Aneesh Abhyankar
RUID - 166006555

Assignment 2

1) Implement the two versions of MergeSort that we discussed in class. Create a table or a plot for the total number of comparisons to sort the data (using data set here) for both cases. Explain.

Data and plot for Top-Down Merge sort is given below.

Data Size	**Number of Comparisons**	
	Unsorted Data	*Sorted Data*
1024	8954	5120
2048	19934	11264
4096	43944	24576
8192	96074	53248
16384	208695	114688
32768	450132	245760

Figure 1 – Number of comparisons as a function of Data Size in Top-Down Merge Sort

Data Size (N)	**Number of Comparisons (C)**	
	Unsorted Data	*Sorted Data*
1024	8954	5120
2048	19934	11264
4096	43944	24576
8192	96074	53248
16384	208695	114688
32768	450132	245760

Figure 2 – Number of comparisons as a function of Data Size in Bottom-Up Merge Sort

We see that the number of comparisons in case of Top-Down and Bottom-up Merge sort is the same. This is because we avoid dividing the array recursively, but instead we pass the sub arrays of doubling sizes starting from 1. This gives the same result as of dividing the array recursively, only that it is saving the overhead caused due to recursion. Plot of the above data is shown in the graph below.

Hypothesis => for Unsorted data

$C = a*N^b$

Therefore, $C1 = a*1024^b$ and $C2 = a*2048^b$

Solving, we get $b = 1.15$ and $a = 3.09$ (approximately)

Therefore, $\mathbf{C = 3.09*N^{1.15}}$

Similarly, for Sorted Data, we get

$$\mathbf{C = 2.03*N^{1.13}}$$

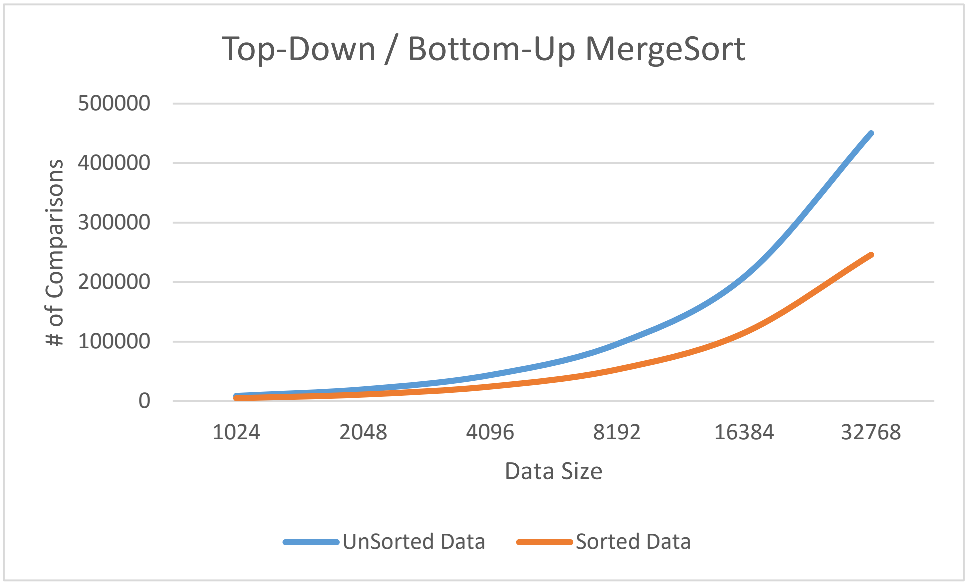

Figure 3 – Plot of Number of Comparisons against Data Size for both types of Merge Sort

2. Implement Shellsort which reverts to insertion sort. (Use the increment sequence 7, 3, 1). Similar to Q1, create a table or a plot for the total number of comparisons made in the sorting the data for both cases (insertion sort phase and shell sort phase). Explain why Shellshort is more effective than Insertion sort in this case.

Data Size	**Number of Comparisons**	
	Unsorted Data	*Sorted Data*
1024	46728	3061
2048	169042	6133
4096	660619	12277
8192	2576270	24565
16384	9950922	49141
32768	39442456	98293

Figure 4 – Number of comparisons as a function of Data Size in Shell Sort

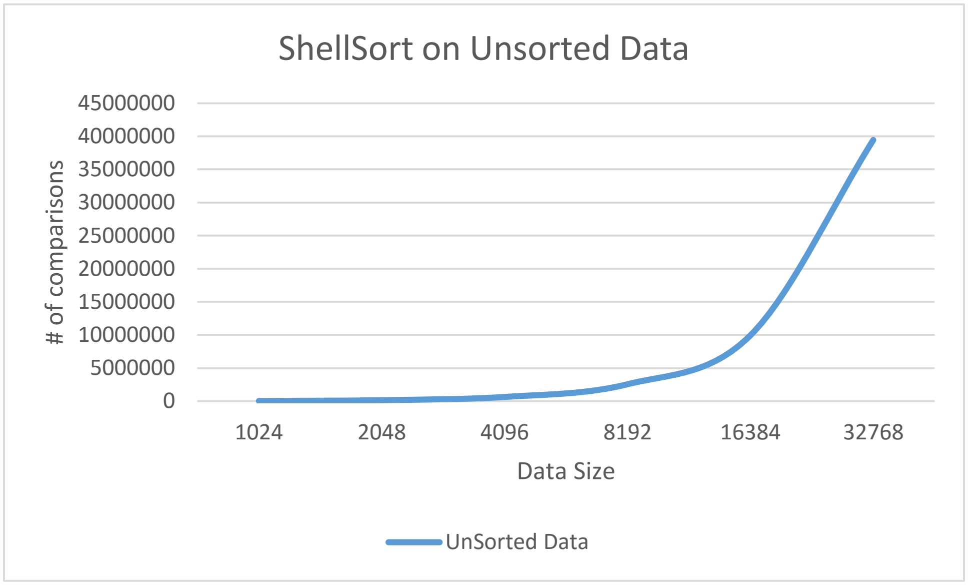

Figure 5 – Plot of Number of Comparisons against Data Size for Shell Sort for unsorted data

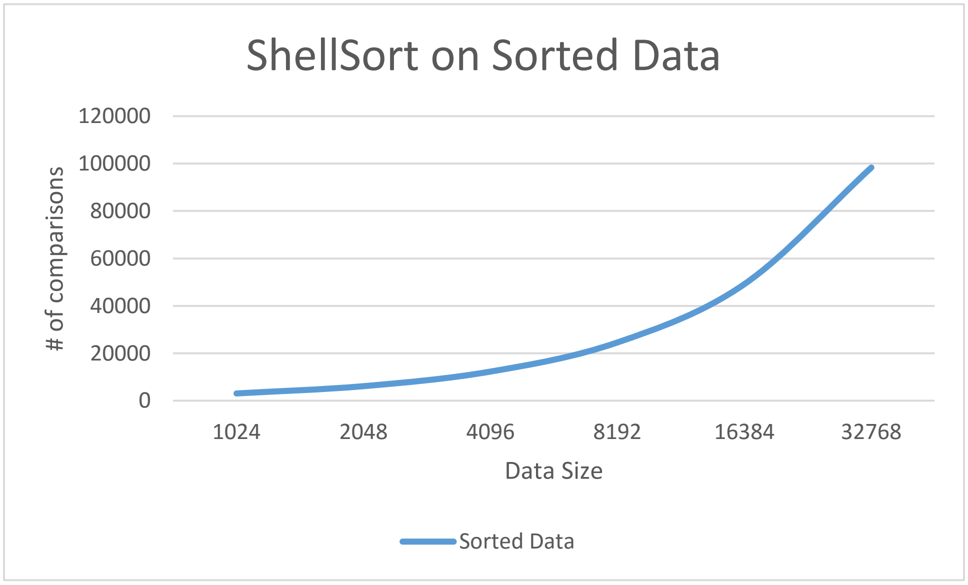

Figure 6 – Plot of Number of Comparisons against Data Size for Shell Sort for sorted data

In case of Shell Sort, we use insertion sort algorithm to work on different subsets of data. When h=7, we take every 7th element of the array and sort the subset of the array so obtained. We repeat this for h=3 and h=1. When h=1, the algorithm is nothing but insertion sort. However, by the time we reach h=1, we have decreased the number of inversions in the array to a very low level, so that insertion sort takes place in almost linear time. *When we run the same data on insertion sort, we get number of comparisons as 26553 and 1029278 for 1024 and 2048 data sizes respectively, which is much higher than shell sort.* Insertion sort shows worst case complexity of O(n^2), whereas Shell Sort is O($N^{1.5}$). That is why shell sort is better than insertion sort.

---

3). The Kendall Tau distance is a variant of the "number of inversions" we discussed in class. It is defined as the number of pairs that are in different order in two permutations. Write an efficient program that computes the Kendall Tau distance in less than quadratic time on average. Plot your results and discuss. Use the dataset provided here. Note: data0.* for convenience is an ordered set of numbers (in powers of two). data1.* are shuffled data sets of sizes (as given by "*").

I have used Merge Sort algorithm to calculate Kendall Tau distance in complexity O($N\log_2 N$). I increment the number of inversions counter when the element from the left subarray is greater than that from the right subarray.

Data Size (N)	**Number of Inversions (I)**
1024	264541
2048	1027236
4096	4183804
8192	16928767
16384	66641183
32768	267933908

Figure 7 – Number of inversions as a function of data size

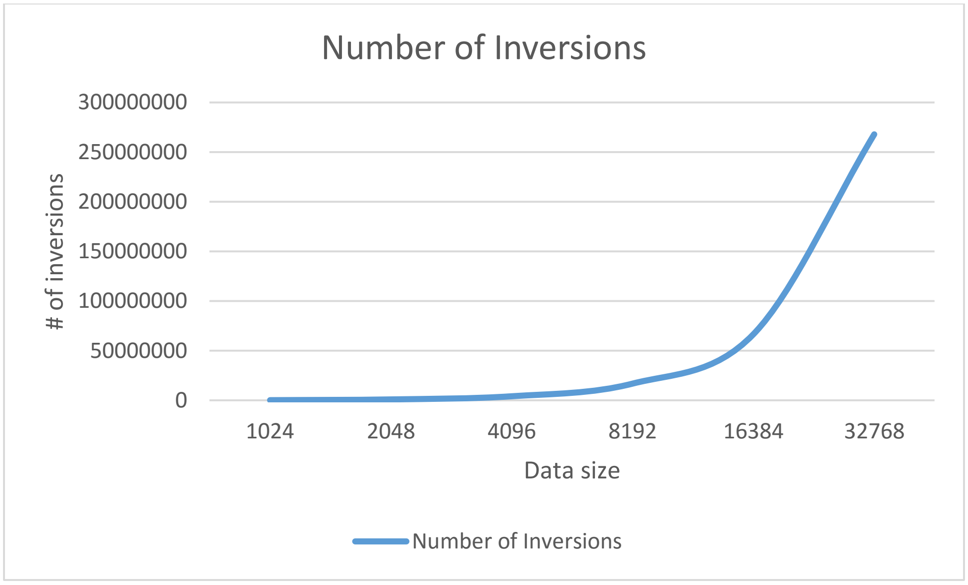

Figure 8 – Plot of Number of Inversions against Data size

Hypothesis => $I = a*N^b$

Therefore, $264541 = a*1024^b$ and $1027236 = a*2048^b$

Solving, we get b = 1.957 and a = 0.339

So, $\mathbf{I = 0.339*N^{1.957}}$

---

4) Create a data set of 8192 entries which has in the following order: 1024 repeats of 1, 2048 repeats of 11, 4096 repeats of 111 and 1024 repeats of 1111. Write a sort algorithm that you think will sort this set "most" effectively. Explain why you think so.

As the given data is already sorted, insertion sort works best on it. Insertion sort gives the best case complexity of O(N), which is the best among all other algorithms. Here, we are leveraging our knowledge about the data that it is already sorted and that it has many repeated values. If this fact is not known to us, we would have used mergesort which works with complexity of $O(N\log_2 N)$.
