Name - Raveet Kumar
Roll No. - 1804310039
Branch - Computer Science and Engineering
College - BIET, Jhansi

Experiment - 03

Aim-

We have to implement 3 Algorithms using divide and conquer Approach.

- *Merge sort*
- *Quick sort*
- *Strassens's Multiplication*

Tools & Language Used-

❏ Java - for coding the algorithm and calculating time

❏ Python - for plotting graphs using matplotlib module.

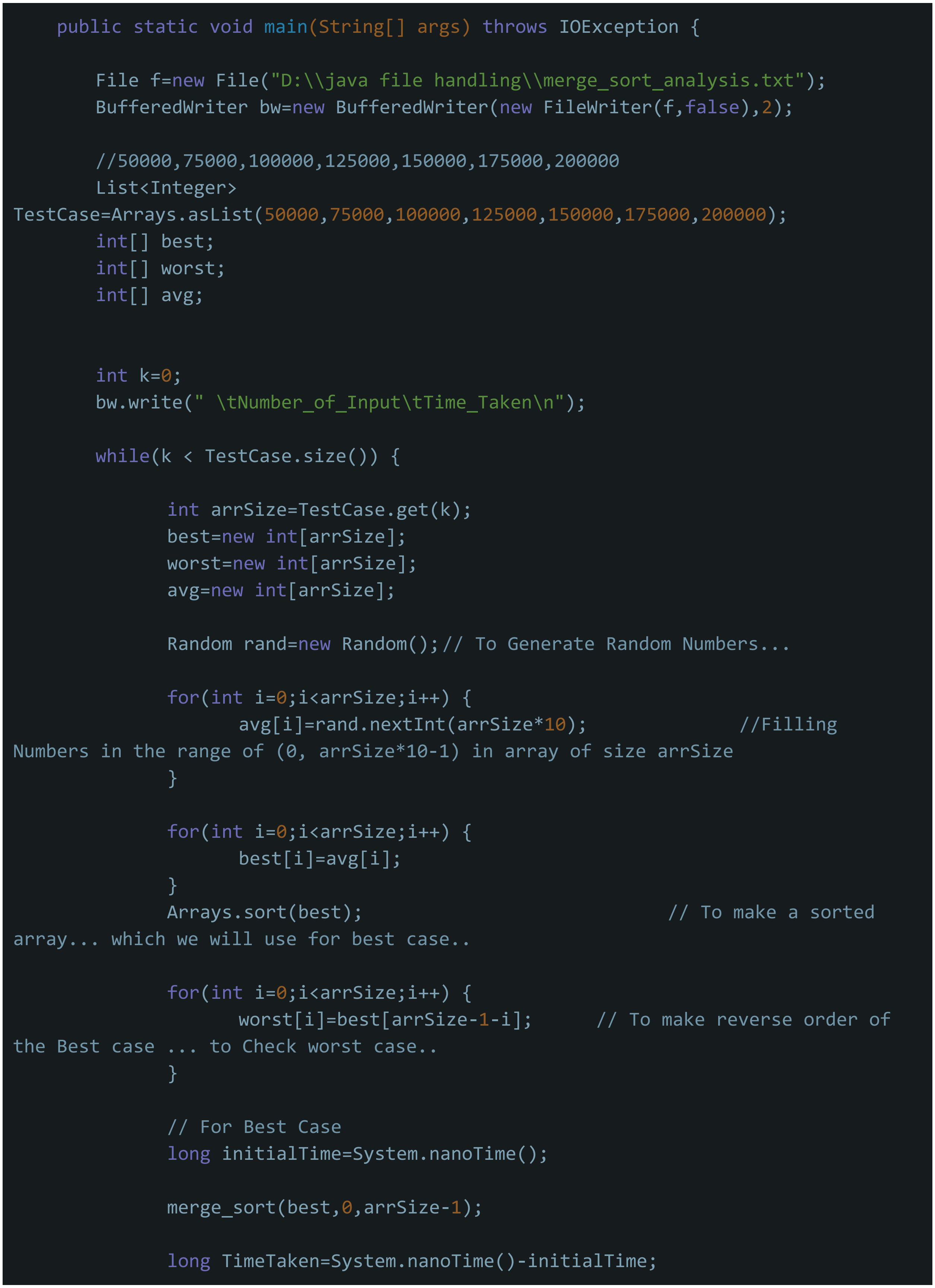
Code & Analysis-

1. Merge sort:

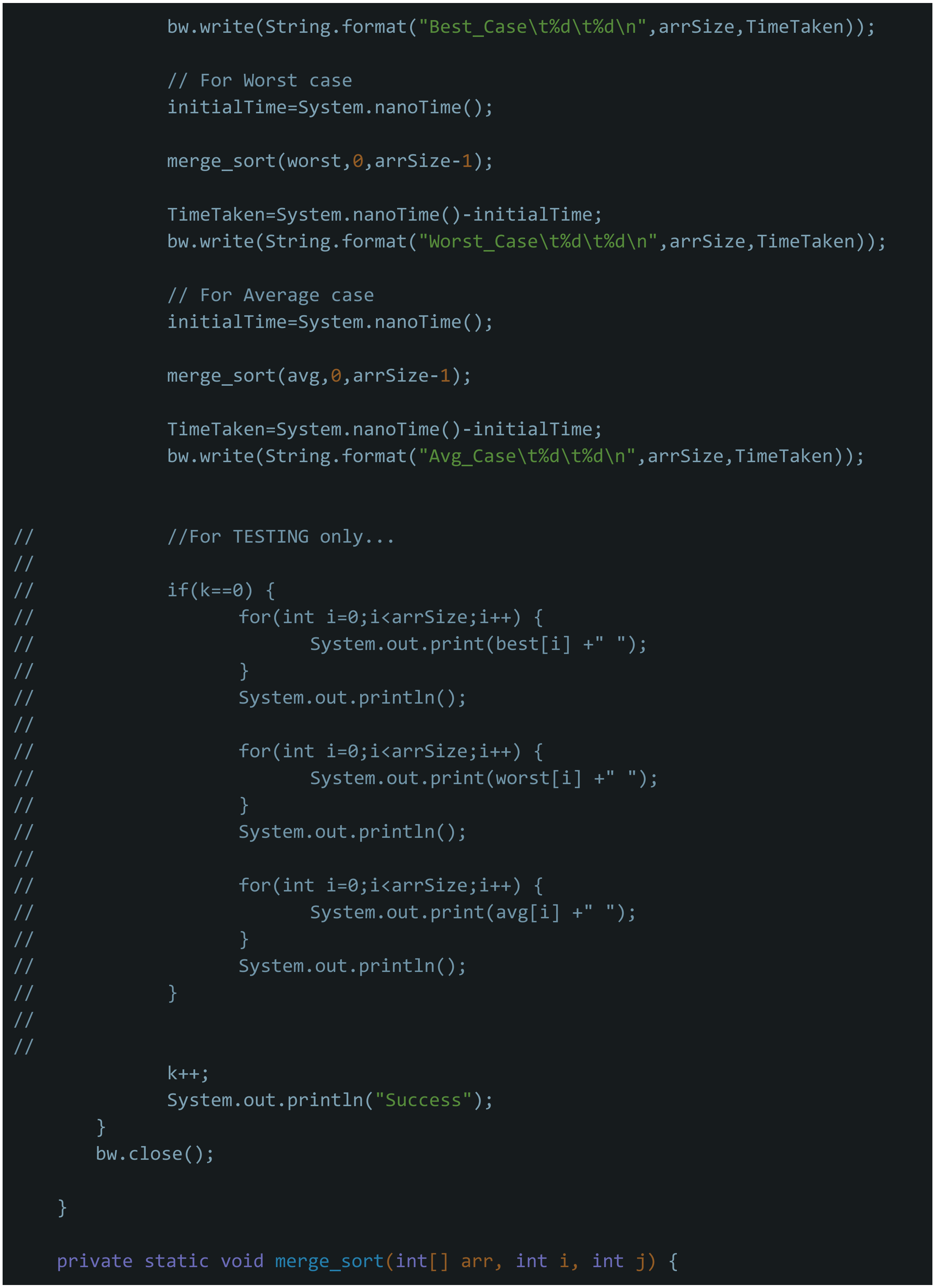
❏ It divides the input array in two halves, calls itself for the two halves and then merges the two sorted halves.

Code:

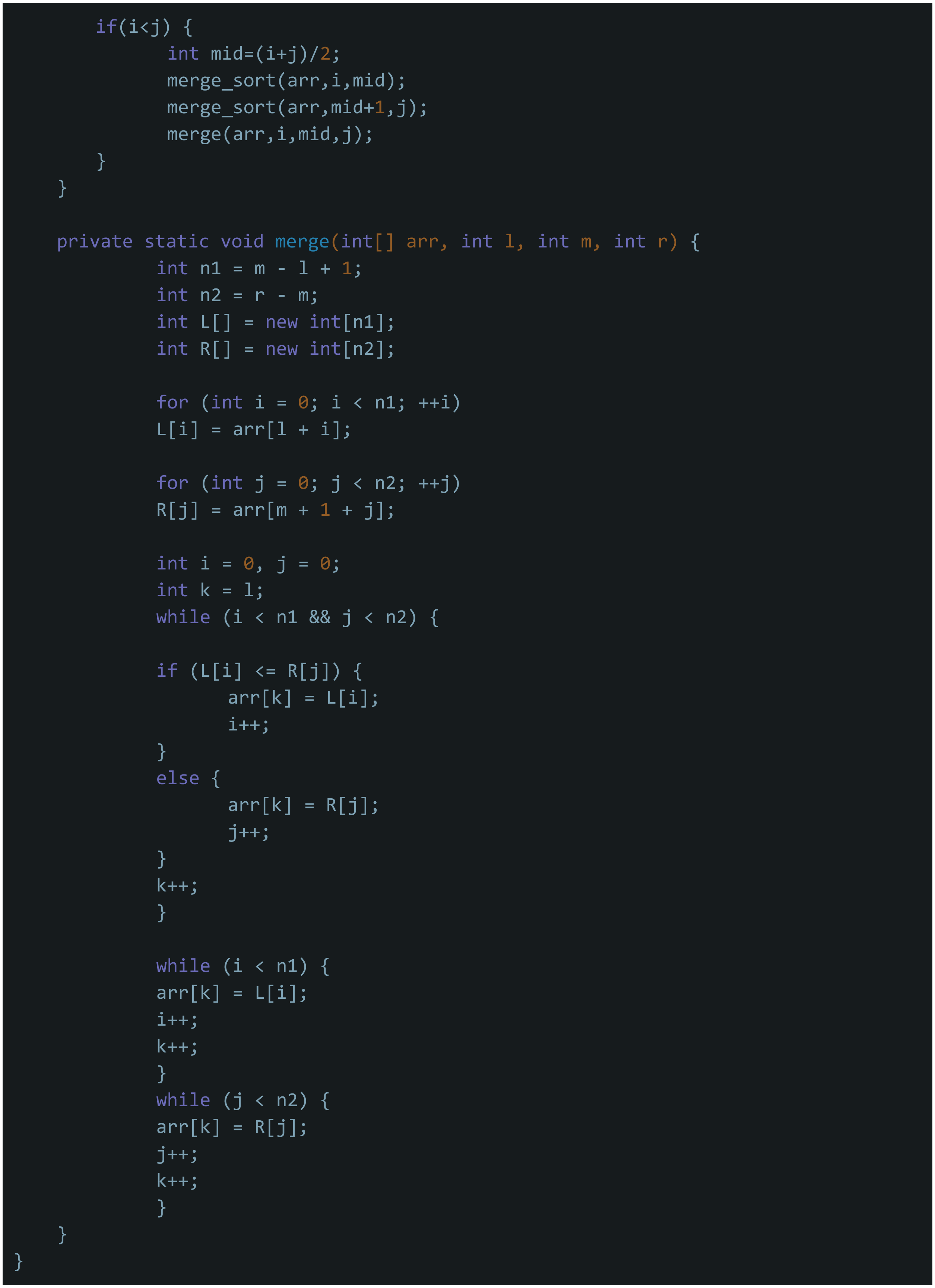
```java
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.Arrays;
import java.util.List;
import java.util.Random;

public class MergeSort {
```



```
    public static void main(String[] args) throws IOException {

        File f=new File("D:\\java file handling\\merge_sort_analysis.txt");
        BufferedWriter bw=new BufferedWriter(new FileWriter(f,false),2);

        //50000,75000,100000,125000,150000,175000,200000
        List<Integer>
TestCase=Arrays.asList(50000,75000,100000,125000,150000,175000,200000);
        int[] best;
        int[] worst;
        int[] avg;


        int k=0;
        bw.write(" \tNumber_of_Input\tTime_Taken\n");

        while(k < TestCase.size()) {

              int arrSize=TestCase.get(k);
              best=new int[arrSize];
              worst=new int[arrSize];
              avg=new int[arrSize];

              Random rand=new Random();// To Generate Random Numbers...

              for(int i=0;i<arrSize;i++) {
                    avg[i]=rand.nextInt(arrSize*10);              //Filling
Numbers in the range of (0, arrSize*10-1) in array of size arrSize
              }

              for(int i=0;i<arrSize;i++) {
                    best[i]=avg[i];
              }
              Arrays.sort(best);                           // To make a sorted
array... which we will use for best case..

              for(int i=0;i<arrSize;i++) {
                    worst[i]=best[arrSize-1-i];      // To make reverse order of
the Best case ... to Check worst case..
              }

              // For Best Case
              long initialTime=System.nanoTime();

              merge_sort(best,0,arrSize-1);

              long TimeTaken=System.nanoTime()-initialTime;
```

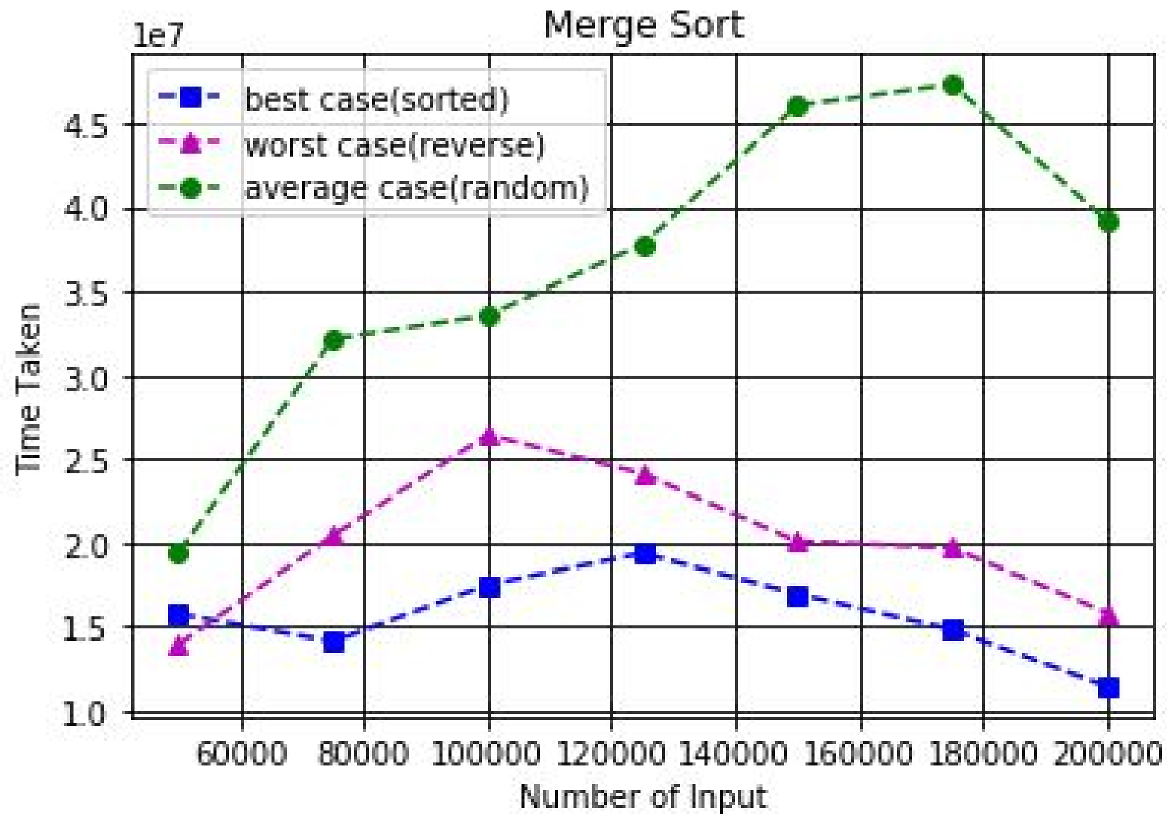


```
              bw.write(String.format("Best_Case\t%d\t%d\n",arrSize,TimeTaken));

              // For Worst case
              initialTime=System.nanoTime();

              merge_sort(worst,0,arrSize-1);

              TimeTaken=System.nanoTime()-initialTime;
              bw.write(String.format("Worst_Case\t%d\t%d\n",arrSize,TimeTaken));

              // For Average case
              initialTime=System.nanoTime();

              merge_sort(avg,0,arrSize-1);

              TimeTaken=System.nanoTime()-initialTime;
              bw.write(String.format("Avg_Case\t%d\t%d\n",arrSize,TimeTaken));


//            //For TESTING only...
//
//            if(k==0) {
//                  for(int i=0;i<arrSize;i++) {
//                        System.out.print(best[i] +" ");
//                  }
//                  System.out.println();
//
//                  for(int i=0;i<arrSize;i++) {
//                        System.out.print(worst[i] +" ");
//                  }
//                  System.out.println();
//
//                  for(int i=0;i<arrSize;i++) {
//                        System.out.print(avg[i] +" ");
//                  }
//                  System.out.println();
//            }
//
//

              k++;
              System.out.println("Success");
        }
        bw.close();

    }

    private static void merge_sort(int[] arr, int i, int j) {
```



```
        if(i<j) {
              int mid=(i+j)/2;
              merge_sort(arr,i,mid);
              merge_sort(arr,mid+1,j);
              merge(arr,i,mid,j);
        }
    }

    private static void merge(int[] arr, int l, int m, int r) {
             int n1 = m - l + 1;
             int n2 = r - m;
             int L[] = new int[n1];
             int R[] = new int[n2];

             for (int i = 0; i < n1; ++i)
             L[i] = arr[l + i];

             for (int j = 0; j < n2; ++j)
             R[j] = arr[m + 1 + j];

             int i = 0, j = 0;
             int k = l;
             while (i < n1 && j < n2) {

             if (L[i] <= R[j]) {
                    arr[k] = L[i];
                    i++;
             }
             else {
                    arr[k] = R[j];
                    j++;
             }
             k++;
             }

             while (i < n1) {
             arr[k] = L[i];
             i++;
             k++;
             }
             while (j < n2) {
             arr[k] = R[j];
             j++;
             k++;
             }
    }
}
```

Graph:

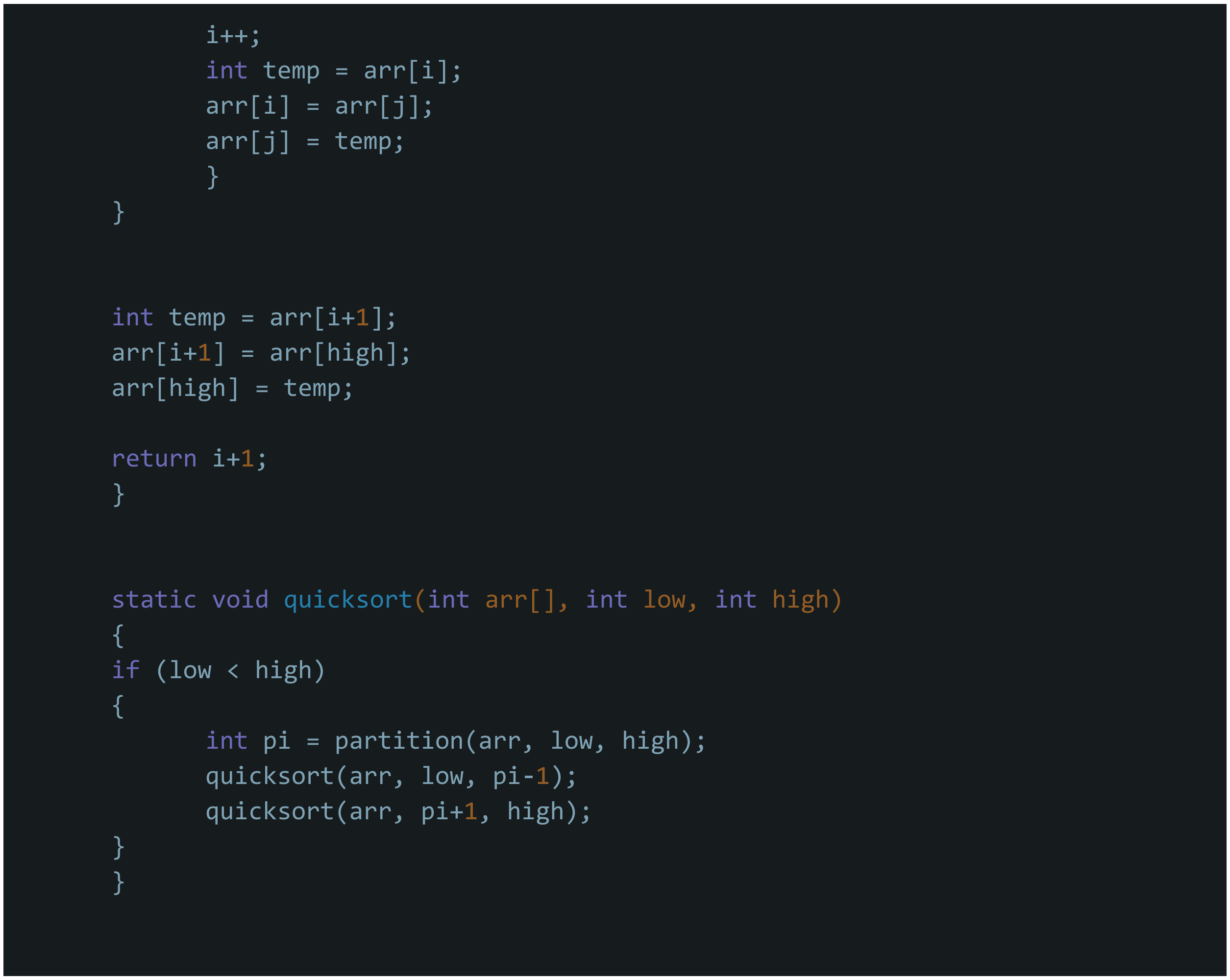


2. Quick Sort

- ❏ QuickSort is a Divide and Conquer algorithm. It picks an element as pivot and partitions the given array around the picked pivot.

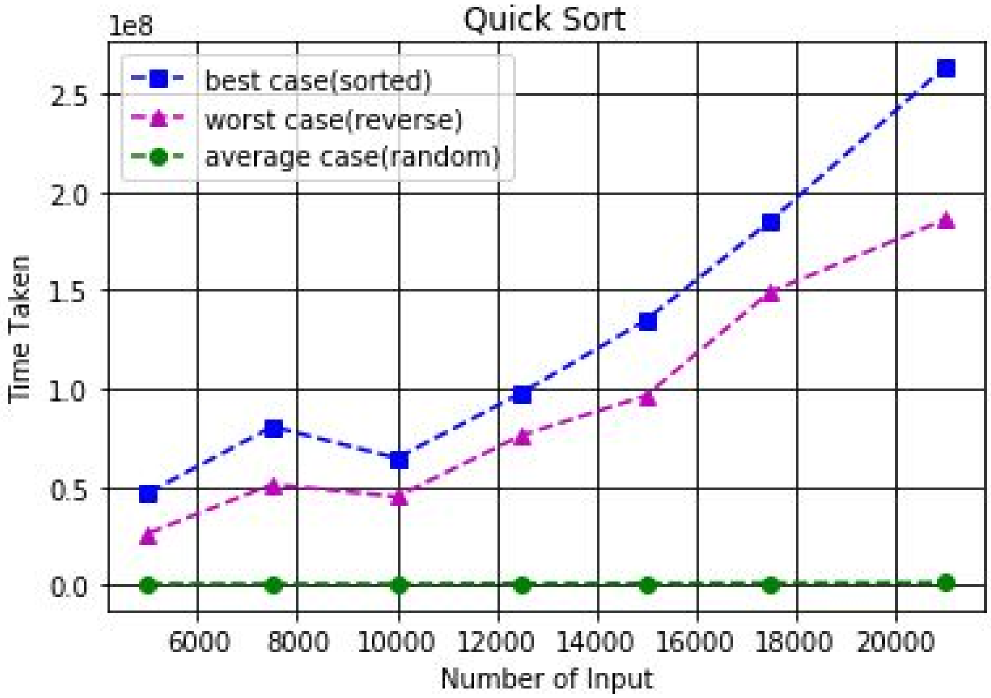
Code:

```
static int partition(int arr[], int low, int high)
       {
       int pivot = arr[high];
       int i = (low-1);
       for (int j=low; j<high; j++)
       {
              if (arr[j] < pivot)
              {
```



```
        i++;
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
        }
}


int temp = arr[i+1];
arr[i+1] = arr[high];
arr[high] = temp;

return i+1;
}


static void quicksort(int arr[], int low, int high)
{
if (low < high)
{
        int pi = partition(arr, low, high);
        quicksort(arr, low, pi-1);
        quicksort(arr, pi+1, high);
}
}
```

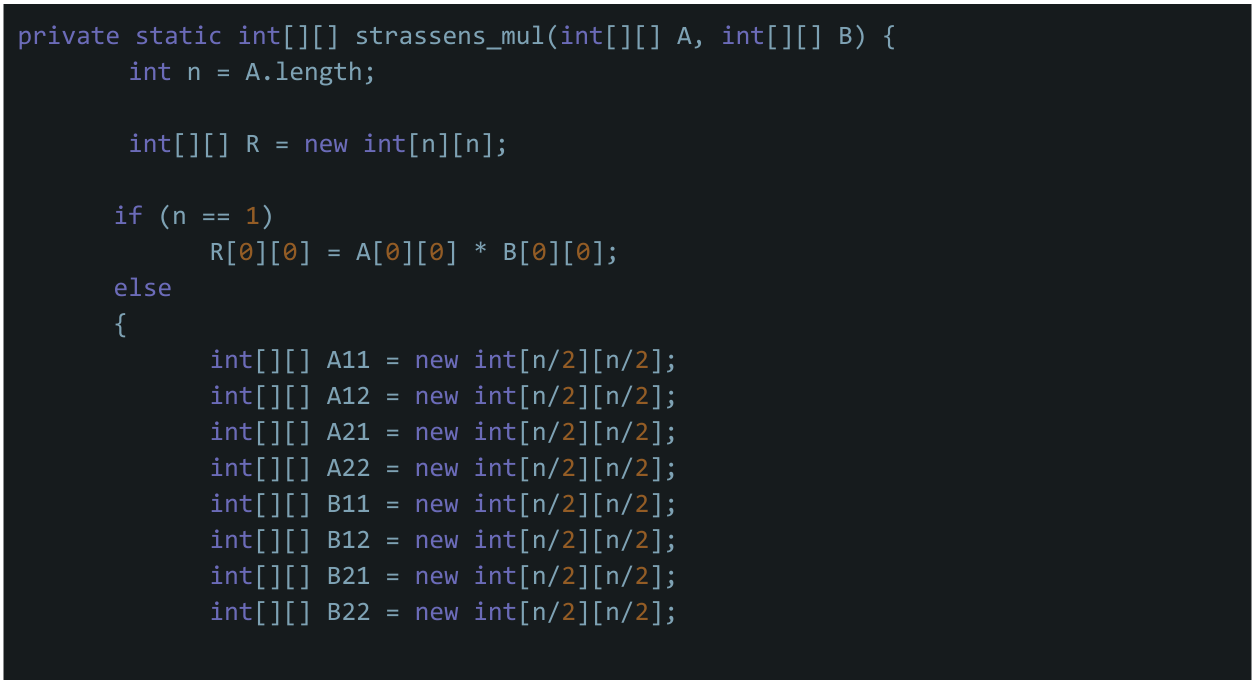
Graph:



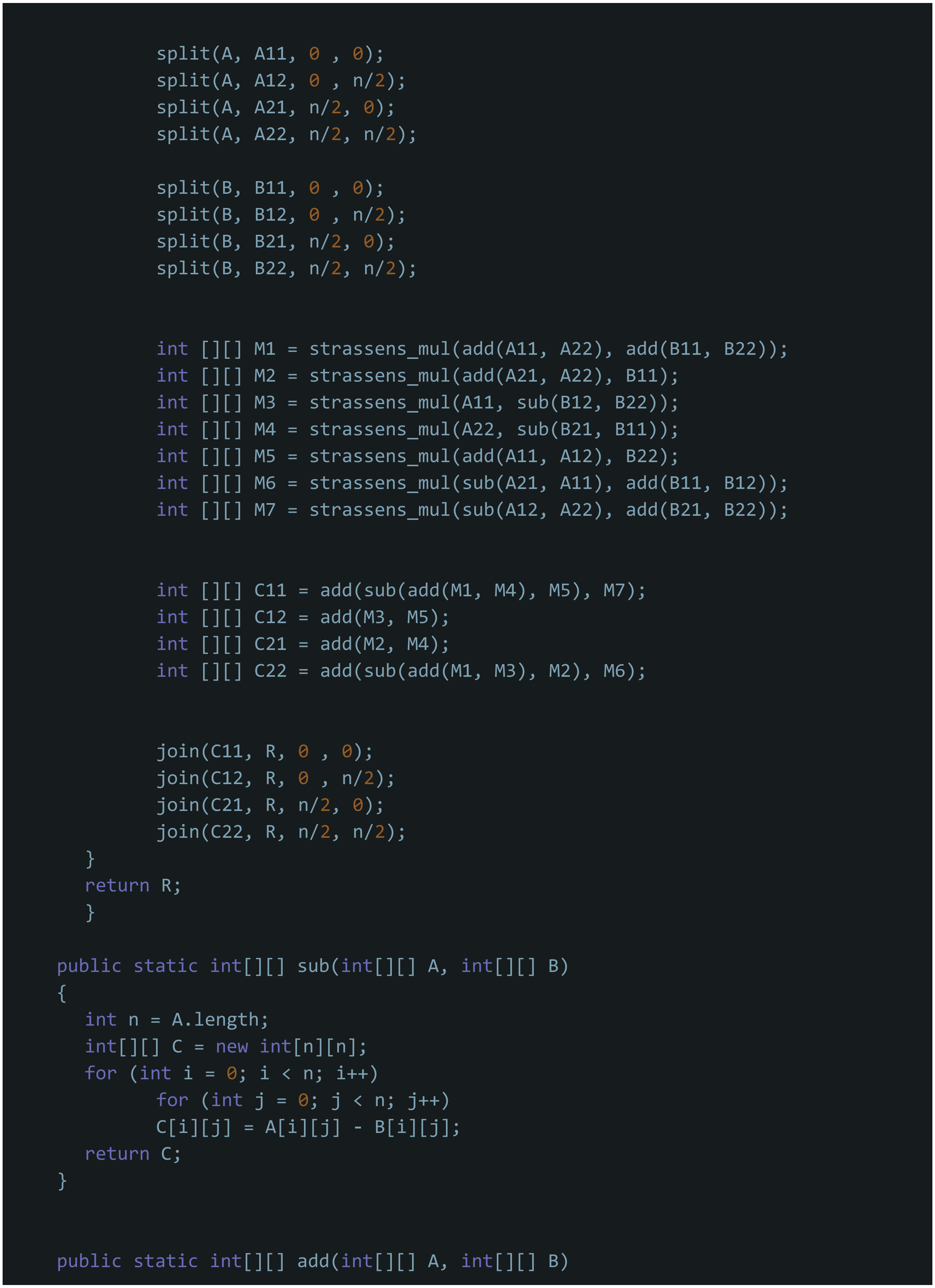
3. Strassen's Multiplication:

- Strassen’s Matrix multiplication can be performed only on square matrices where n is a power of 2. Order of both of the matrices are n × n.

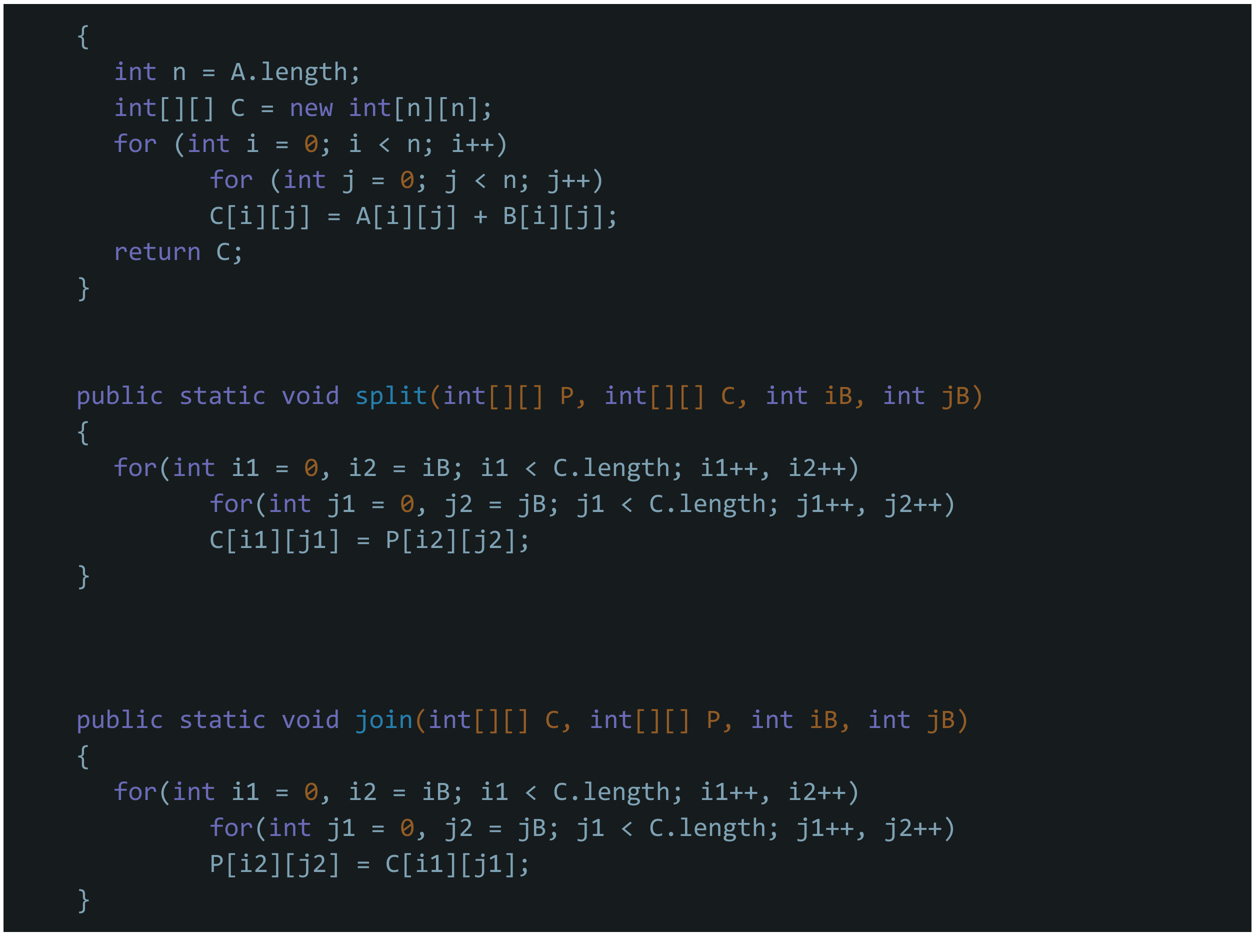
Code:



```
private static int[][] strassens_mul(int[][] A, int[][] B) {
        int n = A.length;

        int[][] R = new int[n][n];

       if (n == 1)
             R[0][0] = A[0][0] * B[0][0];
       else
       {
             int[][] A11 = new int[n/2][n/2];
             int[][] A12 = new int[n/2][n/2];
             int[][] A21 = new int[n/2][n/2];
             int[][] A22 = new int[n/2][n/2];
             int[][] B11 = new int[n/2][n/2];
             int[][] B12 = new int[n/2][n/2];
             int[][] B21 = new int[n/2][n/2];
             int[][] B22 = new int[n/2][n/2];
```

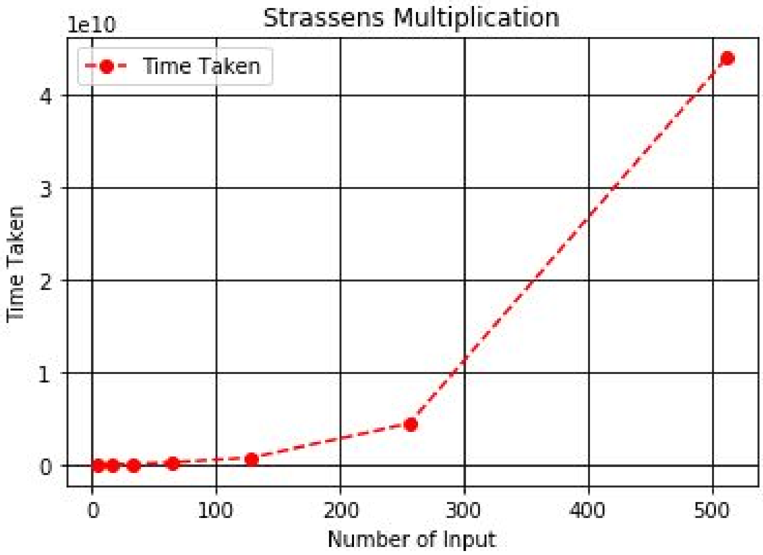


```
         split(A, A11, 0 , 0);
         split(A, A12, 0 , n/2);
         split(A, A21, n/2, 0);
         split(A, A22, n/2, n/2);

         split(B, B11, 0 , 0);
         split(B, B12, 0 , n/2);
         split(B, B21, n/2, 0);
         split(B, B22, n/2, n/2);


         int [][] M1 = strassens_mul(add(A11, A22), add(B11, B22));
         int [][] M2 = strassens_mul(add(A21, A22), B11);
         int [][] M3 = strassens_mul(A11, sub(B12, B22));
         int [][] M4 = strassens_mul(A22, sub(B21, B11));
         int [][] M5 = strassens_mul(add(A11, A12), B22);
         int [][] M6 = strassens_mul(sub(A21, A11), add(B11, B12));
         int [][] M7 = strassens_mul(sub(A12, A22), add(B21, B22));


         int [][] C11 = add(sub(add(M1, M4), M5), M7);
         int [][] C12 = add(M3, M5);
         int [][] C21 = add(M2, M4);
         int [][] C22 = add(sub(add(M1, M3), M2), M6);


         join(C11, R, 0 , 0);
         join(C12, R, 0 , n/2);
         join(C21, R, n/2, 0);
         join(C22, R, n/2, n/2);
   }
   return R;
   }

public static int[][] sub(int[][] A, int[][] B)
{
   int n = A.length;
   int[][] C = new int[n][n];
   for (int i = 0; i < n; i++)
         for (int j = 0; j < n; j++)
         C[i][j] = A[i][j] - B[i][j];
   return C;
}


public static int[][] add(int[][] A, int[][] B)
```



```
{
  int n = A.length;
  int[][] C = new int[n][n];
  for (int i = 0; i < n; i++)
        for (int j = 0; j < n; j++)
        C[i][j] = A[i][j] + B[i][j];
  return C;
}


public static void split(int[][] P, int[][] C, int iB, int jB)
{
  for(int i1 = 0, i2 = iB; i1 < C.length; i1++, i2++)
        for(int j1 = 0, j2 = jB; j1 < C.length; j1++, j2++)
        C[i1][j1] = P[i2][j2];
}



public static void join(int[][] C, int[][] P, int iB, int jB)
{
  for(int i1 = 0, i2 = iB; i1 < C.length; i1++, i2++)
        for(int j1 = 0, j2 = jB; j1 < C.length; j1++, j2++)
        P[i2][j2] = C[i1][j1];
}
```

Graph:



---------------------------------*------------------*----------------*-------------*