ASSIGNMENT NO 4

Title of assignment: Divide and Conquer Strategy

Name: Mayur Savale
PRN:21520010

Strassen's Matrix Multiplication

1. Implement Naive Method multiply tow matrices and justify Complexity is $O(n^3)$.

Ans:

a) **Algorithm: (Pseudocode)**

- We know that value of any element in resultant matrix is the sum of the product of the i^{th} row of first matrix and the j^{th} column of second matrix.
- So we traverse the matrix in to for loop and then apply third loop in the two for loop for sum of the products of the row and column.

b) **Code snapshots of implementation**

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
   int n;
   cout<<"Enter size of square matrix: ";
   cin>>n;
   int a[n][n];
   int b[n][n];
   cout<<"Enter elements of matrix A:\n";
   for(int i=0;i<n;i++)
   {
      for(int j=0;j<n;j++)
         cin>>a[i][j];
   }
   cout<<"Enter elements of matrix B:\n";
```

```
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
            cin>>b[i][j];
    }
    cout<<"Matrix Multiplication is\n";
    int  c[n][n];
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
        {
            c[i][j]=0;
            for(int k=0;k<n;k++)
                c[i][j]+=a[i][k]*b[k][j];
        }
    }
    for(int i=0;i<n;i++)
    {
        for(int j=0;j<n;j++)
            cout<<c[i][j]<<"\t";
        cout<<"\n";
    }
    return 0;
}
```

Output:

```
C:\Users\mayur\OneDrive\Documents\C\c++\DAA1.exe
Enter size of square matrix: 3
Enter elements of matrix A:
2
3
4
5
1
2
3
2
6
Enter elements of matrix B:
7
3
2
6
7
8
9
7
5
Matrix Multiplication is
68      55      48
59      36      28
87      65      52

--------------------------------
Process exited after 19.58 seconds with return value 0
Press any key to continue . . .
```

c) **Complexity of proposed algorithm (Time & Space)**

- Time Complexity: $O(n^3)$
  For accessing all the elements of any matrix we need two for loops. But to find the product we require on additional for loop
- Space Complexity: $O(n^2)$

2. Implement Divide and Conquer multiply two matrices and justify Complexity is $O(n^3)$

ASSIGNMENT NO 4

$$\underset{A}{\begin{bmatrix} a & b \\ c & d \end{bmatrix}} \times \underset{B}{\begin{bmatrix} e & f \\ g & h \end{bmatrix}} = \underset{C}{\begin{bmatrix} ae + bg & af + bh \\ ce + dg & cf + dh \end{bmatrix}}$$

A, B and C are square metrices of size N x N
a, b, c and d are submatrices of A, of size N/2 x N/2
e, f, g and h are submatrices of B, of size N/2 x N/2

Ans:

a) Algorithm: (Pseudocode)

- Divide matrices A and B in 4 sub-matrices of size N/2 x N/2 as shown in figure
- Calculate the values recursively ae+bg,af+bh,ce+dg,cf+dh

b) Code snapshots of implementation

```
#include<bits/stdc++.h>
using namespace std;

const int MAX=100;

void multiplyMatrixRec(int r1,int c1,int a[][MAX],int r2,int c2,int b[][MAX],int c[][MAX])
{
   static int i=0,j=0,k=0;
   if(i>=r1)
      return;
   if(j<c2)
   {
      if(k<c1)
      {
         c[i][j]+=a[i][k]*b[k][j];
         k++;
         multiplyMatrixRec(r1,c1,a,r2,c2,b,c);
```

```
        }
        k=0;
        j++;
        multiplyMatrixRec(r1,c1,a,r2,c2,b,c);
    }
    j=0;
    i++;
    multiplyMatrixRec(r1,c1,a,r2,c2,b,c);
}

void multiplyMatrix(int r1,int c1,int a[][MAX],int r2,int c2,int b[][MAX])
{
    if(r2!=c1)
    {
        cout<<"Not Possible\n";
        return;
    }
    int c[MAX][MAX]={0};
    multiplyMatrixRec(r1,c1,a,r2,c2,b,c);
    for(int i=0;i<r1;i++)
    {
        for(int j=0;j<c2;j++)
            cout<<c[i][j]<<"\t";
        cout<<"\n";
    }
}

int main()
{
    int a[][MAX]={{1,2,3},{4,5,6},{7,8,9}};
    int b[][MAX]={{1,2,3},{4,5,6},{7,8,9}};
    cout<<"Matrix Multiplication is\n";
    multiplyMatrix(3,3,a,3,3,b);
    return 0;
}
```

Output:

```
C:\Users\mayur\OneDrive\Documents\C\c++\DAA2.exe
Matrix Multiplication is
30      36      42
66      81      96
102     126     150

--------------------------------
Process exited after 0.2343 seconds with return value 0
Press any key to continue . . .
```

c) Complexity of proposed algorithm (Time & Space)

- Time Complexity: $O(n^3)$

  To solve this problem using D & C for the given problem we are doing * multiplications for matrices of size N/2 x N/2 and 4 additions

  Addition of two matrices take (n^2) time.

  So we can write the complexity as

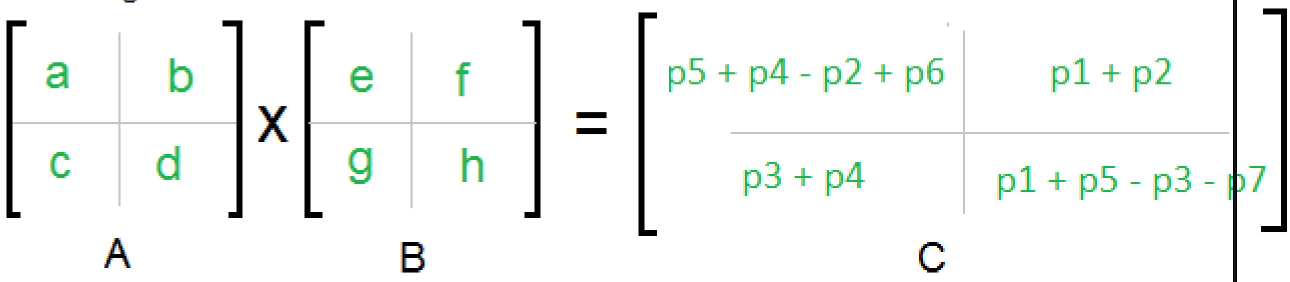
  $T(n) = 8*T(N/2) + O(n^2)$

ASSIGNMENT NO 4

3. Implement Strassen's Matrix Multiplication and justify Complexity is $O(n^{2.8})$

p1 = a(f - h)
p2 = (a + b)h
p3 = (c + d)e
p4 = d(g - e)
p5 = (a + d)(e + h)
p6 = (b - d)(g + h)
p7 = (a - c)(e + f)

The A x B can be calculated using above seven multiplications.
Following are values of four sub-matrices of result C



A, B and C are square metrices of size N x N
a, b, c and d are submatrices of A, of size N/2 x N/2
e, f, g and h are submatrices of B, of size N/2 x N/2
p1, p2, p3, p4, p5, p6 and p7 are submatrices of size N/2 x N/2

Ans:

a) Algorithm: (Pseudocode)

- It is similar as divide and conquer method only difference is that the fore submatrices of the result are calculated using the above formula

b) Code snapshots of implementation

```
#include<bits/stdc++.h>
using namespace std;

int main()
{
   int a[2][2],b[2][2],c[2][2],i,j;
   int m1,m2,m3,m4,m5,m6,m7;

   cout<<"Enter the elements of first matrix:\n";
   for(i=0;i<2;i++)
   {
```

```
        for(j=0;j<2;j++)
            cin>>a[i][j];
    }
    cout<<"Enter the elements of second matrix:\n";
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
            cin>>b[i][j];
    }
    m1=(a[0][0]+a[1][1])*(b[0][0]+b[1][1]);
    m2=(a[1][0]+a[1][1])*b[0][0];
    m3=a[0][0]*(b[0][1]-b[1][1]);
    m4=a[1][1]*(b[1][0]-b[0][0]);
    m5=(a[0][0]+a[0][1])*b[1][1];
    m6=(a[1][0]-a[0][0])*(b[0][0]+b[0][1]);
    m7=(a[0][1]-a[1][1])*(b[1][0]+b[1][1]);
    c[0][0]=m1+m4-m5+m7;
    c[0][1]=m3+m5;
    c[1][0]=m2+m4;
    c[1][1]=m1-m2+m3+m6;
    cout<<"After multiplication of matrices:\n";
    for(i=0;i<2;i++)
    {
        for(j=0;j<2;j++)
            cout<<c[i][j]<<"\t";
            cout<<"\n";
    }
}
```

Output:

```
C:\Users\mayur\OneDrive\Documents\C\c++\DAA3.exe
Enter the elements of first matrix:
2
3
4
5
Enter the elements of second matrix:
7
8
6
9
After multiplication of matrices:
32      43
58      77

--------------------------------
Process exited after 12.86 seconds with return value 0
Press any key to continue . . .
```

c) **Complexity of proposed algorithm (Time & Space)**

- Time Complexity:

  Addition and Subtraction of two matrices take O(N2) time.

  So time complexity can be written as

  T(N) = 7T(N/2) + O(N2)

  From Master's Theorem, time complexity of above methos is O(Nlog7) which is approximately O(N2.8074)

  Generally, Strassen's Method is not preferred for practical application for the following reasons.

  1. The constants used is Strassen's method are high and for a typical application Naïve method works better.
  2. For Spares matrices, there are better methods especially designed for them.