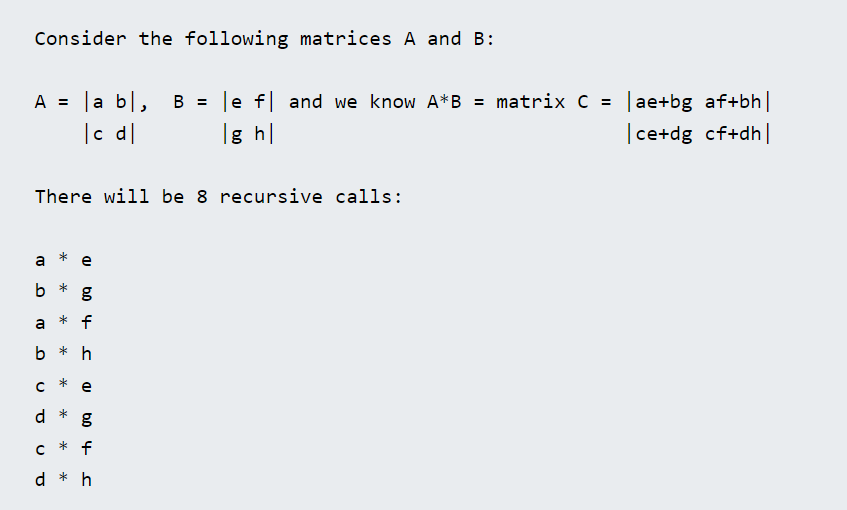
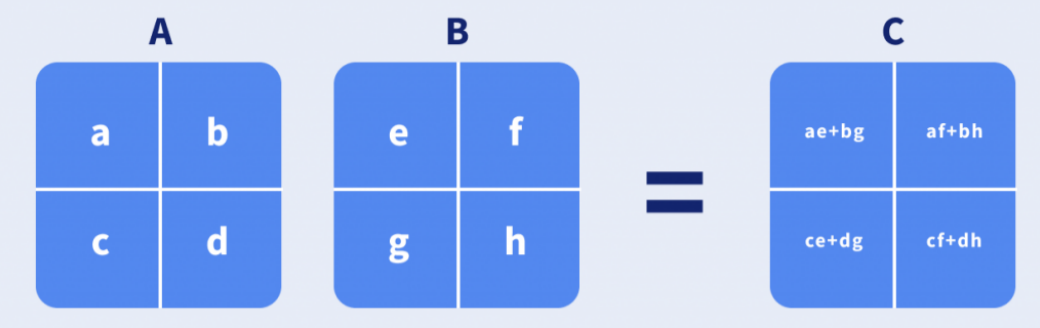
|  |  |
| --- | --- |
| **Name** | Akshaya Karande |
| **UID No.** | 2021300056 |
| **Class & Division** | S.E. COMPS A (BATCH-D) |
| **Experiment No.** | 3 |

**Aim:** Strassen’s Matrix Multiplication

**Theory:**

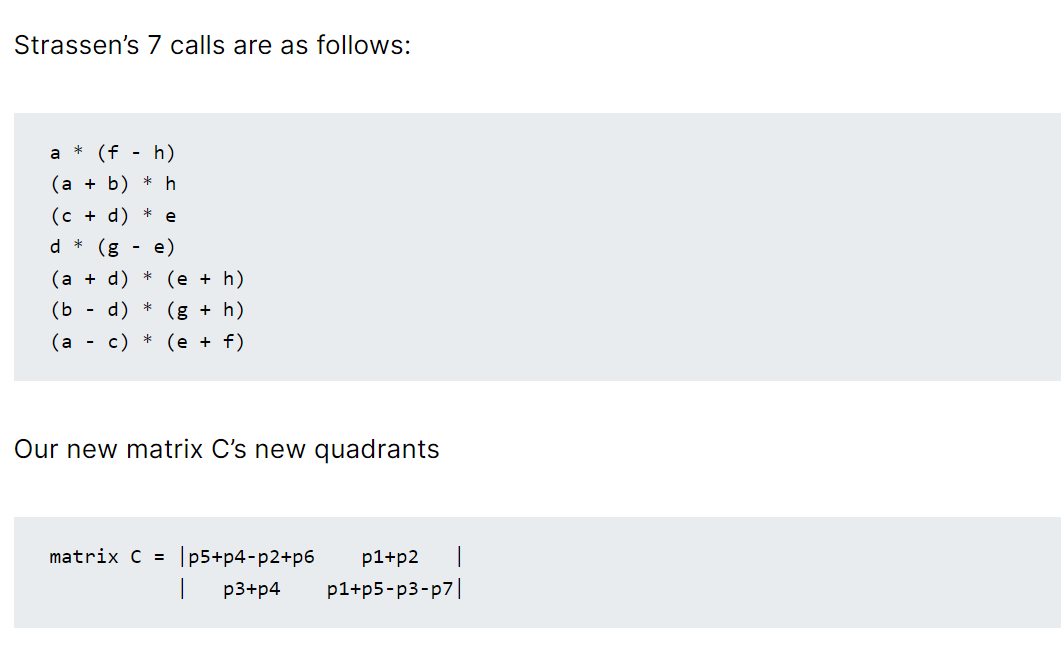
Naïve Method:

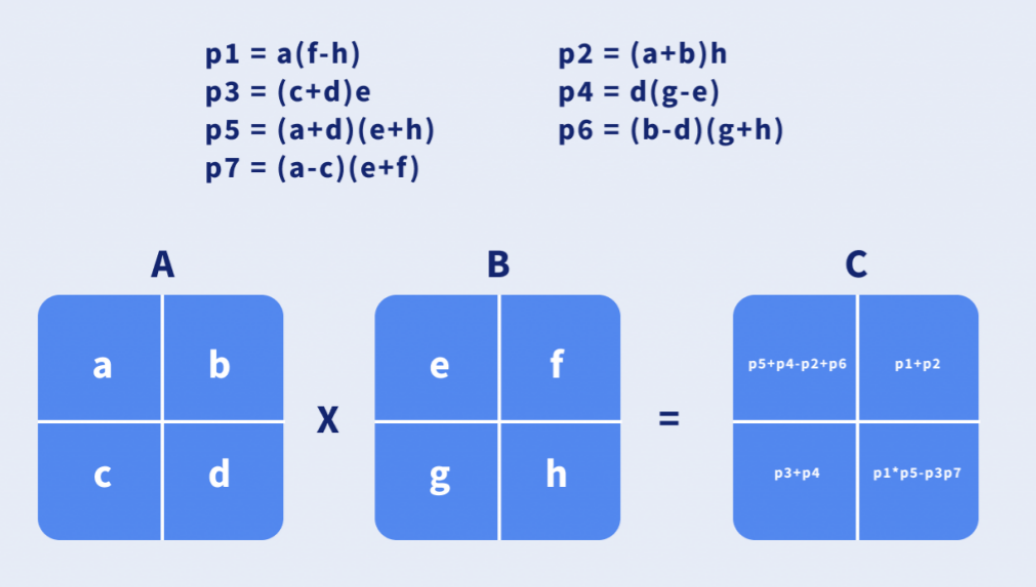


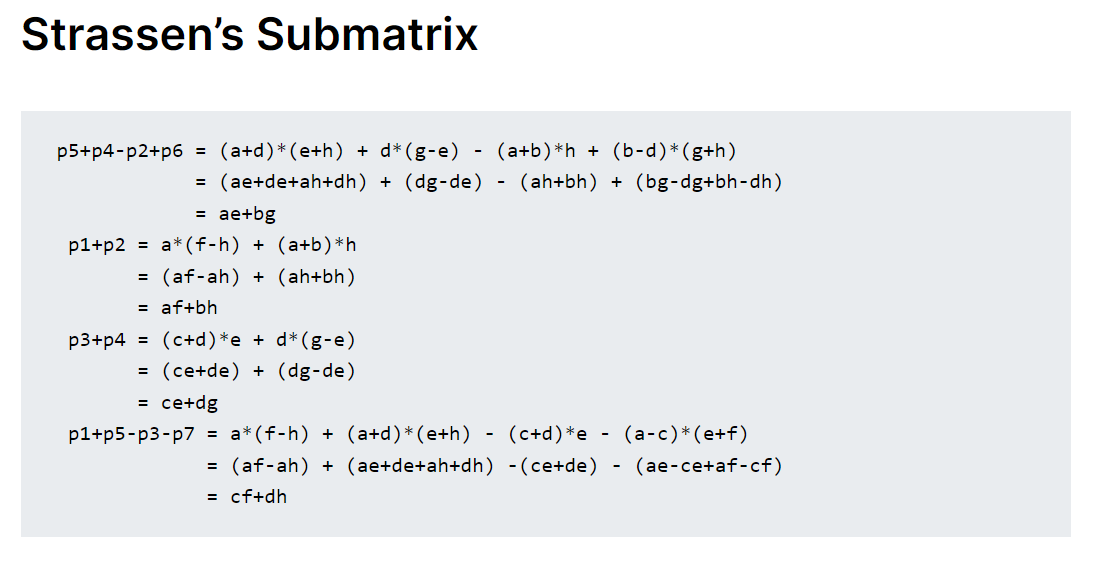


Using the Master Theorem with **T(n) = 8T(n/2) + O(n^2)** we still get a runtime of **O(n^3)**.

Strassen’s Matrix Multiplication:







The time complexity using the Master Theorem.

**T(n) = 7T(n/2) + O(n^2)** = **O(n^log(7))** runtime.

Approximately **O(n^2.8074)** which is better than O(n^3)

**Strassen’s Matrix Multiplication Algorithm:**

* Divide matrix A and matrix B in 4 sub-matrices of size N/2 x N/2 as shown in the above diagram.
* Calculate the 7 matrix multiplications recursively.
* Compute the submatrices of C.
* Combine these submatricies into our new matrix C

**Code:**

#include<bits/stdc++.h>

using namespace std;

int main(){

int n=2;

vector<vector<int>> a(n, vector<int> (n,0)), b(n, vector<int> (n,0));

vector<vector<int>> c(n, vector<int> (n,0));

cout<<"Enter Matrix 1: "<<endl;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++) cin>>a[i][j];

}

cout<<"Enter Matrix 2: "<<endl;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++) cin>>b[i][j];

}

int s1=b[0][1]-b[1][1];

int s2=a[0][0]+a[0][1];

int s3=a[1][0]+a[1][1];

int s4=b[1][0]-b[0][0];

int s5=a[0][0]+a[1][1];

int s6=b[0][0]+b[1][1];

int s7=a[0][1]-a[1][1];

int s8=b[1][0]+b[1][1];

int s9=a[0][0]-a[1][0];

int s10=b[0][0]+b[0][1];

int p1=a[0][0]\*s1;

int p2=b[1][1]\*s2;

int p3=b[0][0]\*s3;

int p4=a[1][1]\*s4;

int p5=s5\*s6;

int p6=s7\*s8;

int p7=s9\*s10;

c[0][0]=p5+p4-p2+p6;

c[0][1]=p1+p2;

c[1][0]=p3+p4;

c[1][1]=p5+p1-p3-p7;

cout<<"Matrix1 \* Matrix2: "<<endl;

for (int i = 0; i < n; i++)

{

for (int j = 0; j < n; j++) cout<<c[i][j]<<" ";

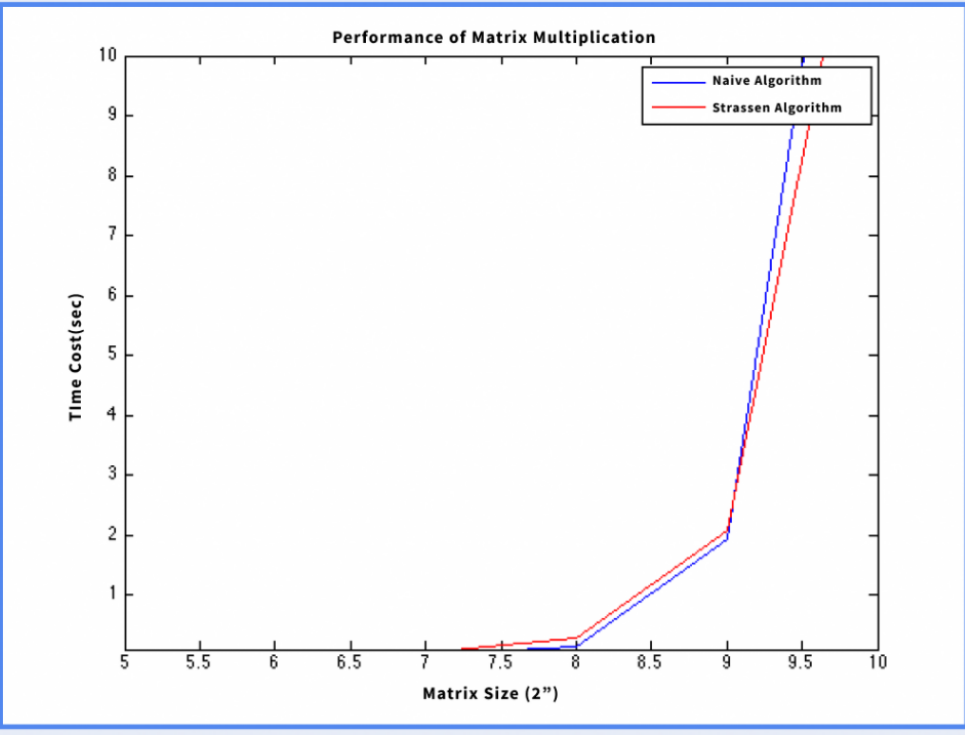
cout<<endl;

}

return 0;

}

**Observation**:



**Conclusion**:

It is faster than the standard matrix multiplication algorithm for large matrices, with a better asymptotic complexity.