  algorithm  with base case of n > 1
3
```

4- Wait until the program finish.

....

5- After it finished, it would note you.

```
Enter a value of n where your matrices are of size 2^n X 2^n: 8

Enter input file path:
C:\Users\Muj\Desktop\matrix_08-13\matrix_08.txt

Enter multiplication algorithm number:
1- Iterative
2- Divide and conquer recursive
3- Strassen’s algorithm  with base case of n = 1
4- Strassen's  algorithm  with base case of n > 1
3

Done result matrix stored in output.txt
```

6- The file output will be now in the program folder.

.settings	٤٠/٠٤/٠١ ص ٠٩:٠١	File folder	
bin	٤٠/٠٤/٠٢ م ٠٧:٠٧	File folder	
src	٤٠/٠٤/٠١ ص ٠٩:٠٣	File folder	
.classpath	٤٠/٠٤/٠١ ص ٠٩:٠١	CLASSPATH File	1 KB
.project	٤٠/٠٤/٠١ ص ٠٩:٠١	PROJECT File	1 KB
output	٤٠/٠٤/٠٣ م ٠٣:٣٢	Text Document	410 KB

7- Now, you can see the result of the multiplication and the elapsed time in seconds.

output - Notepad

File Edit Format View Help

```
Elapsed Time in seconds : 3.184

68518   -34483  -78049  19079   -39157  -9595   -8932   16455   -24287  -25831
-12571  15655   -36205  34173   -9471   46536   8362    -17567  64330   -35585
140992  46910   -46460  -27114  53988   106548  -23688  -14910  -12788  -30827
-45000  -24039  54615   5737    8855    21547   -79650  -21289  -82897  -28012
56884   -11576  -39295  12940   -45373  71497   -60943  -20828  3208    -19363
57994   -4914   2371    20619   28686   23112   88280   -109211 43986   -144737
-78321  47324   29951   458     23145   -14577  -30641  78321   36706   67596
3014    -23060  -29936  28100   -82027  30841   -6003   -78332  -4708   121516
-34489  -31319  18286   41417   2760    -153251 38034   -19459  -96511  -21052
-17521  7020    -131271 80414   28284   23715   -85176  -28809  -49214  39220
-12000  -15844  122716  -57171  -2507   -22579  -72769  -15720  -4425   49153
41769   8437    55615   13839   -55133  3532    -91289  -27678  -65759  -44465
-23605  -26878  19752   -29043  61812   -98674  -46330  36822   -101595 -42414
-121440 17857   -129943 -106342 64051   13138   -6608   7688    36357   -26328
51722   -23623  112958  -56967  -93639  32506   68307   -20039  -48537  -19447
-15916  27783   -44816  13041   26288   -58138  105043  39675   -27409  -66351
-97147  7185    -50938  24087   -44389  -94052  -55679  -12977  35022   47263
512     75950   24622   -68067  -96382  -27198  -11594  10980   -27699  -23658
-24464  35071   18844   -94082  -7635   -52595  -17759  80633   -38235  62629
```

b. (15 points) Documentation of all experiments carried out in the form of a comparative table.

		N = 6	N = 8	N = 9	N = 10	N = 11	N = 12	N = 13	N = 14
1	Iterative	0.004 s	0.07 s	0.28 s	8 s	80 s	714 s		
2	Divide and conquer recursive	0.175 s	5.5 s	34 s	336 s	2303 s	>13,890.8 s		
3	Strassen's algorithm with base case of n = 1	0.104 s	3 s	22 s	197 s	1351 s	6572.25		
4	Strassen's algorithm with base case of n > 1	Base = 3 0.001 s	Base = 4 0.11 s	Base = 5 0.23 s	Base = 5 1.48 s	Base = 6 11 s	Base = 6 60.39 s	Base = 4 495.28 s	
		Base = 2 0.014 s	Base = 2 0.353 s	Base = 7 0.26 s	Base = 3 3.23 s	Base = 3 22 s	Base = 3 119.79 s	Base = 5 419.25 s	
		Base = 4 0.008 s	Base = 6 0.102 s	Base = 10 0.31 s	Base = 8 3.10 s	Base = 11 81 s	Base = 9 113.08 s	Base = 6 429.59 s	

**For n=12 and n=13 findings we used different machine, and not all test conducted are in the table.*

c. (25 points) Analysis of the results present in the table. If there are any unexpected results, please highlight them and give possible justification for them.

- At first, we expected from our online research that the iterative and divide and conquer implementation have similar time complexities, but after conducting our own implementations and testing as seen in the above table we noticed that the iterative version is much faster than divide and conquer and even Strassen's with base equal to one, which is slightly faster than divide and conquer. So, we concluded that the iterative version has complexity of Big-O(n^3), where the recurrence relation in divide and conquer, Strassen's base equal one result in complexities of Big-theta(n^3), Big-theta(n^2.8) respectively, which justified our findings that we reordered in the above table.

d. (10 points) Analysis of the results for different base values for Strassen's algorithm.

- We ran many different bases of Strassen's with base values of n > 1 on different sizes of matrices' and we reached a conclusion that the best fastest ones were where the base is close to (n/2) or ([n-1]/2) for multiplying two matrices of size (2^n)X(2^n). For example best base for multiplying matrixes of size (2^12)X(2^12) is base 5, and the higher your base value is the closer your result is going to be to the classical Strassen's with base 1 time, and the smaller the base that you use the closer your result is going to be to the classical iterative version result time.

e. (5 points) Your conclusions of when to use each algorithm and the best Strassen's base case.

- We think that for not very big matrices like (2^8)X(2^8) and smaller the iterative approach would be just fine since there is no significant or noticeable difference between it and Strassen's with a suitable base value and in iterative no need for extra base value input and it is much less complicated to implement. However for very big matrices the best approach is to use Strassen's with base value of (n/2) or ([n-1]/2) for multiplying two matrices of size (2^n)X(2^n), and in general the classic divide and conquer recursive approach should be avoided since it always preformed the worst for many different sizes of matrices that we tested.

f. Who did what and the approximate percentage of total work done in this assignment for each member?

- (1) Mujtaba Al-Mohsin did (50%) :

- The classical iterative version of the matrix multiplication algorithm.

- Read input file and write to output file.

- Reviewing codes.

- Some experimentations.

- Writing the report.

- (2) Abdulaziz Al-Amer did (50%) :

- The classical divide and conquer recursive version.

- The classical Strassen’s divide and conquer recursive algorithm, n=1.

- Strassen’s divide and conquer recursive algorithm, n>1.

- Some experimentations.

- Reviewing the report.