Walchand College of Engineering, Sangli
Computer Science & Engineering
Third Year
Course: Design and analysis of algorithm Lab (3CS351)
Lab course coordinator:
Mrs A M Chimanna- Batch: - T1, T2, T3,T4

Week 4 Assignment

Part: 2

PRN : 21510111

Batch : T1

Divide and conquer strategy

Strassen's Matrix Multiplication

1) Implement Naive method multiply two matrices and Justify complexity is o(n3)

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

void multiply(vector<vector<int>>A, vector<vector<int>> B){
    vector<vector<int> > C = { { 0, 0, 0, 0 },
                               { 0, 0, 0, 0 },
                               { 0, 0, 0, 0 },
                               { 0, 0, 0, 0 }};

    for (int i = 0; i < 4; i++)
    {
        for (int j = 0; j < 4; j++)
        {
            C[i][j] = 0;
            for (int k = 0; k < 4; k++)
            {
                C[i][j] += A[i][k]*B[k][j];
            }
        }
    }

    for (int i = 0; i <= 3; i++) {
        for (int j = 0; j <= 3; j++) {
            cout << C[i][j] << " ";
        }
        cout << endl;
    }
    cout << endl;
}

int main(){
    vector<vector<int> > matrix_A = { { 1, 8, 1, 1 },
```

```
                                        { 2, 2, 0, 2 },
                                        { 3, 3, 3, 3 },
                                        { 2, 2, 2, 2 } };

    vector<vector<int> > matrix_B = { { 1, 1, 1, 1 },
                                      { 8, 2, 2, 2 },
                                      { 3, 3, 3, 3 },
                                      { 2, 2, 2, 2 } };

    multiply(matrix_A, matrix_B);

}
```

2) Implement Strassen's matrix multiplication for 3*3 matrix.
Do analysis of algorithm with respect to time complexity.

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

#define ROW_1 4
#define COL_1 4

#define ROW_2 4
#define COL_2 4

void print(string display, vector<vector<int> > matrix,
        int start_row, int start_column, int end_row,
        int end_column)
{
    cout << endl << display << " =>" << endl;
    for (int i = start_row; i <= end_row; i++) {
        for (int j = start_column; j <= end_column; j++) {
            cout << setw(10);
            cout << matrix[i][j];
        }
        cout << endl;
    }
    cout << endl;
    return;
}
```

```
vector<vector<int> >
add_matrix(vector<vector<int> > matrix_A,
        vector<vector<int> > matrix_B, int split_index,
        int multiplier = 1)
{
    for (auto i = 0; i < split_index; i++)
        for (auto j = 0; j < split_index; j++)
            matrix_A[i][j]
                = matrix_A[i][j]
                + (multiplier * matrix_B[i][j]);
    return matrix_A;
}

vector<vector<int> >
multiply_matrix(vector<vector<int> > matrix_A,
                vector<vector<int> > matrix_B)
{
    int col_1 = matrix_A[0].size();
    int row_1 = matrix_A.size();
    int col_2 = matrix_B[0].size();
    int row_2 = matrix_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> > result_matrix(row_1,
                                       result_matrix_row);

    if (col_1 == 1)
```

```
        result_matrix[0][0]
            = matrix_A[0][0] * matrix_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] = matrix_A[i][j];
                a01[i][j] = matrix_A[i][j + split_index];
                a10[i][j] = matrix_A[split_index + i][j];
                a11[i][j] = matrix_A[i + split_index]
                                    [j + split_index];
                b00[i][j] = matrix_B[i][j];
                b01[i][j] = matrix_B[i][j + split_index];
                b10[i][j] = matrix_B[split_index + i][j];
                b11[i][j] = matrix_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++) {
            result_matrix[i][j]
                = result_matrix_00[i][j];
            result_matrix[i][j + split_index]
                = result_matrix_01[i][j];
            result_matrix[split_index + i][j]
                = result_matrix_10[i][j];
            result_matrix[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 result_matrix;
}

int main()
{
    vector<vector<int> > matrix_A = { { 1, 1, 1, 1 },
                                     { 2, 2, 2, 2 },
                                     { 3, 3, 3, 3 },
                                     { 2, 2, 2, 2 } };

    print("Array A", matrix_A, 0, 0, ROW_1 - 1, COL_1 - 1);
```

```
    vector<vector<int> > matrix_B = { { 1, 1, 1, 1 },
                                      { 2, 2, 2, 2 },
                                      { 3, 3, 3, 3 },
                                      { 2, 2, 2, 2 } };

    print("Array B", matrix_B, 0, 0, ROW_2 - 1, COL_2 - 1);

    vector<vector<int> > result_matrix(
        multiply_matrix(matrix_A, matrix_B));

    print("Result Array", result_matrix, 0, 0, ROW_1 - 1,
        COL_2 - 1);
}
```