DAA – Lab5

Vishal Gauba
1410110501

Algorithm	**Run Time**
Normal Matrix Multiplication	0.54 seconds
Recursive Multiplication	0.3 seconds
Strassen's Multiplication	0.15 seconds

Source Code:

```
#include "stdio.h"
#include "stdlib.h"
#include "time.h"

int** allocM(int rows){
   int** matrix = (int**) malloc(sizeof(int*) * rows);
   int i;
   for(i=0; i<rows; i++){
      matrix[i] = (int*) malloc(sizeof(int) * rows);
   }
   return matrix;
}

void freeM(int** matrix, int rows){
   int i;
   for(i=0; i<rows; i++){
      free(matrix[i]);
   }
   free(matrix);
}

void prettyPrint(int** matrix, int rows, int columns){
   printf("\n");
   int i, j;
   for(i=0; i<rows; i++){
      for(j=0; j<columns; j++){
         printf("%d ", matrix[i][j]);
      }
      printf("\n");
   }
```

```
}

void addMatrices(int** matrix1, int** matrix2, int** result, int a1, int a2, int b1, int b2, int rows, int
sub){
   int i;
   for(int i=0; i<rows; i++){
      for(int j=0;j<rows;j++)
         if(sub == 0)
            result[i][j] = m1[a1 + i][a2 + j] + m2[b1 + i][b2 + j];
         else
            result[i][j] = m1[a1 + i][a2 + j] - m2[b1 + i][b2 + j];
   }
}

// void add(int** matrix1, int** matrix2, int** result, int rows){
//    int i, j;
//    for(i=0; i<rows; i++){
//       for(j=0; j<rows; j++){
//          result[i][j] = matrix1[i][j] + matrix2[i][j];
//       }
//    }
// }

void multiply(int** matrix1, int** matrix2, int** result, int rows){
   int i, j, k, sum = 0;
   for(i=0; i<rows; i++){
      for(j=0; j<rows; j++){
         for(k=0; k<rows; k++){
            sum = sum + matrix1[i][k]*matrix2[k][j];
         }
         result[i][j] = sum;
         sum = 0;
      }
   }
}




void recursive(int** a, int** b, int** r, int a1, int a2, int b1, int b2,int n){
   /*
      If n > 2
      Divide a into a11, a12, a21, a22 and b into b11, b12, b21, b22
      r11 = a11 * b11 + a12 * b21
      r12 = a11 * b12 + a12 * b22
      r21 = a21 * b11 + a22 * b21
      r22 = a21 * b12 + a22 * b22
```

```
   if(n > 2){
      int n2 = n/2;
      printf("n2 = %d\n\n", n2);
      int** a11b11 = createMtrx(n2, n2);
      recursive(a,b,a11b11,a1,a2,b1,b2,n2);
//       prnMtrx(a11b11, n2, n2);

      int** a12b21 = createMtrx(n2, n2);
      recursive(a,b,a12b21,a1,a2+(n2),b1+(n2),b2,n2);
      prnMtrx(a12b21, n2, n2);

      int** a11b12 = createMtrx(n2, n2);
      recursive(a,b,a11b12,a1,a2,b1,b2+n2,n2);
//       prnMtrx(a11b12, n2, n2);

      int** a12b22 = createMtrx(n2, n2);
      recursive(a,b,a12b22,0,n2,n2,n2,n2);

      int** a21b11 = createMtrx(n2, n2);
      recursive(a,b,a21b11,n2,0,0,0,n2);

      int** a22b21 = createMtrx(n2, n2);
      recursive(a,b,a22b21,n2,n2,n2,0,n2);

      int** a21b12 = createMtrx(n2, n2);
      recursive(a,b,a21b12,n2,0,0,n2,n2);

      int** a22b22 = createMtrx(n2, n2);
      recursive(a,b,a22b22,n2,n2,n2,n2,n2);

      int** r11 = createMtrx(n2, n2);
      addMtrx(a11b11, a12b21, r11, n2);
      int** r12 = createMtrx(n2, n2);
      addMtrx(a11b12, a12b22, r12, n2);
      int** r21 = createMtrx(n2, n2);
      addMtrx(a21b11, a22b21, r21, n2);
      int** r22 = createMtrx(n2, n2);
      addMtrx(a21b12, a22b22, r22, n2);

      for(int i=0;i<n2;i++){
         for(int j=0;j<n2;j++){
            r[i][j] = r11[i][j];
         }
      }
      for(int i=0;i<n2;i++){
         for(int j=0;j<n2;j++){
            r[i][n2 + j] = r12[i][j];
         }
```

```
        }
        for(int i=0;i<n2;i++){
            for(int j=0;j<n2;j++){
                r[n2 + i][j] = r21[i][j];
            }
        }
        for(int i=0;i<n2;i++){
            for(int j=0;j<n2;j++){
                r[n2 + i][n2 + j] = r22[i][j];
            }
        }
        return;
    }
    // If n == 2
    r[0][0] = a[a1][a2]*b[b1][b2] + a[a1][a1 + 1]*b[b1 + 1][b2];
    r[0][1] = a[a1][a2]*b[b1][b2 + 1] + a[a1][a2 + 1]*b[b1 + 1][b2 + 1];
    r[1][0] = a[a1 + 1][a2]*b[b1][b2] + a[a1 + 1][a2 + 1]*b[b1 + 1][b2];
    r[1][1] = a[a1 + 1][a2]*b[b1][b2 + 1] + a[a1 + 1][a2 + 1]*b[b1 + 1][b2 + 1];
    //prnMtrx(r, n, n);
    return;
}

int main()
{
    int n=6;
    int** a = createMtrx(n,n);
    int** b = createMtrx(n,n);
    int** r = createMtrx(n,n);

    for(int i=0;i<n;i++){
        for(int j=0;j<n;j++){
            a[i][j] = i + j + 1;
            b[i][j] = i + j;
        }
    }

    prnMtrx(a,n,n);
    printf("Normal : \n");
    prnMtrx(r,n,n);
    printf("\nRecursive: \n");
    recursive(a, b, r, 0,0,0,0,n);
    prnMtrx(r, n,n);
    return 0;

}
```