Selin Dinç 150150229

Analysis of Algorithms
HW-1
Selin Dinç
150150229

a)

Bubble Sort Pseudo-code

```
length = size of my vector
cars = my vector

BubbleSort(vector)
For i←0 to length-1
        do for j←0 to length-i-1
                do if cars[j] > cars[j+1]
                        temp1← cars[j]
                        temp2 ← cars[j+1]
                        cars[j+1] ← temp1
                        cars[j] ← temp2
```

Bubble sort has an asymptotic bound which is O(n^2). In my code, first for calls the second for length-1 times. My second for loop iterates length-i-1 times to swap. The second loop has a cost of O(1). That is why my code for Bubble Sort algorithm has an asymptotic bound which is O(n^2).

Merge Sort Pseudo-code

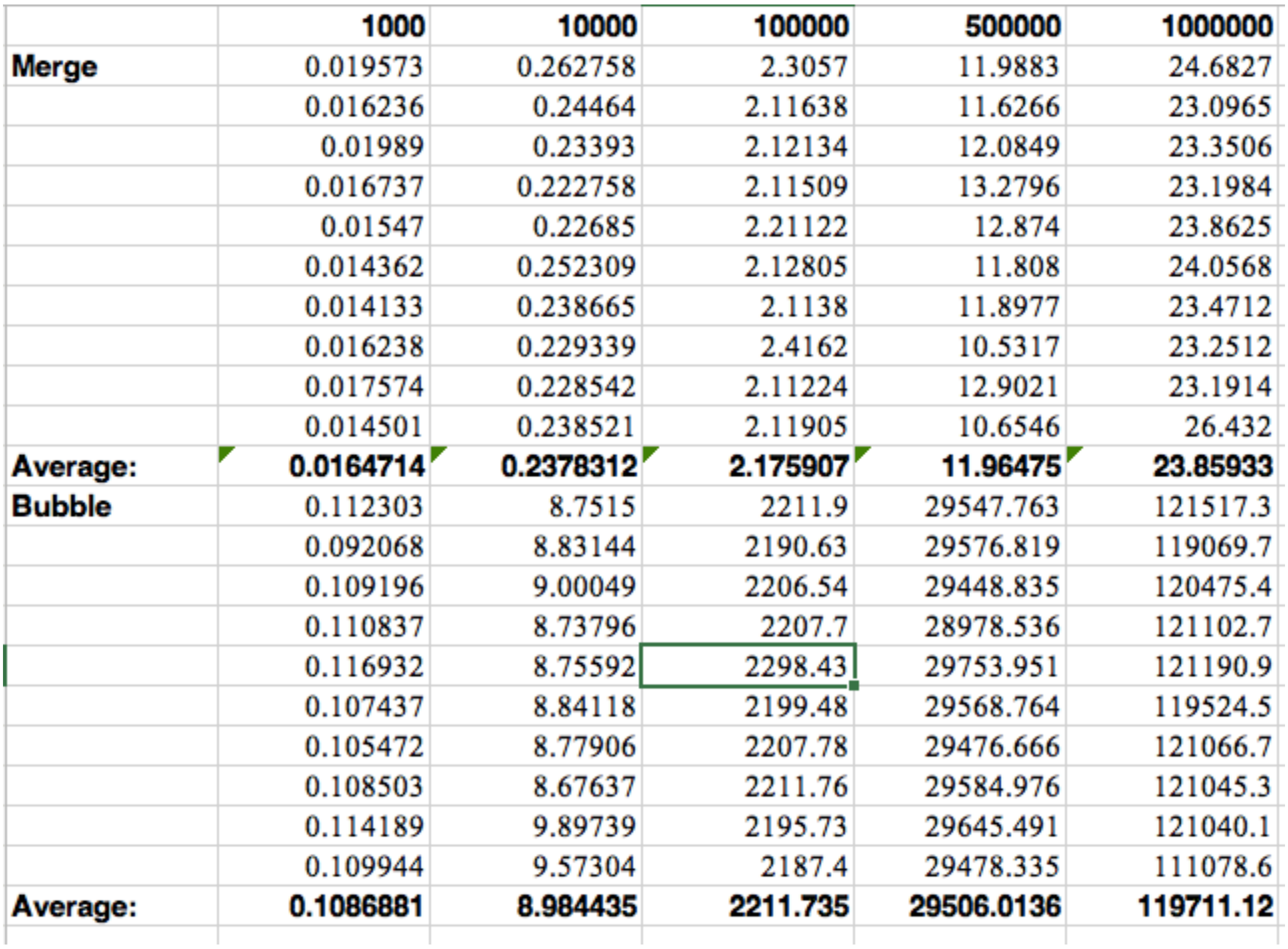
```
MergeSort(vector)

if length < 2
        returns cars
m← length/2
for i←0 to m
        push_back cars[i] to left
for j←m to length-1
        push_back cars[j] to right
temp1 ← MergeSort(left)
temp2 ← MergeSort(right)
temp3 ← Merge(temp1, temp2)

Merge(vector left,vector right)

i ← 0
j ← 0
temp1← vector
while i<sizeleft and j<sizeright
        do if left[i] <= right[j]

                i←i+1
        else

                j←j+1
while i<sizeleft
```

Selin Dinç 150150229

```
        push_back left[i] to temp1
        i←i+1
while j<sizeright
        push_back right [j] to temp1
        j←j+1
```

Merge Sort has an asymptotic bound of O(nlgn) for all best, average and worst cases. In my Merge function, I have three while loops((i+j) +(sizeleft-i)+(sizeright-j)) which iterates equals the total amount of O(sizeright+sizeleft). That makes an O(n). I'm calling the Merge function recursively at MergeSort function lgn times. That's make an O(nlgn). That is why my Merge Sort has an asymptotic bound of O(nlgn).

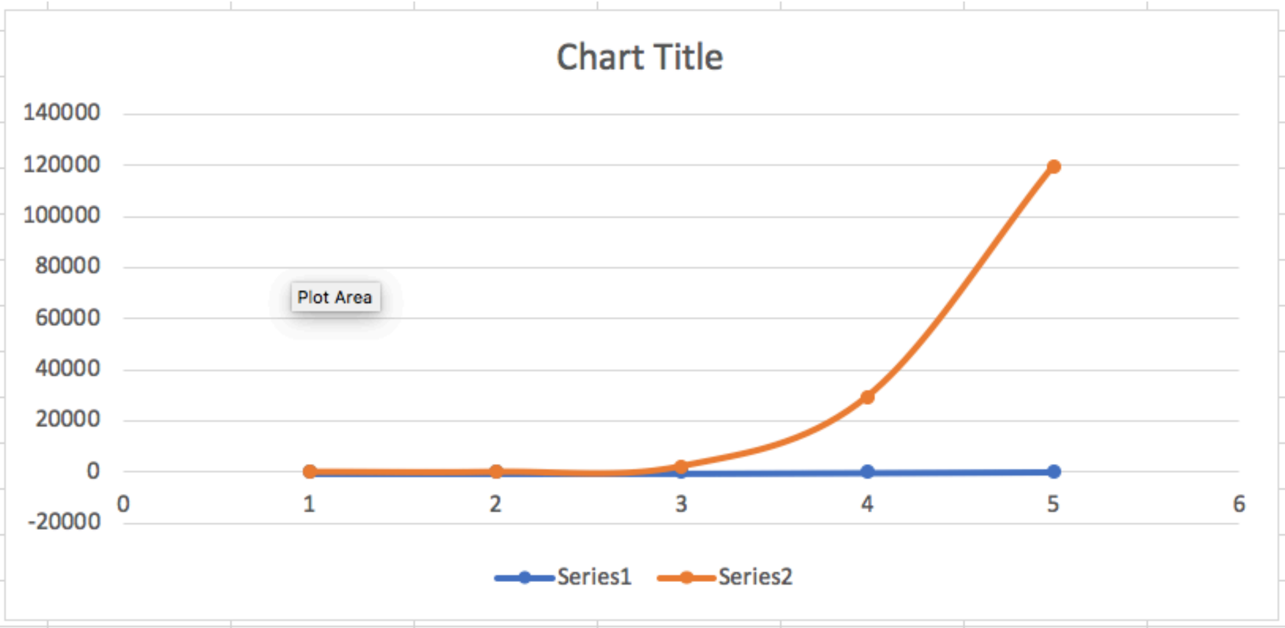
b)



	1000	**10000**	**100000**	**500000**	**1000000**
Merge	0.019573	0.262758	2.3057	11.9883	24.6827
	0.016236	0.24464	2.11638	11.6266	23.0965
	0.01989	0.23393	2.12134	12.0849	23.3506
	0.016737	0.222758	2.11509	13.2796	23.1984
	0.01547	0.22685	2.21122	12.874	23.8625
	0.014362	0.252309	2.12805	11.808	24.0568
	0.014133	0.238665	2.1138	11.8977	23.4712
	0.016238	0.229339	2.4162	10.5317	23.2512
	0.017574	0.228542	2.11224	12.9021	23.1914
	0.014501	0.238521	2.11905	10.6546	26.432
Average:	**0.0164714**	**0.2378312**	**2.175907**	**11.96475**	**23.85933**
Bubble	0.112303	8.7515	2211.9	29547.763	121517.3
	0.092068	8.83144	2190.63	29576.819	119069.7
	0.109196	9.00049	2206.54	29448.835	120475.4
	0.110837	8.73796	2207.7	28978.536	121102.7
	0.116932	8.75592	2298.43	29753.951	121190.9
	0.107437	8.84118	2199.48	29568.764	119524.5
	0.105472	8.77906	2207.78	29476.666	121066.7
	0.108503	8.67637	2211.76	29584.976	121045.3
	0.114189	9.89739	2195.73	29645.491	121040.1
	0.109944	9.57304	2187.4	29478.335	111078.6
Average:	**0.1086881**	**8.984435**	**2211.735**	**29506.0136**	**119711.12**

Selin Dinç 150150229

c)



Orange line: Bubble Sort
Blue line: Merge Sort
Y-axis: time
x-axis: N

In this plot, Bubble Sort has higher rate of increase than Merge Sort. This proves that, Bubble Sort's asymptotic bound which is O(n^2) in my case, has a faster growth rate than my Merge Sort's asymptotic bound which is O(nlgn). That is why Bubble Sort is a slower sorting algorithm than Merge Sort.