**Name: Khushi Nitinkumar Patel**

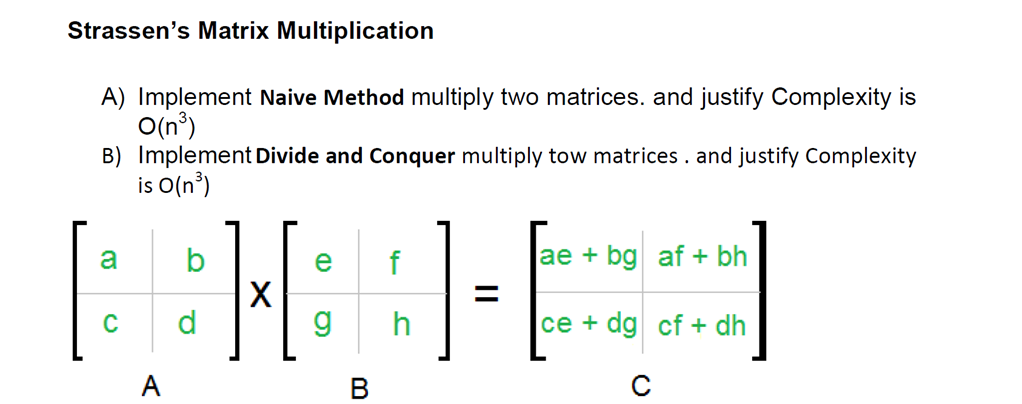
**PRN: 2020BTECS00037**

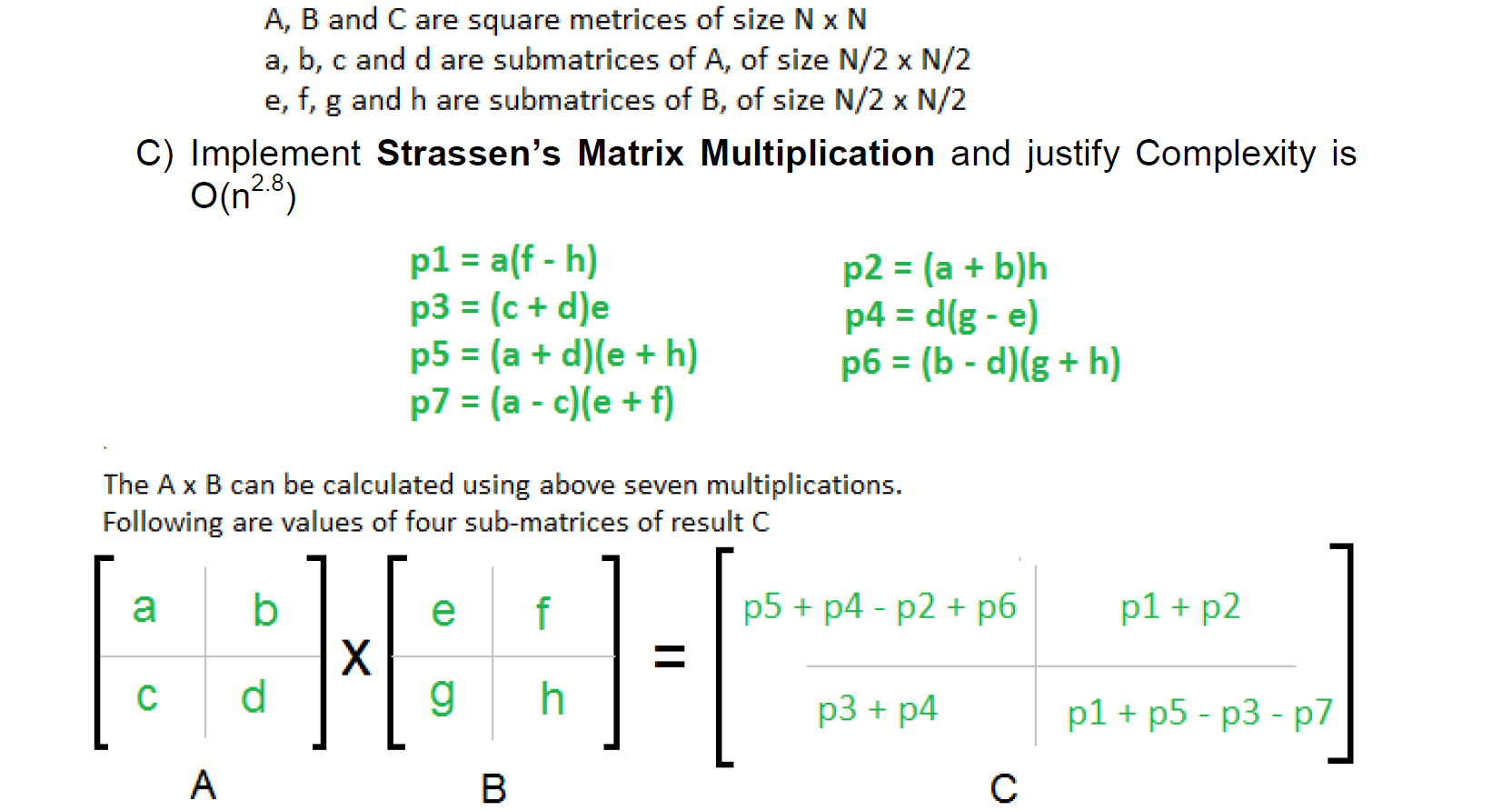
**Batch: T5**

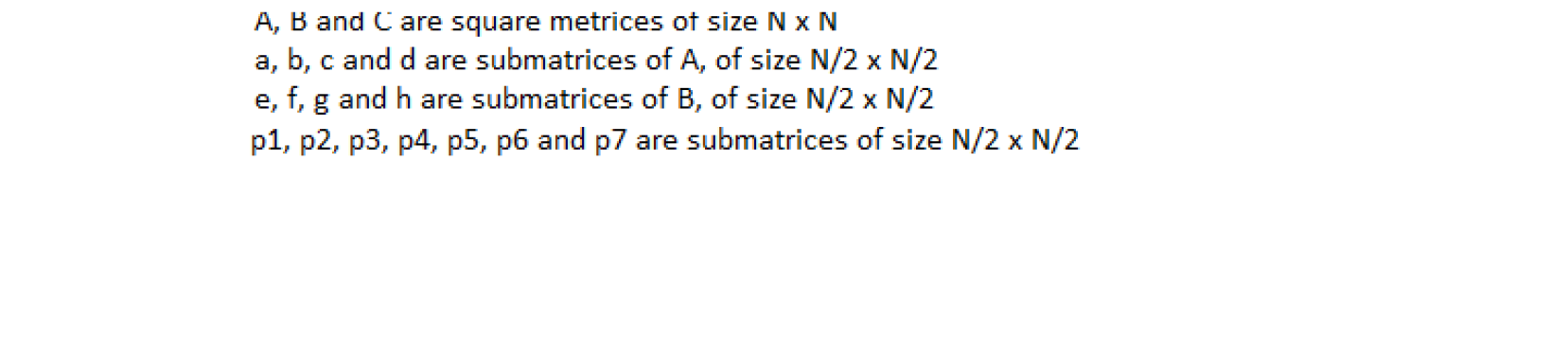
**Design and analysis of algorithm Lab**

**Week 4 Assignment**

**Part 2: Divide and conquer strategy**



****

****

1. **Naive Method:**

Algorithm:

Algorithm matrix\_multiply(a,b,c) // a and b are input matrices of size n x n and c is the output matrix of size n x n

for i = 0 to n-1 do

for j = 0 to n-1 do

for k = 0 to n-1 do

c[i][j]= c[i][j] + a[i][k]\*[k][j]

end

end

end

**Code:**

#include<iostream>

using namespace std;

#include<vector>

void matrix\_multiply(int a[][2],int b[][2],int c[][2])

{

    for(int i=0;i<2;i++)

    {

        for(int j=0;j<2;j++)

        {

            for(int k=0;k<2;k++)

            {

                c[i][j]+=a[i][k]\*b[k][j];

            }

        }

    }

}

int main()

{

    int a[2][2]={{1,2},{3,2}},b[2][2]={{2,1},{4,5}},c[2][2]={0};

    matrix\_multiply(a,b,c);

    for(int i=0;i<2;i++)

    {

        for(int j=0;j<2;j++)

        {

            cout<<c[i][j]<<" ";

        }

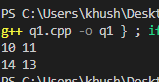
        cout<<endl;

    }

    return 0;

}

**Output:**

****

**Complexity Analysis:**

Time complexity: O(n3) where n is size of matrix

Space complexity: O(n2) As extra 2D array is used to store result

1. **Divide and Conquer Method:**

Algorithm:

Algorithm multiply\_matrix(A, B, n) //A and B are input matrices and n is size of matrices

if(n<=2)

c11= (a11\*b11) + (a12\*b21)

c12= (a11\*b12) + (a12\*b22)

c21= (a21\*b11) + (a22\*b21)

c22= (a21\*b12) + (a22\*b22)

else

multiply\_matrix(A11, B11, n/2) + (A12, B21, n/2)

multiply\_matrix(A11, B12, n/2) + (A12, B22, n/2)

multiply\_matrix(A21, B11, n/2) + (A22, B21, n/2)

multiply\_matrix(A21, B12, n/2) + (A22, B22, n/2)

end

**Code:**

#include <iostream>

using namespace std;

#include<iomanip>

#include<vector>

void add\_matrix(vector<vector<int>> A,vector<vector<int>> B,vector<vector<int>>& C,int split\_index)

{

    for (auto i = 0; i < split\_index; i++)

        for (auto j = 0; j < split\_index; j++)

            C[i][j]= A[i][j] + B[i][j];

}

vector<vector<int>>multiply\_matrix(vector<vector<int>> A,vector<vector<int>> B)

{

    int col\_1 = A[0].size();

    int row\_1 = A.size();

    int col\_2 = B[0].size();

    int row\_2 = B.size();

    if (col\_1 != row\_2) {

        cout << "\nError: The number of columns in Matrix "

                "A must be equal to the number of rows in "

                "Matrix B\n";

        return {};

    }

    vector<int> row(col\_2, 0);

    vector<vector<int>> ans(row\_1,row);

    if (col\_1 == 1) ans[0][0]= A[0][0] \* B[0][0];

    else

    {

        int split\_index = col\_1 / 2;

        vector<int> row\_vector(split\_index, 0);

        vector<vector<int>> ans\_00(split\_index,row\_vector);

        vector<vector<int>> ans\_01(split\_index,row\_vector);

        vector<vector<int>> ans\_10(split\_index,row\_vector);

        vector<vector<int>> ans\_11(split\_index,row\_vector);

        vector<vector<int>> a00(split\_index, row\_vector);

        vector<vector<int>> a01(split\_index, row\_vector);

        vector<vector<int>> a10(split\_index, row\_vector);

        vector<vector<int>> a11(split\_index, row\_vector);

        vector<vector<int>> b00(split\_index, row\_vector);

        vector<vector<int>> b01(split\_index, row\_vector);

        vector<vector<int>> b10(split\_index, row\_vector);

        vector<vector<int>> b11(split\_index, row\_vector);

        for (auto i = 0; i < split\_index; i++)

            for (auto j = 0; j < split\_index; j++) {

                a00[i][j] = A[i][j];

                a01[i][j] = A[i][j + split\_index];

                a10[i][j] = A[split\_index + i][j];

                a11[i][j] = A[i + split\_index][j + split\_index];

                b00[i][j] = B[i][j];

                b01[i][j] = B[i][j + split\_index];

                b10[i][j] = B[split\_index + i][j];

                b11[i][j] = B[i + split\_index][j + split\_index];

            }

        add\_matrix(multiply\_matrix(a00, b00),multiply\_matrix(a01, b10),ans\_00, split\_index);

        add\_matrix(multiply\_matrix(a00, b01),multiply\_matrix(a01, b11),ans\_01, split\_index);

        add\_matrix(multiply\_matrix(a10, b00),multiply\_matrix(a11, b10),ans\_10, split\_index);

        add\_matrix(multiply\_matrix(a10, b01),multiply\_matrix(a11, b11),ans\_11, split\_index);

        for (auto i = 0; i < split\_index; i++)

            for (auto j = 0; j < split\_index; j++) {

                ans[i][j]= ans\_00[i][j];

                ans[i][j + split\_index]= ans\_01[i][j];

                ans[split\_index + i][j]= ans\_10[i][j];

                ans[i + split\_index][j + split\_index]= ans\_11[i][j];

            }

        ans\_00.clear();

        ans\_01.clear();

        ans\_10.clear();

        ans\_11.clear();

        a00.clear();

        a01.clear();

        a10.clear();

        a11.clear();

        b00.clear();

        b01.clear();

        b10.clear();

        b11.clear();

    }

    return ans;

}

int main()

{

    vector<vector<int> > A = {{ 1, 1, 0, 1 },

                              { 2, 5, 2, 6 },

                              { 3, 3, 4, 3 },

                              { 2, 0, 1, 4 }};

    vector<vector<int> > B = {{ 0, 1, 1, 1 },

                              { 2, 6, 2, 3 },

                              { 3, 2, 3, 1 },

                              { 2, 1, 0, 2 }};

    vector<vector<int> > ans(multiply\_matrix(A, B));

    cout<<"Multiplication :"<<endl;

    for(int i=0;i<4;i++)

    {

        for(int j=0;j<4;j++)

        {

            cout<<setw(4)<<ans[i][j];

        }

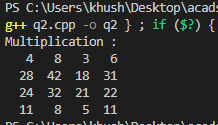
        cout<<endl;

    }

    return 0;

}

**Output:**

****

**Complexity Analysis:**

Time complexity: O(n3)

T(n) = 8T(n/2) + O(n2)

T(n)= O(n3) …using Master’s theorem

Space complexity: O(n2)

1. **Strassen’s Matrix Multiplication:**

Algorithm:

Calculate

P=(A11 + A22)\*(B11+B22)

Q=(A21 + A22)\* B11

R= A11\*(B12 – B22)

S= A22\*(B21 – B11)

T=(A11 + A12)\*B22

U=(A21 - A11)\*(B11 + B12)

V=(A12 – A22)\*(B21 + B22)

Then,

C11= P + S – T + V

C12= R + T

C21= Q + S

C22= P + R – Q + U

**Code:**

#include<iostream>

using namespace std;

#include<iomanip>

#include<vector>

vector<vector<int>>add\_matrix(vector<vector<int>> A,vector<vector<int>> B, int split\_index,int multiplier = 1)

{

    for (auto i = 0; i < split\_index; i++)

        for (auto j = 0; j < split\_index; j++)

            A[i][j]= A[i][j]+ (multiplier \* B[i][j]);

    return A;

}

vector<vector<int>>multiply\_matrix(vector<vector<int>> A,vector<vector<int>> B)

{

    int col\_1 = A[0].size();

    int row\_1 = A.size();

    int col\_2 = B[0].size();

    int row\_2 = B.size();

    if (col\_1 != row\_2) {

        cout << "\nError: The number of columns in Matrix "

                "A must be equal to the number of rows in "

                "Matrix B\n";

        return {};

    }

    vector<int> result\_matrix\_row(col\_2, 0);

    vector<vector<int> > ans(row\_1,result\_matrix\_row);

    if (col\_1 == 1)

        ans[0][0]= A[0][0] \* B[0][0];

    else

    {

        int split\_index = col\_1 / 2;

        vector<int> row\_vector(split\_index, 0);

        vector<vector<int> > a00(split\_index, row\_vector);

        vector<vector<int> > a01(split\_index, row\_vector);

        vector<vector<int> > a10(split\_index, row\_vector);

        vector<vector<int> > a11(split\_index, row\_vector);

        vector<vector<int> > b00(split\_index, row\_vector);

        vector<vector<int> > b01(split\_index, row\_vector);

        vector<vector<int> > b10(split\_index, row\_vector);

        vector<vector<int> > b11(split\_index, row\_vector);

        for (auto i = 0; i < split\_index; i++)

            for (auto j = 0; j < split\_index; j++) {

                a00[i][j] = A[i][j];

                a01[i][j] = A[i][j + split\_index];

                a10[i][j] = A[split\_index + i][j];

                a11[i][j] = A[i + split\_index][j + split\_index];

                b00[i][j] = B[i][j];

                b01[i][j] = B[i][j + split\_index];

                b10[i][j] = B[split\_index + i][j];

                b11[i][j] = B[i + split\_index][j + split\_index];

            }

        vector<vector<int> > p(multiply\_matrix(a00, add\_matrix(b01, b11, split\_index, -1)));

        vector<vector<int> > q(multiply\_matrix(add\_matrix(a00, a01, split\_index), b11));

        vector<vector<int> > r(multiply\_matrix(add\_matrix(a10, a11, split\_index), b00));

        vector<vector<int> > s(multiply\_matrix(a11, add\_matrix(b10, b00, split\_index, -1)));

        vector<vector<int> > t(multiply\_matrix(add\_matrix(a00, a11, split\_index),add\_matrix(b00, b11, split\_index)));

        vector<vector<int> > u(multiply\_matrix(add\_matrix(a01, a11, split\_index, -1),add\_matrix(b10, b11, split\_index)));

        vector<vector<int> > v(multiply\_matrix(add\_matrix(a00, a10, split\_index, -1),add\_matrix(b00, b01, split\_index)));

        vector<vector<int> > result\_matrix\_00(add\_matrix(add\_matrix(add\_matrix(t, s, split\_index), u,split\_index),q, split\_index, -1));

        vector<vector<int> > result\_matrix\_01(add\_matrix(p, q, split\_index));

        vector<vector<int> > result\_matrix\_10(add\_matrix(r, s, split\_index));

        vector<vector<int> > result\_matrix\_11(add\_matrix(add\_matrix(add\_matrix(t, p, split\_index), r,split\_index, -1),v, split\_index, -1));

        for (auto i = 0; i < split\_index; i++)

            for (auto j = 0; j < split\_index; j++) {

                ans[i][j]= result\_matrix\_00[i][j];

                ans[i][j + split\_index]= result\_matrix\_01[i][j];

                ans[split\_index + i][j]= result\_matrix\_10[i][j];

                ans[i + split\_index][j + split\_index]= result\_matrix\_11[i][j];

            }

        a00.clear();

        a01.clear();

        a10.clear();

        a11.clear();

        b00.clear();

        b01.clear();

        b10.clear();

        b11.clear();

        p.clear();

        q.clear();

        r.clear();

        s.clear();

        t.clear();

        u.clear();

        v.clear();

        result\_matrix\_00.clear();

        result\_matrix\_01.clear();

        result\_matrix\_10.clear();

        result\_matrix\_11.clear();

    }

    return ans;

}

int main()

{

    vector<vector<int> > A = {{ 1, 1, 0, 1 },

                              { 2, 5, 2, 6 },

                              { 3, 4, 4, 3 },

                              { 2, 0, 1, 4 }};

    vector<vector<int> > B = {{ 0, 1, 1, 1 },

                              { 2, 6, 2, 3 },

                              { 3, 2, 3, 1 },

                              { 2, 1, 0, 2 }};

    vector<vector<int> > ans(multiply\_matrix(A, B));

    cout<<"Multiplication :"<<endl;

    for(int i=0;i<4;i++)

    {

        for(int j=0;j<4;j++)

        {

            cout<<setw(4)<<ans[i][j];

        }

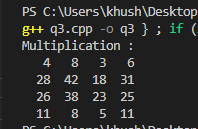
        cout<<endl;

    }

    return 0;

}

**Output:**

****

**Complexity Analysis:**

Time complexity: O(n2.81)

T(n) = 7T(n/2)+O(n2)

T(n)=O(nlog 7) …using Master’s theorem

Space complexity: O(n2)