Kanishk Singh-SE21UCAM005 (CAM)

Lab 4- Naive Matrix multiplication vs Strassen Matrix multiplication

Functions used:

I have used 3 functions for matrix multiplication:

1. For loop method

```
def matrix_multiply(n, a, b):
    answer = np.zeros([n, n])
    for i in range(n):
        for j in range(n):
            for k in range(n):
                answer[i][j] += a[i][k] * b[k][j]
    return answer
```

2. np.matmul and the @ operator :included in numpy, also has $O(n^3)$ complexity

```
def vectorised_matrix_multi(A, B):
    result = A @ B
    return result

def vectorised_matrix_multi(A, B):
    result = np.matmul(A, B)
    return result
```

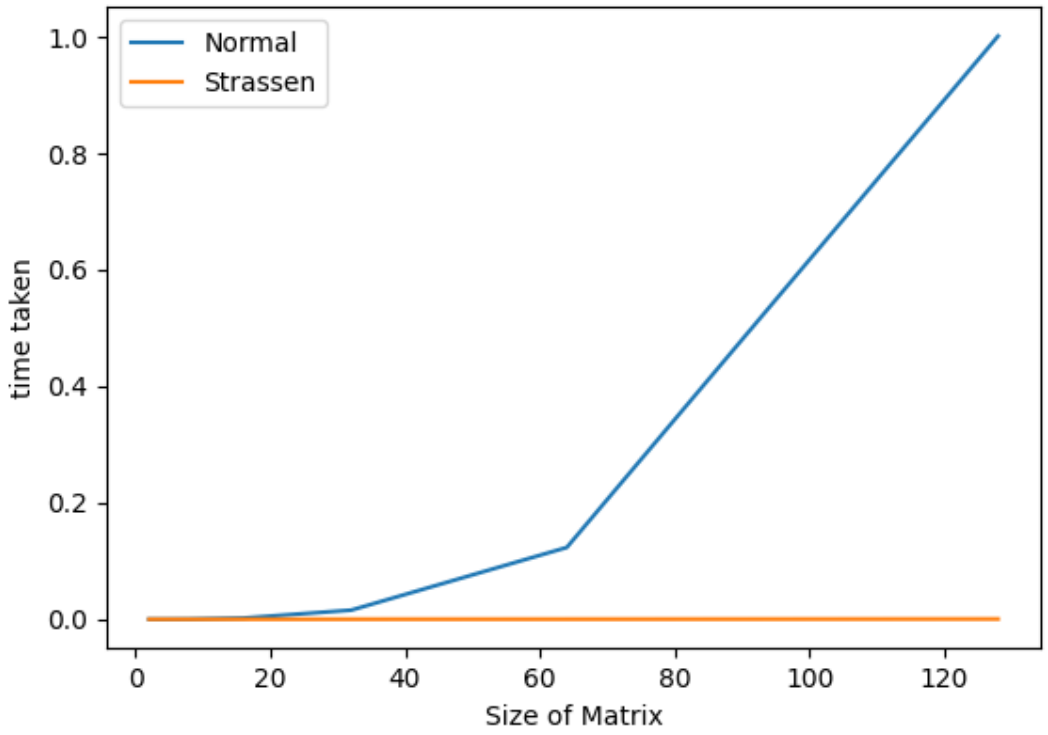
Although I have used matmul directly since there was no need for a func

3. Strassen Method:

```
def strassen_matrix_multiply(n, A, B):
    if not log2(n).is_integer():
        raise Exception("Cannot multiply")
    half = int(n / 2)
    a = A[0:half, 0:half]
    b = A[0:half, half:]
    c = A[half:, 0:half]
    d = A[half:, half:]
    e = B[0:half, 0:half]
    f = B[0:half, half:]
    g = B[half:, 0:half]
    h = B[half:, half:]
    m1 = np.matmul((a + c), (e + f))
    m2 = np.matmul((b + d), (g + h))
```

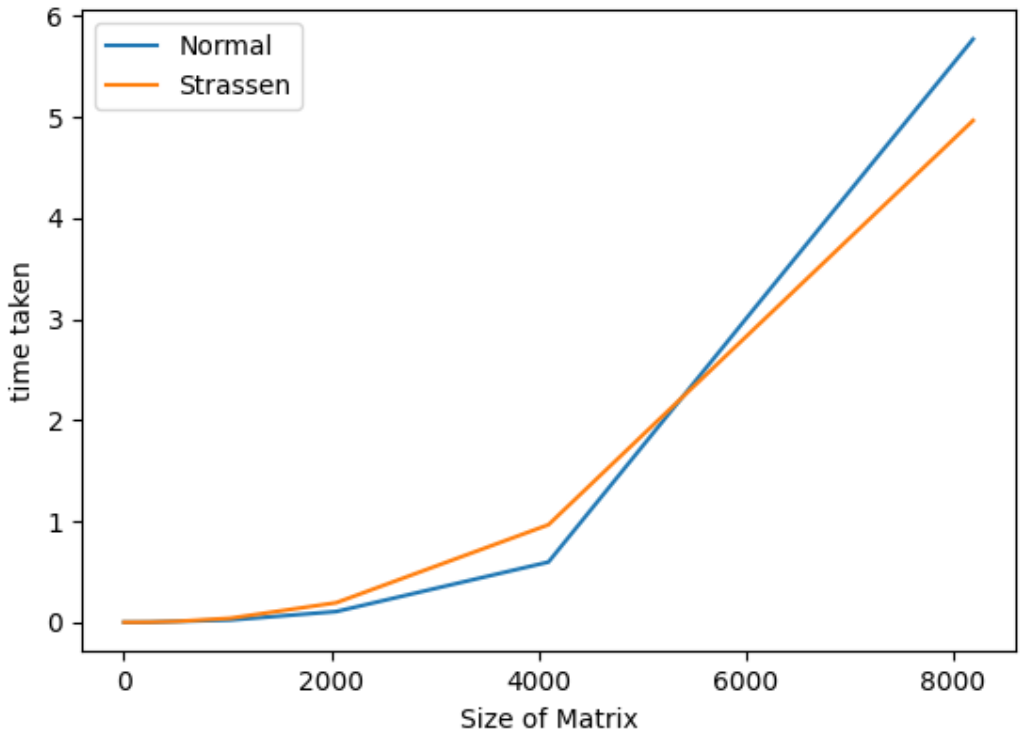
```
    m3 = np.matmul((a - d), (e + h))
    m4 = np.matmul(a, (f - h))
    m5 = np.matmul((c + d), e)
    m6 = np.matmul((a + b), h)
    m7 = np.matmul(d, (g - e))
    result = np.zeros([n, n])
    result[0:half, 0:half] = m2 + m3 - m6 - m7
    result[0:half, half:] = m4 + m6
    result[half:, 0:half] = m5 + m7
    result[half:, half:] = m1 - m3 - m4 - m5
    return result
```

Case 1: For loop vs Strassen



As we can see nested for loop is so slow that strassen looks flat in comparison, So in next case we will do a more fair comparison to see how and when strassen is efficient

Case 2:



Here I have used numpy's BLAS(Basic Linear Algebra Subprograms) based Matrix multiplier.

Since it can utilise multi thread it is faster than strassen up until matrices of size 2^{12} in case of my computer which has 8 cores and 16 threads.

Still, we can see that strassen is more efficient of larger size matrix even when compared to a highly optimised Algorithm from Numpy.

Conclusion: Strassen is a very efficient way to multiply matrix compared to nested for loops and even numpy tools when needed for large matrices of size 2^n where n is a positive integer.