ANALYSIS:

All the details regarding the timing of the program is given below. Here I am including the summary of my analysis.

Strassen's Algorithm for matrix multiplication seems to be a very non-trivial algorithm that isn't obvious to one's understanding at all. I implemented Strassen's algorithm in 2 ways. The source code Strassensmod.py contains Strassen's Algorithm implementation with a leaf size of 1 i.e the matrices are directly multiplied using the normal method and returned if the matrix size is 1x1. The other implementation, Strassens.py, is when leaf-size=2 i.e the lowermost size of matrix dimensions for them to be multiplied directly in a normal manner and returned is 2x2. In the first case, the elements are of the form int when their size is 1x1(instead of a list) and that's why there had to be many special conditions incorporated into the code to counter that issue as opposed to the latter code. That's the reason I implemented 2 codes.
The third code is normalMatMulti.py which implements matrix multiplication in normal manner. I recorded the time usage of each code using cProfile module of Python and the results are attached below.

Testcases: I used many test cases, but I am including the 2 main ones as below. The file LabStrassenInput.txt includes the required input that was given to us along with the assignment, which has 2x2, 4x4, 8x8 matrices to be multiplied. The other testcase input that I made testcase2.txt has 2x2,4x4,8x8 and 16x16 matrices that needs to be multiplied. I added 16x16 in this and kept the rest of the others same so that the outputs can be compared across the programs and testcases for the correctness of the solution. The summary of the details is as below:

Test case file	Strassensmod.py	Strassens.py	normalMatMulti.py
LabStrassenInput.txt	Calls made: 3226 Exec Time: 0.026 sec	Calls made: 896 Exec Time: 0.017 sec	Calls made: 147 Exec Time: 0.030sec
	BETTER	BEST	GOOD
testcase2.txt	Calls made: 22083 Exec Time: 0.061 sec	Calls made: 5851 Exec Time: 0.022sec	Calls made: 281 Exec Time: 0.004sec
	GOOD	BETTER	BEST

Seeing the above summary statistics, for the normal required input, we can see that the normal multiplication took more time that the Strassen's implementations, which was what was expected. Between the 2 Strassen's implementation, it looks as if setting the leaf-size to 2 had a better effect on the overall timing of the program execution than setting it at 1. I suppose the tradeoff of breaking down the matrices into smaller ones and increasing the number of function calls increased the execution time. Stopping at 2x2 matrix and not breaking it down further reduced the overall execution time by reducing the number of function calls. It is a good point to think as to when to stop dividing the matrices into n/2 matrices, which is proven by my testcase. In testcase2.txt, we can see that the best/least time was taken by normalMatMulti.py and the worse was by Strassensmod.py, which initially would be counterintuitive. But as I mentioned before, deciding the point where to stop dividing n into n/2 and thereby reducing the number of function calls and hence the execution time, is something that needs to analysed. But doing it at 2x2 was also not appropriate, since the overall exec time was higher. Maybe stopping it at 16x16 would have made a difference, which is what I would like to change later and analyze further.

When it comes to the question of storage, Strassensmod.py and Strassens.py took more space than the normalMatMulti.py because of the initialization of many submatrices and temporary variables for algorithm's implementation. In general there were atleast S1..S10 and P1..P7 matrices for each n and n/2 matrices of both A and B input matrices. This was a significantly more storage overhead as opposed to normalMatMultiply.py, which didn't have any but to store C as the final output. So it can be intuitively concluded that as the size of the input grows, the storage overhead for Strassen's implementation would grow significantly as opposed to normalMatMulti.py. Thus, we need to keep the resources at hand in mind and then decide between the tradeoff

of speed and storage vs the input size.
I would like to run these codes for larger size input matrices and for different leaf-sizes, like we get in the field of Bioinformatics. When we go to understand and implement neural networks and like areas, we feel the need of performing matrix operations with the least amount of time possible. This would be where Strassen's T(n) of n^(lg7) would probably make more difference. But at the level of present inputs that we had, going the normal matrix multiplication way makes more sense in terms of time and space cost analysis. Even though in the first testcase, the time for Strassen's was better, the storage overhead was a lot more than the normal matrix multiplication and the execution time was not much different. So based on present testcases, if the matrix size is small, to maybe medium, doing normal matrix multiplication or doing Strassen's with a larger leaf-size would be more efficient.

Code Details:
My code has different functions for different operations to be performed and has recursion in the function Strassen, which divides the matrices into n/2 if the n>leaf-size and implements the Strassen's Algorithm with recursive calls to the function upon computation of P1…P7.
I have used lists in python3 for storing my matrices and its operations. The input file format and reading is the same way as described in the assignment details. The output are the two matrices A and B along with C being the multiplied matrix output.

```
SHUBHIs-MacBook-Pro:Module 2 Sudhir$ python3 -m cProfile Strassensmod.py
C =
 [[16, 17], [26, 22]]
C =
 [[-1, -4, -2, 13], [7, -6, 2, 6], [11, 5, -1, 1], [-6, 9, -8, 3]]
C =
 [[-4, 4, 6, 3, -4, -3, -6, -10], [7, 4, 0, -5, 11, 0, 0, 8], [6, -11, 0, -8, -4, 4, 8, 5], [-4, 11, 7, 1, -6, 1, -4, -3], [-7, 2, -
2, 8, 6, -5, -4, 6], [9, -1, 10, -3, 11, 4, 5, 10], [18, 9, 13, 12, 5, 3, 0, -2], [18, -5, -4, -11, -3, 1, 3, 3]]
         3226 function calls (2764 primitive calls) in 0.026 seconds

   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.027    0.027 Strassensmod.py:1(<module>)
       40    0.000    0.000    0.000    0.000 Strassensmod.py:120(<listcomp>)
       57    0.000    0.000    0.000    0.000 Strassensmod.py:139(<listcomp>)
      441    0.002    0.000    0.003    0.000 Strassensmod.py:151(addMat)
      441    0.001    0.000    0.001    0.000 Strassensmod.py:152(<listcomp>)
      291    0.001    0.000    0.002    0.000 Strassensmod.py:158(subMat)
      291    0.001    0.000    0.001    0.000 Strassensmod.py:159(<listcomp>)
      399    0.000    0.000    0.000    0.000 Strassensmod.py:174(matMultiply)
       14    0.000    0.000    0.000    0.000 Strassensmod.py:24(<listcomp>)
       14    0.000    0.000    0.000    0.000 Strassensmod.py:26(<listcomp>)
        1    0.000    0.000    0.027    0.027 Strassensmod.py:3(main)
    465/3    0.003    0.000    0.010    0.003 Strassensmod.py:31(Strassen)
       66    0.000    0.000    0.000    0.000 Strassensmod.py:38(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassensmod.py:39(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassensmod.py:40(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassensmod.py:41(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassensmod.py:43(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassensmod.py:44(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassensmod.py:45(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassensmod.py:46(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassensmod.py:62(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassensmod.py:63(<listcomp>)
        1    0.000    0.000    0.027    0.027 Strassensmod.py:8(readmat)
        1    0.000    0.000    0.000    0.000
_bootlocale.py:23(getpreferredencoding)
        1    0.000    0.000    0.000    0.000 codecs.py:259(__init__)
        1    0.000    0.000    0.000    0.000 codecs.py:308(__init__)
        2    0.000    0.000    0.000    0.000 codecs.py:318(decode)
        2    0.000    0.000    0.000    0.000 codecs.py:330(getstate)
        2    0.000    0.000    0.000    0.000 {built-in method
_codecs.utf_8_decode}
        1    0.000    0.000    0.000    0.000 {built-in method
_locale.nl_langinfo}
        1    0.000    0.000    0.027    0.027 {built-in method builtins.exec}
        3    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.000    0.000    0.000    0.000 {built-in method io.open}
       28    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of
'_lsprof.Profiler' objects}
       35    0.016    0.000    0.016    0.000 {method 'readline' of
'_io.TextIOWrapper' objects}
       28    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
        3    0.000    0.000    0.000    0.000 {method 'strip' of 'str' objects}
```

SHUBHIs-MacBook-Pro:Module 2 Sudhir$ python3 -m cProfile Strassens.py

```
C =
 [[16, 17], [26, 22]]
C =
 [[-1, -4, -2, 13], [7, -6, 2, 6], [11, 5, -1, 1], [-6, 9, -8, 3]]
C =
 [[-4, 4, 6, 3, -4, -3, -6, -10], [7, 4, 0, -5, 11, 0, 0, 8], [6, -11, 0, -8, -4, 4, 8, 5], [-4, 11, 7, 1, -6, 1, -4, -3], [-7, 2, -2, 8, 6, -5, -4, 6], [9, -1, 10, -3, 11, 4, 5, 10], [18, 9, 13, 12, 5, 3, 0, -2], [18, -5, -4, -11, -3, 1, 3, 3]]
```

896 function calls (833 primitive calls) in 0.017 seconds

Ordered by: standard name

```
   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.017    0.017 Strassens.py:1(<module>)
       40    0.000    0.000    0.000    0.000 Strassens.py:110(<listcomp>)
       99    0.000    0.000    0.001    0.000 Strassens.py:122(addMat)
      220    0.000    0.000    0.000    0.000 Strassens.py:123(<listcomp>)
       63    0.000    0.000    0.001    0.000 Strassens.py:130(subMat)
       63    0.000    0.000    0.000    0.000 Strassens.py:131(<listcomp>)
       57    0.000    0.000    0.001    0.000 Strassens.py:138(matMultiply)
       57    0.000    0.000    0.000    0.000 Strassens.py:139(<listcomp>)
       14    0.000    0.000    0.000    0.000 Strassens.py:24(<listcomp>)
       14    0.000    0.000    0.000    0.000 Strassens.py:26(<listcomp>)
        1    0.000    0.000    0.017    0.017 Strassens.py:3(main)
     66/3    0.001    0.000    0.003    0.001 Strassens.py:31(Strassen)
        9    0.000    0.000    0.000    0.000 Strassens.py:38(<listcomp>)
        9    0.000    0.000    0.000    0.000 Strassens.py:39(<listcomp>)
        9    0.000    0.000    0.000    0.000 Strassens.py:40(<listcomp>)
        9    0.000    0.000    0.000    0.000 Strassens.py:41(<listcomp>)
        9    0.000    0.000    0.000    0.000 Strassens.py:43(<listcomp>)
        9    0.000    0.000    0.000    0.000 Strassens.py:44(<listcomp>)
        9    0.000    0.000    0.000    0.000 Strassens.py:45(<listcomp>)
        9    0.000    0.000    0.000    0.000 Strassens.py:46(<listcomp>)
        9    0.000    0.000    0.000    0.000 Strassens.py:62(<listcomp>)
        9    0.000    0.000    0.000    0.000 Strassens.py:63(<listcomp>)
        1    0.000    0.000    0.017    0.017 Strassens.py:8(readmat)
        1    0.000    0.000    0.000    0.000 
_bootlocale.py:23(getpreferredencoding)
        1    0.000    0.000    0.000    0.000 codecs.py:259(__init__)
        1    0.000    0.000    0.000    0.000 codecs.py:308(__init__)
        2    0.000    0.000    0.000    0.000 codecs.py:318(decode)
        2    0.000    0.000    0.000    0.000 codecs.py:330(getstate)
        2    0.000    0.000    0.000    0.000 {built-in method 
_codecs.utf_8_decode}
        1    0.000    0.000    0.000    0.000 {built-in method 
_locale.nl_langinfo}
        1    0.000    0.000    0.017    0.017 {built-in method builtins.exec}
        3    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.000    0.000    0.000    0.000 {built-in method io.open}
       28    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of 
'_lsprof.Profiler' objects}
       35    0.013    0.000    0.013    0.000 {method 'readline' of 
'_io.TextIOWrapper' objects}
       28    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
        3    0.000    0.000    0.000    0.000 {method 'strip' of 'str' objects}
```

SHUBHIs-MacBook-Pro:Module 2 Sudhir$ python3 -m cProfile normalMatMulti.py

C =
 [[16, 17], [26, 22]]
C =
 [[-1, -4, -2, 13], [7, -6, 2, 6], [11, 5, -1, 1], [-6, 9, -8, 3]]
C =
 [[-4, 4, 6, 3, -4, -3, -6, -10], [7, 4, 0, -5, 11, 0, 0, 8], [6, -11, 0, -8, -4, 4, 8, 5], [-4, 11, 7, 1, -6, 1, -4, -3], [-7, 2, -2, 8, 6, -5, -4, 6], [9, -1, 10, -3, 11, 4, 5, 10], [18, 9, 13, 12, 5, 3, 0, -2], [18, -5, -4, -11, -3, 1, 3, 3]]

147 function calls in 0.030 seconds

Ordered by: standard name

```
   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.000    0.000 
_bootlocale.py:23(getpreferredencoding)
        1    0.000    0.000    0.000    0.000 codecs.py:259(__init__)
        1    0.000    0.000    0.000    0.000 codecs.py:308(__init__)
        2    0.000    0.000    0.000    0.000 codecs.py:318(decode)
        2    0.000    0.000    0.000    0.000 codecs.py:330(getstate)
        1    0.000    0.000    0.030    0.030 normalMatMulti.py:1(<module>)
        1    0.000    0.000    0.030    0.030 normalMatMulti.py:1(main)
       14    0.000    0.000    0.000    0.000 normalMatMulti.py:22(<listcomp>)
       14    0.000    0.000    0.000    0.000 normalMatMulti.py:24(<listcomp>)
        3    0.000    0.000    0.000    0.000 normalMatMulti.py:28(matMultiply)
        3    0.000    0.000    0.000    0.000 normalMatMulti.py:29(<listcomp>)
        1    0.000    0.000    0.030    0.030 normalMatMulti.py:6(readmat)
        2    0.000    0.000    0.000    0.000 {built-in method 
_codecs.utf_8_decode}
```

```
        1    0.000    0.000    0.000    0.000 {built-in method
_locale.nl_langinfo}
        1    0.000    0.000    0.030    0.030 {built-in method builtins.exec}
        3    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.000    0.000    0.000    0.000 {built-in method io.open}
       28    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of
'_lsprof.Profiler' objects}
       35    0.029    0.001    0.029    0.001 {method 'readline' of
'_io.TextIOWrapper' objects}
       28    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
        3    0.000    0.000    0.000    0.000 {method 'strip' of 'str' objects}
```

SHUBHIs-MacBook-Pro:Module 2 Sudhir$ python3 -m cProfile Strassensmod.py

```
C =
 [[16, 17], [26, 22]]
C =
 [[-1, -4, -2, 13], [7, -6, 2, 6], [11, 5, -1, 1], [-6, 9, -8, 3]]
C =
 [[-4, 4, 6, 3, -4, -3, -6, -10], [7, 4, 0, -5, 11, 0, 0, 8], [6, -11, 0, -8, -4, 4, 8, 5], [-4, 11, 7, 1, -6, 1, -4, -3], [-7, 2, -2, 8, 6, -5, -4, 6], [9, -1, 10, -3, 11, 4, 5, 10], [18, 9, 13, 12, 5, 3, 0, -2], [18, -5, -4, -11, -3, 1, 3, 3]]
C =
 [[4, -1, -3, -5, 9, -9, -8, -5, -3, 0, -1, -18, -2, -8, -9, -7], [14, 18, 13, 10, -4, -7, -3, 4, 13, 22, 4, 3, -3, 1, -3, 10], [14, 14, -6, -5, 24, 5, -3, 11, 20, -6, 13, -15, 16, 11, 14, 11], [1, 13, 17, 25, 0, 19, 29, 29, 38, 23, 4, 34, 14, 23, 26, 46], [22, 26, -10, -4, 26, 22, -4, 18, 30, -6, 32, -4, 22, 30, 34, 20], [49, 58, 7, 29, 69, 106, 17, 73, 114, 2, 96, 39, 61, 122, 127, 71], [10, 7, 6, 10, 20, 13, 3, 13, 28, 6, 10, -1, 11, 18, 17, 13], [1, 13, 17, 25, 0, 19, 29, 29, 38, 23, 4, 34, 14, 23, 26, 46], [4, -1, -3, -5, 9, -9, -8, -5, -3, 0, -1, -18, -2, -8, -9, -7], [14, 18, 13, 10, -4, -7, -3, 4, 13, 22, 4, 3, -3, 1, -3, 10], [14, 14, -6, -5, 24, 5, -3, 11, 20, -6, 13, -15, 16, 11, 14, 11], [1, 13, 17, 25, 0, 19, 29, 29, 38, 23, 4, 34, 14, 23, 26, 46], [22, 26, -10, -4, 26, 22, -4, 18, 30, -6, 32, -4, 22, 30, 34, 20], [49, 58, 7, 29, 69, 106, 17, 73, 114, 2, 96, 39, 61, 122, 127, 71], [10, 7, 6, 10, 20, 13, 3, 13, 28, 6, 10, -1, 11, 18, 17, 13], [1, 13, 17, 25, 0, 19, 29, 29, 38, 23, 4, 34, 14, 23, 26, 46]]
```

22083 function calls (18821 primitive calls) in 0.061 seconds

Ordered by: standard name

```
   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.068    0.068 Strassensmod.py:1(<module>)
      308    0.000    0.000    0.000    0.000 Strassensmod.py:120(<listcomp>)
      400    0.001    0.000    0.001    0.000 Strassensmod.py:139(<listcomp>)
     3126    0.012    0.000    0.019    0.000 Strassensmod.py:151(addMat)
     3126    0.005    0.000    0.007    0.000 Strassensmod.py:152(<listcomp>)
     2062    0.008    0.000    0.013    0.000 Strassensmod.py:158(subMat)
     2062    0.004    0.000    0.005    0.000 Strassensmod.py:159(<listcomp>)
     2800    0.001    0.000    0.001    0.000 Strassensmod.py:174(matMultiply)
       30    0.000    0.000    0.000    0.000 Strassensmod.py:24(<listcomp>)
       30    0.000    0.000    0.000    0.000 Strassensmod.py:26(<listcomp>)
        1    0.000    0.000    0.068    0.068 Strassensmod.py:3(main)
   3266/4    0.021    0.000    0.066    0.017 Strassensmod.py:31(Strassen)
      466    0.001    0.000    0.001    0.000 Strassensmod.py:38(<listcomp>)
      466    0.001    0.000    0.001    0.000 Strassensmod.py:39(<listcomp>)
      466    0.001    0.000    0.001    0.000 Strassensmod.py:40(<listcomp>)
      466    0.001    0.000    0.001    0.000 Strassensmod.py:41(<listcomp>)
      466    0.001    0.000    0.001    0.000 Strassensmod.py:43(<listcomp>)
      466    0.001    0.000    0.001    0.000 Strassensmod.py:44(<listcomp>)
      466    0.001    0.000    0.001    0.000 Strassensmod.py:45(<listcomp>)
      466    0.001    0.000    0.001    0.000 Strassensmod.py:46(<listcomp>)
      466    0.001    0.000    0.001    0.000 Strassensmod.py:62(<listcomp>)
      466    0.001    0.000    0.001    0.000 Strassensmod.py:63(<listcomp>)
        1    0.000    0.000    0.068    0.068 Strassensmod.py:8(readmat)
        1    0.000    0.000    0.000    0.000
_bootlocale.py:23(getpreferredencoding)
        1    0.000    0.000    0.000    0.000 codecs.py:259(__init__)
        1    0.000    0.000    0.000    0.000 codecs.py:308(__init__)
        2    0.000    0.000    0.000    0.000 codecs.py:318(decode)
        2    0.000    0.000    0.000    0.000 codecs.py:330(getstate)
        2    0.000    0.000    0.000    0.000 {built-in method
_codecs.utf_8_decode}
        1    0.000    0.000    0.000    0.000 {built-in method
_locale.nl_langinfo}
        1    0.000    0.000    0.068    0.068 {built-in method builtins.exec}
        4    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.000    0.000    0.000    0.000 {built-in method io.open}
       60    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of
'_lsprof.Profiler' objects}
       69    0.000    0.000    0.000    0.000 {method 'readline' of
'_io.TextIOWrapper' objects}
       60    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
        4    0.000    0.000    0.000    0.000 {method 'strip' of 'str' objects}
```

SHUBHIs-MacBook-Pro:Module 2 Sudhir$ python3 -m cProfile Strassens.py

```
C =
 [[16, 17], [26, 22]]
C =
 [[-1, -4, -2, 13], [7, -6, 2, 6], [11, 5, -1, 1], [-6, 9, -8, 3]]
C =
 [[-4, 4, 6, 3, -4, -3, -6, -10], [7, 4, 0, -5, 11, 0, 0, 8], [6, -11, 0, -8, -4, 4, 8, 5], [-4, 11, 7, 1, -6, 1, -4, -3], [-7, 2, -2, 8, 6, -5, -4, 6], [9, -1, 10, -3, 11, 4, 5, 10], [18, 9, 13, 12, 5, 3, 0, -2], [18, -5, -4, -11, -3, 1, 3, 3]]
C =
 [[4, -1, -3, -5, 9, -9, -8, -5, -3, 0, -1, -18, -2, -8, -9, -7], [14, 18, 13, 10, -4, -7, -3, 4, 13, 22, 4, 3, -3, 1, -3, 10], [14, 14, -6, -5, 24, 5, -3, 11, 20, -6, 13, -15, 16, 11, 14, 11], [1, 13, 17, 25, 0, 19, 29, 29, 38, 23, 4, 34, 14, 23, 26, 46], [22, 26, -10, -4, 26, 22, -4, 18, 30, -6, 32, -4, 22, 30, 34, 20], [49, 58, 7, 29, 69, 106, 17, 73, 114, 2, 96, 39, 61, 122, 127, 71], [10, 7, 6, 10, 20, 13, 3, 13, 28, 6, 10, -1, 11, 18, 17, 13], [1, 13, 17, 25, 0, 19, 29, 29, 38, 23, 4, 34, 14, 23, 26, 46], [4, -1, -3, -5, 9, -9, -8, -5, -3, 0, -1, -18, -2, -8, -9, -7], [14, 18, 13, 10, -4, -7, -3, 4, 13, 22, 4, 3, -3, 1, -3, 10], [14, 14, -6, -5, 24, 5, -3, 11, 20, -6, 13, -15, 16, 11, 14, 11], [1, 13, 17, 25, 0, 19, 29, 29, 38, 23, 4, 34, 14, 23, 26, 46], [22, 26, -10, -4, 26, 22, -4, 18, 30, -6, 32, -4, 22, 30, 34, 20], [49, 58, 7, 29, 69, 106, 17, 73, 114, 2, 96, 39, 61, 122, 127, 71], [10, 7, 6, 10, 20, 13, 3, 13, 28, 6, 10, -1, 11, 18, 17, 13], [1, 13, 17, 25, 0, 19, 29, 29, 38, 23, 4, 34, 14, 23, 26, 46]]
```

5851 function calls (5389 primitive calls) in 0.022 seconds

```
   Ordered by: standard name

   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.027    0.027 Strassens.py:1(<module>)
```

```
      308    0.000    0.000    0.000    0.000 Strassens.py:110(<listcomp>)
      726    0.004    0.000    0.007    0.000 Strassens.py:122(addMat)
     1694    0.001    0.000    0.001    0.000 Strassens.py:123(<listcomp>)
      462    0.003    0.000    0.005    0.000 Strassens.py:130(subMat)
      462    0.001    0.000    0.002    0.000 Strassens.py:131(<listcomp>)
      400    0.004    0.000    0.005    0.000 Strassens.py:138(matMultiply)
      400    0.001    0.000    0.001    0.000 Strassens.py:139(<listcomp>)
       30    0.000    0.000    0.000    0.000 Strassens.py:24(<listcomp>)
       30    0.000    0.000    0.000    0.000 Strassens.py:26(<listcomp>)
        1    0.000    0.000    0.027    0.027 Strassens.py:3(main)
    466/4    0.005    0.000    0.025    0.006 Strassens.py:31(Strassen)
       66    0.000    0.000    0.000    0.000 Strassens.py:38(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassens.py:39(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassens.py:40(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassens.py:41(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassens.py:43(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassens.py:44(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassens.py:45(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassens.py:46(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassens.py:62(<listcomp>)
       66    0.000    0.000    0.000    0.000 Strassens.py:63(<listcomp>)
        1    0.000    0.000    0.027    0.027 Strassens.py:8(readmat)
        1    0.000    0.000    0.000    0.000
_bootlocale.py:23(getpreferredencoding)
        1    0.000    0.000    0.000    0.000 codecs.py:259(__init__)
        1    0.000    0.000    0.000    0.000 codecs.py:308(__init__)
        2    0.000    0.000    0.000    0.000 codecs.py:318(decode)
        2    0.000    0.000    0.000    0.000 codecs.py:330(getstate)
        2    0.000    0.000    0.000    0.000 {built-in method
_codecs.utf_8_decode}
        1    0.000    0.000    0.000    0.000 {built-in method
_locale.nl_langinfo}
        1    0.000    0.000    0.027    0.027 {built-in method builtins.exec}
        4    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.000    0.000    0.000    0.000 {built-in method io.open}
       60    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of
'_lsprof.Profiler' objects}
       69    0.000    0.000    0.000    0.000 {method 'readline' of
'_io.TextIOWrapper' objects}
       60    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
        4    0.000    0.000    0.000    0.000 {method 'strip' of 'str' objects}
```

SHUBHIs-MacBook-Pro:Module 2 Sudhir$ python3 -m cProfile normalMatMulti.py

```
C =
 [[16, 17], [26, 22]]
C =
 [[-1, -4, -2, 13], [7, -6, 2, 6], [11, 5, -1, 1], [-6, 9, -8, 3]]
C =
 [[-4, 4, 6, 3, -4, -3, -6, -10], [7, 4, 0, -5, 11, 0, 0, 8], [6, -11, 0, -8, -4, 4, 8, 5], [-4, 11, 7, 1, -6, 1, -4, -3], [-7, 2, -
2, 8, 6, -5, -4, 6], [9, -1, 10, -3, 11, 4, 5, 10], [18, 9, 13, 12, 5, 3, 0, -2], [18, -5, -4, -11, -3, 1, 3, 3]]
C =
 [[4, -1, -3, -5, 9, -9, -8, -5, -3, 0, -1, -18, -2, -8, -9, -7], [14, 18, 13, 10, -4, -7, -3, 4, 13, 22, 4, 3, -3, 1, -3, 10],
[14, 14, -6, -5, 24, 5, -3, 11, 20, -6, 13, -15, 16, 11, 14, 11], [1, 13, 17, 25, 0, 19, 29, 29, 38, 23, 4, 34, 14, 23, 26,
```

46], [22, 26, -10, -4, 26, 22, -4, 18, 30, -6, 32, -4, 22, 30, 34, 20], [49, 58, 7, 29, 69, 106, 17, 73, 114, 2, 96, 39, 61, 122, 127, 71], [10, 7, 6, 10, 20, 13, 3, 13, 28, 6, 10, -1, 11, 18, 17, 13], [1, 13, 17, 25, 0, 19, 29, 29, 38, 23, 4, 34, 14, 23, 26, 46], [4, -1, -3, -5, 9, -9, -8, -5, -3, 0, -1, -18, -2, -8, -9, -7], [14, 18, 13, 10, -4, -7, -3, 4, 13, 22, 4, 3, -3, 1, -3, 10], [14, 14, -6, -5, 24, 5, -3, 11, 20, -6, 13, -15, 16, 11, 14, 11], [1, 13, 17, 25, 0, 19, 29, 29, 38, 23, 4, 34, 14, 23, 26, 46], [22, 26, -10, -4, 26, 22, -4, 18, 30, -6, 32, -4, 22, 30, 34, 20], [49, 58, 7, 29, 69, 106, 17, 73, 114, 2, 96, 39, 61, 122, 127, 71], [10, 7, 6, 10, 20, 13, 3, 13, 28, 6, 10, -1, 11, 18, 17, 13], [1, 13, 17, 25, 0, 19, 29, 29, 38, 23, 4, 34, 14, 23, 26, 46]]

281 function calls in 0.004 seconds

Ordered by: standard name

```
   ncalls  tottime  percall  cumtime  percall filename:lineno(function)
        1    0.000    0.000    0.000    0.000 
_bootlocale.py:23(getpreferredencoding)
        1    0.000    0.000    0.000    0.000 codecs.py:259(__init__)
        1    0.000    0.000    0.000    0.000 codecs.py:308(__init__)
        2    0.000    0.000    0.000    0.000 codecs.py:318(decode)
        2    0.000    0.000    0.000    0.000 codecs.py:330(getstate)
        1    0.000    0.000    0.003    0.003 normalMatMulti.py:1(<module>)
       30    0.000    0.000    0.000    0.000 normalMatMulti.py:25(<listcomp>)
       30    0.000    0.000    0.000    0.000 normalMatMulti.py:27(<listcomp>)
        4    0.002    0.001    0.002    0.001 normalMatMulti.py:32(matMultiply)
        4    0.000    0.000    0.000    0.000 normalMatMulti.py:33(<listcomp>)
        1    0.000    0.000    0.003    0.003 normalMatMulti.py:4(main)
        1    0.000    0.000    0.003    0.003 normalMatMulti.py:9(readmat)
        2    0.000    0.000    0.000    0.000 {built-in method 
_codecs.utf_8_decode}
        1    0.000    0.000    0.000    0.000 {built-in method 
_locale.nl_langinfo}
        1    0.000    0.000    0.003    0.003 {built-in method builtins.exec}
        4    0.000    0.000    0.000    0.000 {built-in method builtins.print}
        1    0.000    0.000    0.000    0.000 {built-in method io.open}
       60    0.000    0.000    0.000    0.000 {method 'append' of 'list' objects}
        1    0.000    0.000    0.000    0.000 {method 'disable' of 
'_lsprof.Profiler' objects}
       69    0.000    0.000    0.000    0.000 {method 'readline' of 
'_io.TextIOWrapper' objects}
       60    0.000    0.000    0.000    0.000 {method 'split' of 'str' objects}
        4    0.000    0.000    0.000    0.000 {method 'strip' of 'str' objects}
```