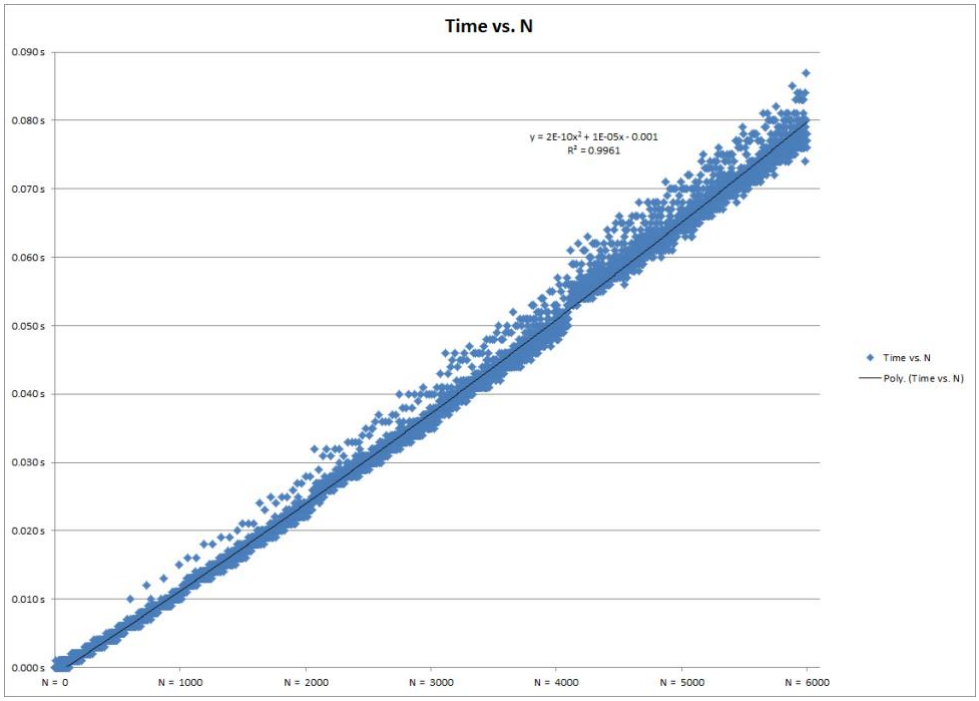
ALGORITHM AND GRAPH FOR MERGE SORT:

```
class MergeSort
{
	// Merges two subarrays of arr[].
	// First subarray is arr[l..m]
	// Second subarray is arr[m+1..r]
	void merge(int arr[], int l, int m, int r)
	{
		// Find sizes of two subarrays to be merged
		int n1 = m - l + 1;
		int n2 = r - m;

		/* Create temp arrays */
		int L[] = new int [n1];
		int R[] = new int [n2];

		/*Copy data to temp arrays*/
		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];


		/* Merge the temp arrays */

		// Initial indexes of first and second subarrays
		int i = 0, j = 0;

		// Initial index of merged subarry array
		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++;
		}

		/* Copy remaining elements of L[] if any */
		while (i < n1)
		{
			arr[k] = L[i];
			i++;
			k++;
		}

		/* Copy remaining elements of R[] if any */
		while (j < n2)
		{
			arr[k] = R[j];
```

```
                j++;
                k++;
        }
    }

    // Main function that sorts arr[l..r] using
    // merge()
    void sort(int arr[], int l, int r)
    {
        if (l < r)
        {
            // Find the middle point
            int m = (l+r)/2;

            // Sort first and second halves
            sort(arr, l, m);
            sort(arr , m+1, r);

            // Merge the sorted halves
            merge(arr, l, m, r);
        }
    }

    /* A utility function to print array of size n */
    static void printArray(int arr[])
    {
        int n = arr.length;
        for (int i=0; i<n; ++i)
            System.out.print(arr[i] + " ");
        System.out.println();
    }

    // Driver method
    public static void main(String args[])
    {
        int arr[] = {12, 11, 13, 5, 6, 7};

        System.out.println("Given Array");
        printArray(arr);

        MergeSort ob = new MergeSort();
        ob.sort(arr, 0, arr.length-1);

        System.out.println("\nSorted array");
        printArray(arr);
    }
}
```


Time vs. N
0.090 s
0.080 s
0.070 s
0.060 s
0.050 s
0.040 s
0.030 s
0.020 s
0.010 s
0.000 s
N = 0
N = 1000
N = 2000
N = 3000
N = 4000
N = 5000
N = 6000
y = 2E-10x² + 1E-05x - 0.001
R² = 0.9961
Time vs. N
Poly. (Time vs. N)