BLG335E

ANALYSIS OF ALGORITHMS

ASSIGNMENT 1 REPORT

Sinem Elif Haseki
150160026

Sinem Elif Haseki
150160026

(a)

Time complexity of Merge Sort functions:

	Statement	S/E	Frequency	Total Steps
1.	Algorithm MergeSort(A,p,r)	0	-	-
2.	if p < r then	1	1	1
3.	q=FLOOR[(p+r)/2]	1	1	1
4.	MergeSort(A,p,q)	T(n/2)	1	T(n/2)
5.	MergeSort(A,q+1,r)	T(n/2)	1	T(n/2)
6.	Merge(A,p,q,r)	O(n)	1	O(n)
Total (T(n))				2T(n/2)+O(n)+2

Table 1. MergeSort function time complexity table

	Statement	S/e	Frequency		Total Steps	
			if-true	if-false	if-true	if-false
1.	Algorithm Merge(a,low,mid,high)	0	-		-	
2.	k:=low;i:=low;j=mid+1;	3	1	1	1	1
3.	while((k<=mid)and(j<=high))do	1	n/2+1	n/2+1	n/2+1	n/2+1
4.	if(a[k]<=a[i]) then	1	n/2	n/2	n/2	n/2
5.	{b[i]:=a[k]; k:=k+1;}	2	n/2	0	n	0
6.	else					
7.	{b[i]:=a[j]; j:=j+1;}	2	0	n/2	0	n
8.	i:=i+1;	1	n/2	n/2	n/2	n/2
9.	if(k>mid) then		1	1	1	1
10.	for h:=j to high do	1	n/2+1	0	n/2+1	0
11.	{b[i]:=a[h]; i=i+1;}	2	n/2	0	n	0
12.	else					
13	for h:=k to mid do	1	0	n/2+1	0	n/2+1
14.	{b[i]:=a[h]; i=i+1;}	2	0	n/2	0	n
15.	for h:=low to high do	1	n+1	n+1	n+1	n+1
16.	a[h]:=b[h];	1	n	n	n	n
Total (T(n))					O(n)	Ω(n)

Table 2. Merge function time complexity table

- *MergeSort* function calls itself whilst sending only half elements of it, making its time complexity T(n/2) at those calls. According to Master Theorem, T(n) = 2T(n/2) + O(1) gives us the time complexity of O(nlogn), hence worst time complexity, i.e. asymptotic upper bound is O(nlogn).
- This algorithm also has the same time complexity for its best case, calling the *MergeSort* function gives us the best time complexity of Ω(nlogn), hence asymptotic lower bound of this algorithm is Ω(nlogn).
- In the *Merge* function, both the best and worst cases are in the order of n, hence asymptotic upper bound of *Merge* is O(n), whereas its asymptotic lower bound is Ω(n).

Sinem Elif Haseki
150160026

Time complexity of Bubble Sort function:

	Statement	S/e	Frequency		Total Steps	
			if-true	if-false	if-true	if-false
1.	Algorithm BubbleSort(…)	0	-		-	
2.	i ← length[A]	1	1	1	1	1
3.	sorted ← False	1	1	1	1	1
4.	while i>1 && sorted==False:	2	n	2	2n	4
5.	sorted ← True;	1	n-1	1	n-1	1
6.	for j=1 to i-1:	1	(n-1)*n	n	(n-1)*n	1
7.	if A[j] < A[j-1]:	1	(n-1)*n-1	n-1	(n-1)*n-1	n-1

8.	temp:=A[j-1];
	A[j-1]:=A[j];
	A[j]:=temp;
	sorted:=false;
		4	(n-1)*n-1	0	4*(n-1)*n-1	0

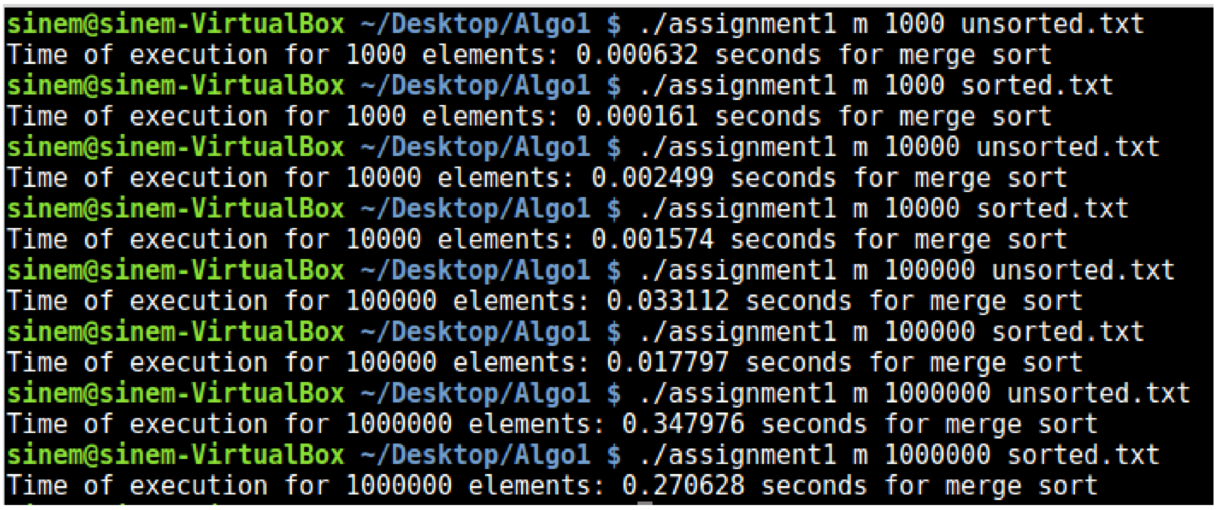
9.	i ← i-1		n-1	1	n-1	1
Total (T(n))					$6n^2$-3n-1	n+8

Table 3. Bubble sort function time complexity table

- *BubbleSort* function has the worst time complexity of T(n)= $6n^2$-3n-1, leading it to obtain an asymptotic upper bound of O(n^2). Also, best time complexity of this function is T(n)=n+8, hence asymptotic lower bound of *BubbleSort* is Ω(n).
- According to these results, when the array that the algorithm is working on is sorted (or almost sorted), time complexity is linear; otherwise it’s quadratic.

(b)

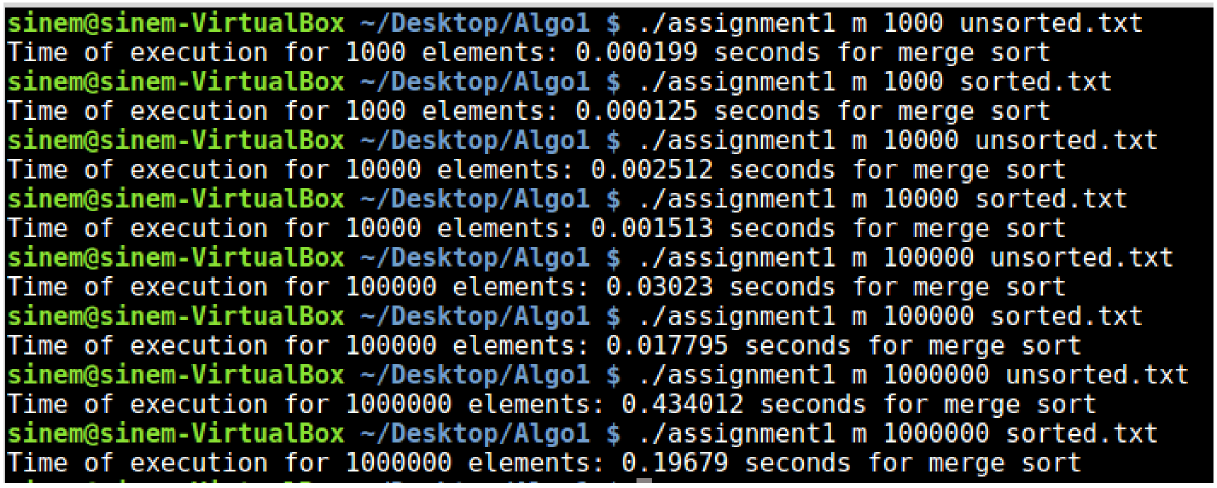
- For calculating the average time of execution, I have run the code for different values for several times for both *merge sort* and *bubble sort* algorithms.
- **Merge sort:**
    - 1st run:



```
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000 unsorted.txt
Time of execution for 1000 elements: 0.000632 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000 sorted.txt
Time of execution for 1000 elements: 0.000161 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 10000 unsorted.txt
Time of execution for 10000 elements: 0.002499 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 10000 sorted.txt
Time of execution for 10000 elements: 0.001574 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 100000 unsorted.txt
Time of execution for 100000 elements: 0.033112 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 100000 sorted.txt
Time of execution for 100000 elements: 0.017797 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000000 unsorted.txt
Time of execution for 1000000 elements: 0.347976 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000000 sorted.txt
Time of execution for 1000000 elements: 0.270628 seconds for merge sort
```

Figure 1

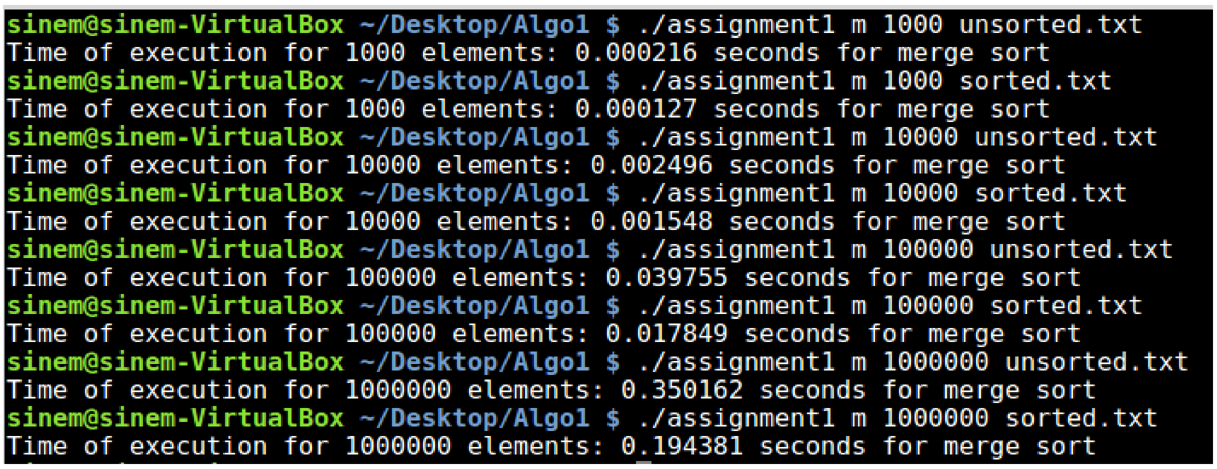
- 2nd run:



```
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000 unsorted.txt
Time of execution for 1000 elements: 0.000199 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000 sorted.txt
Time of execution for 1000 elements: 0.000125 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 10000 unsorted.txt
Time of execution for 10000 elements: 0.002512 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 10000 sorted.txt
Time of execution for 10000 elements: 0.001513 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 100000 unsorted.txt
Time of execution for 100000 elements: 0.03023 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 100000 sorted.txt
Time of execution for 100000 elements: 0.017795 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000000 unsorted.txt
Time of execution for 1000000 elements: 0.434012 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000000 sorted.txt
Time of execution for 1000000 elements: 0.19679 seconds for merge sort
```

Figure 2

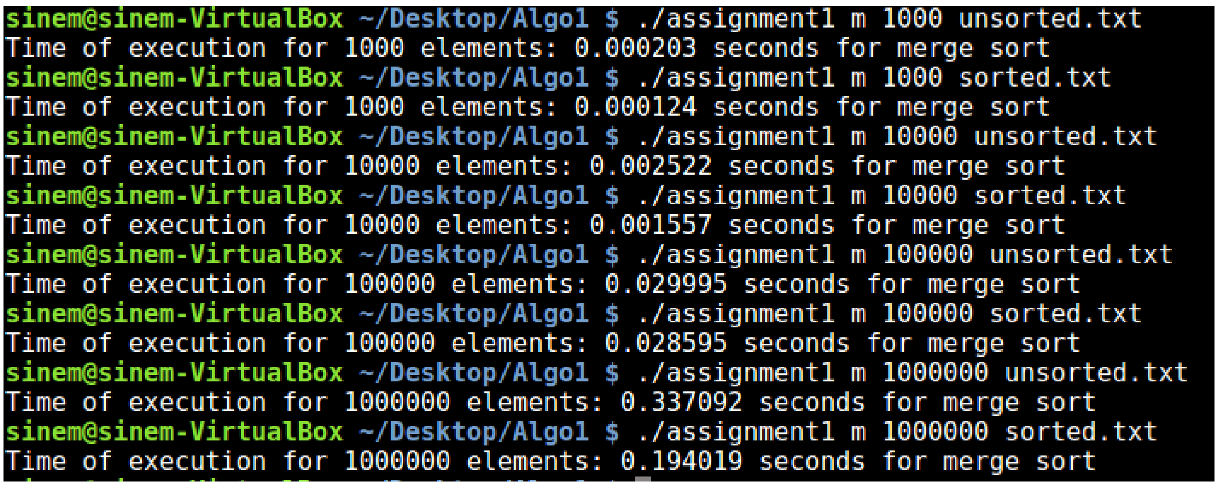
- 3rd run:



```
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000 unsorted.txt
Time of execution for 1000 elements: 0.000216 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000 sorted.txt
Time of execution for 1000 elements: 0.000127 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 10000 unsorted.txt
Time of execution for 10000 elements: 0.002496 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 10000 sorted.txt
Time of execution for 10000 elements: 0.001548 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 100000 unsorted.txt
Time of execution for 100000 elements: 0.039755 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 100000 sorted.txt
Time of execution for 100000 elements: 0.017849 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000000 unsorted.txt
Time of execution for 1000000 elements: 0.350162 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000000 sorted.txt
Time of execution for 1000000 elements: 0.194381 seconds for merge sort
```

Figure 3

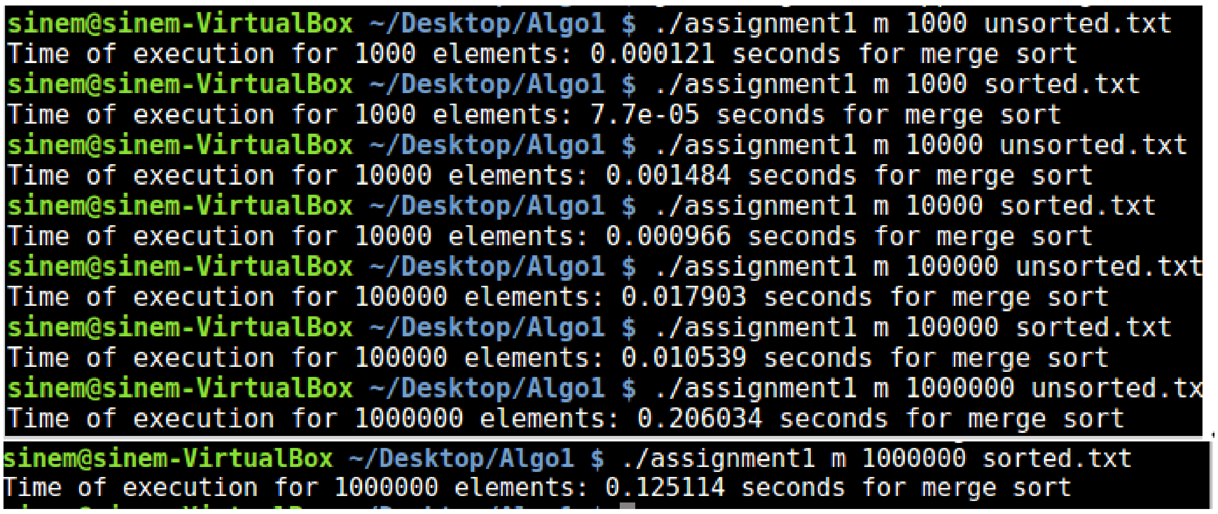
- 4th run:



```
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000 unsorted.txt
Time of execution for 1000 elements: 0.000203 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000 sorted.txt
Time of execution for 1000 elements: 0.000124 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 10000 unsorted.txt
Time of execution for 10000 elements: 0.002522 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 10000 sorted.txt
Time of execution for 10000 elements: 0.001557 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 100000 unsorted.txt
Time of execution for 100000 elements: 0.029995 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 100000 sorted.txt
Time of execution for 100000 elements: 0.028595 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000000 unsorted.txt
Time of execution for 1000000 elements: 0.337092 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000000 sorted.txt
Time of execution for 1000000 elements: 0.194019 seconds for merge sort
```

Figure 4

- 5th run:



```
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000 unsorted.txt
Time of execution for 1000 elements: 0.000121 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000 sorted.txt
Time of execution for 1000 elements: 7.7e-05 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 10000 unsorted.txt
Time of execution for 10000 elements: 0.001484 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 10000 sorted.txt
Time of execution for 10000 elements: 0.000966 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 100000 unsorted.txt
Time of execution for 100000 elements: 0.017903 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 100000 sorted.txt
Time of execution for 100000 elements: 0.010539 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000000 unsorted.tx
Time of execution for 1000000 elements: 0.206034 seconds for merge sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 m 1000000 sorted.txt
Time of execution for 1000000 elements: 0.125114 seconds for merge sort
```

Figure 5

- Hence, average execution of merge sort for: (*both sorted and unsorted*)
  - **1000 elements:**
    - **Unsorted.txt:**

      (0.000632+0.000199+0.000216+0.000203+0.000121)/5 = **0.0002742 seconds**

    - **Sorted.txt:**

      (0.000161+0.000125+0.000127+0.00124+ 7.7e-05)/5 **= 0.000346 seconds**
  - **10000 elements:**
    - **Unsorted.txt:**

      (0.002499+0.002512+0.002496+0.002522+0.001484)/5=**0.0023026 seconds**

    - **Sorted.txt:**

      (0.001574+0.001513+0.001548+0.001557+0.000966)/5=**0.0014316 seconds**
  - **100000 elements:**
    - **Unsorted.txt:**

      (0.033112+0.03023+0.039755+0.029995+0.017903)/5=**0.030199 seconds**

    - **Sorted.txt:**

      (0.017797+0.017795+0.017849+0.028595+0.010539)/5= **0.018515 seconds**
  - **1000000 elements:**
    - **Unsorted.txt:**

      (0.347976+0.434012+0.350162+0.337092+0. 206034)/5=**0.2938484 seconds**

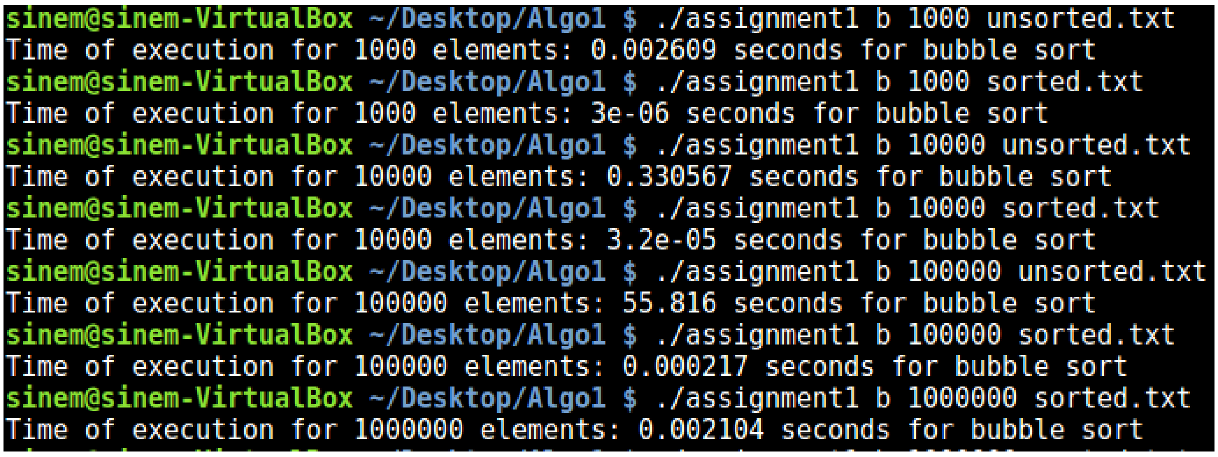
    - **Sorted.txt:**

      (0.270628+0.19679+0.194381+0.194019+ 0.125114)/5=**0.1961864 seconds**

Sinem Elif Haseki
150160026

❖ **Bubble sort:**

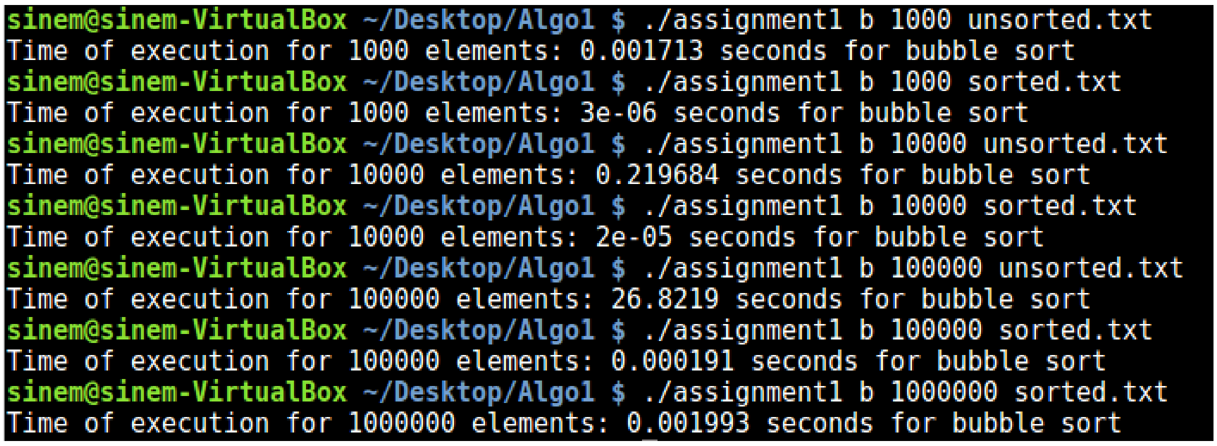
- 1st run:



```
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000 unsorted.txt
Time of execution for 1000 elements: 0.002609 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000 sorted.txt
Time of execution for 1000 elements: 3e-06 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 10000 unsorted.txt
Time of execution for 10000 elements: 0.330567 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 10000 sorted.txt
Time of execution for 10000 elements: 3.2e-05 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 100000 unsorted.txt
Time of execution for 100000 elements: 55.816 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 100000 sorted.txt
Time of execution for 100000 elements: 0.000217 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000000 sorted.txt
Time of execution for 1000000 elements: 0.002104 seconds for bubble sort
```

Figure 1

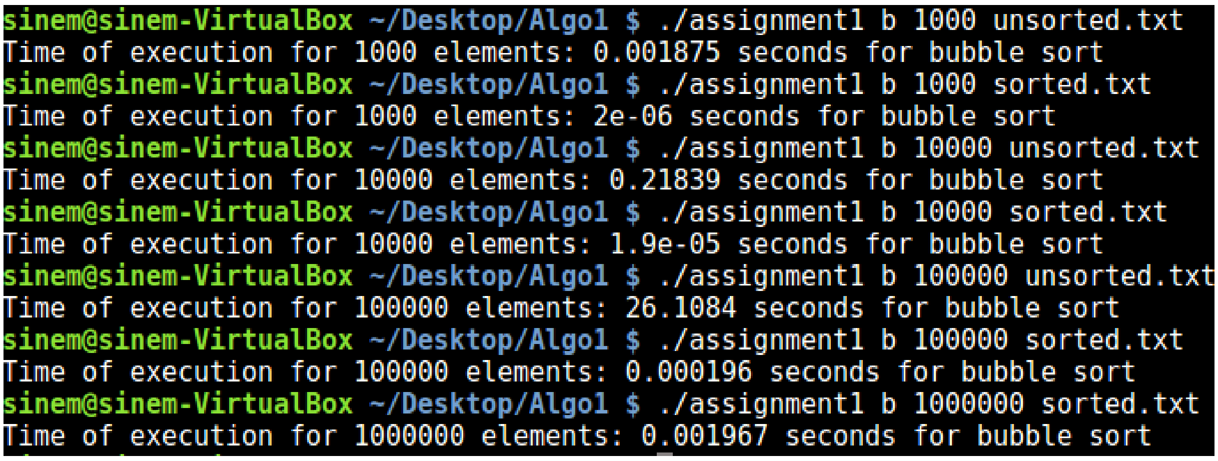
- 2nd run:



```
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000 unsorted.txt
Time of execution for 1000 elements: 0.001713 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000 sorted.txt
Time of execution for 1000 elements: 3e-06 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 10000 unsorted.txt
Time of execution for 10000 elements: 0.219684 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 10000 sorted.txt
Time of execution for 10000 elements: 2e-05 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 100000 unsorted.txt
Time of execution for 100000 elements: 26.8219 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 100000 sorted.txt
Time of execution for 100000 elements: 0.000191 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000000 sorted.txt
Time of execution for 1000000 elements: 0.001993 seconds for bubble sort
```

Figure 2

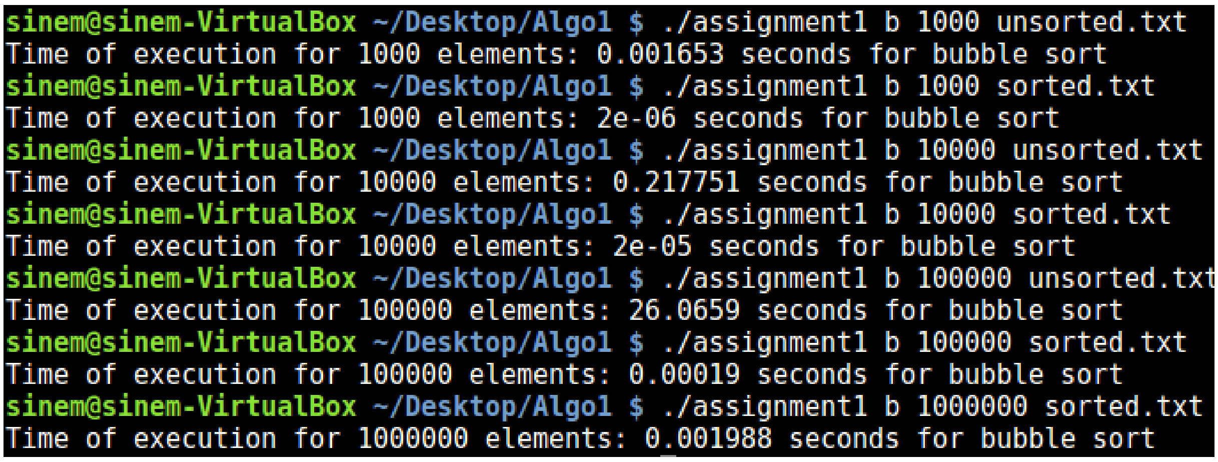
- 3rd run:



```
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000 unsorted.txt
Time of execution for 1000 elements: 0.001875 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000 sorted.txt
Time of execution for 1000 elements: 2e-06 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 10000 unsorted.txt
Time of execution for 10000 elements: 0.21839 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 10000 sorted.txt
Time of execution for 10000 elements: 1.9e-05 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 100000 unsorted.txt
Time of execution for 100000 elements: 26.1084 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 100000 sorted.txt
Time of execution for 100000 elements: 0.000196 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000000 sorted.txt
Time of execution for 1000000 elements: 0.001967 seconds for bubble sort
```

Figure 3

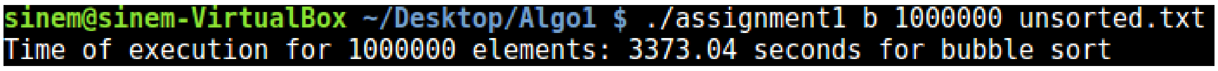
- 4th run:



```
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000 unsorted.txt
Time of execution for 1000 elements: 0.001653 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000 sorted.txt
Time of execution for 1000 elements: 2e-06 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 10000 unsorted.txt
Time of execution for 10000 elements: 0.217751 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 10000 sorted.txt
Time of execution for 10000 elements: 2e-05 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 100000 unsorted.txt
Time of execution for 100000 elements: 26.0659 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 100000 sorted.txt
Time of execution for 100000 elements: 0.00019 seconds for bubble sort
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000000 sorted.txt
Time of execution for 1000000 elements: 0.001988 seconds for bubble sort
```

Figure 4

- For N = 1000000 and unsorted.txt, I've run Bubble Sort algorithm only once since it's already taking too much of time:



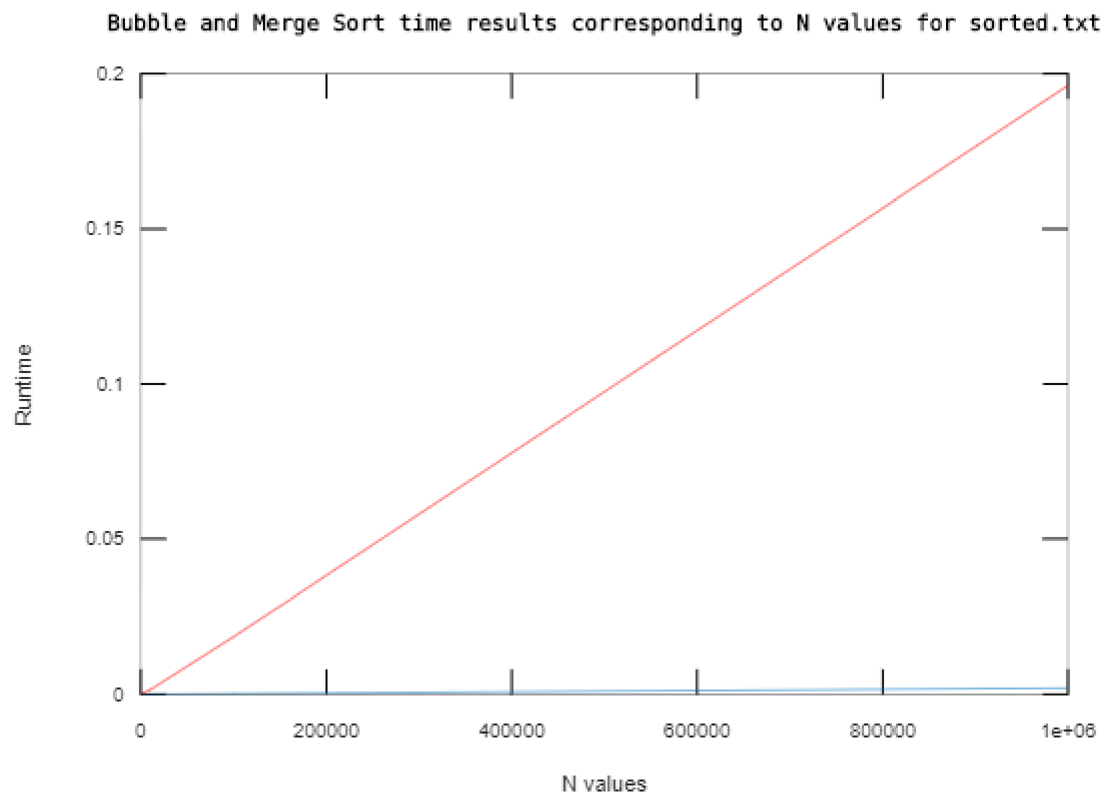
```
sinem@sinem-VirtualBox ~/Desktop/Algo1 $ ./assignment1 b 1000000 unsorted.txt
Time of execution for 1000000 elements: 3373.04 seconds for bubble sort
```

Figure 5

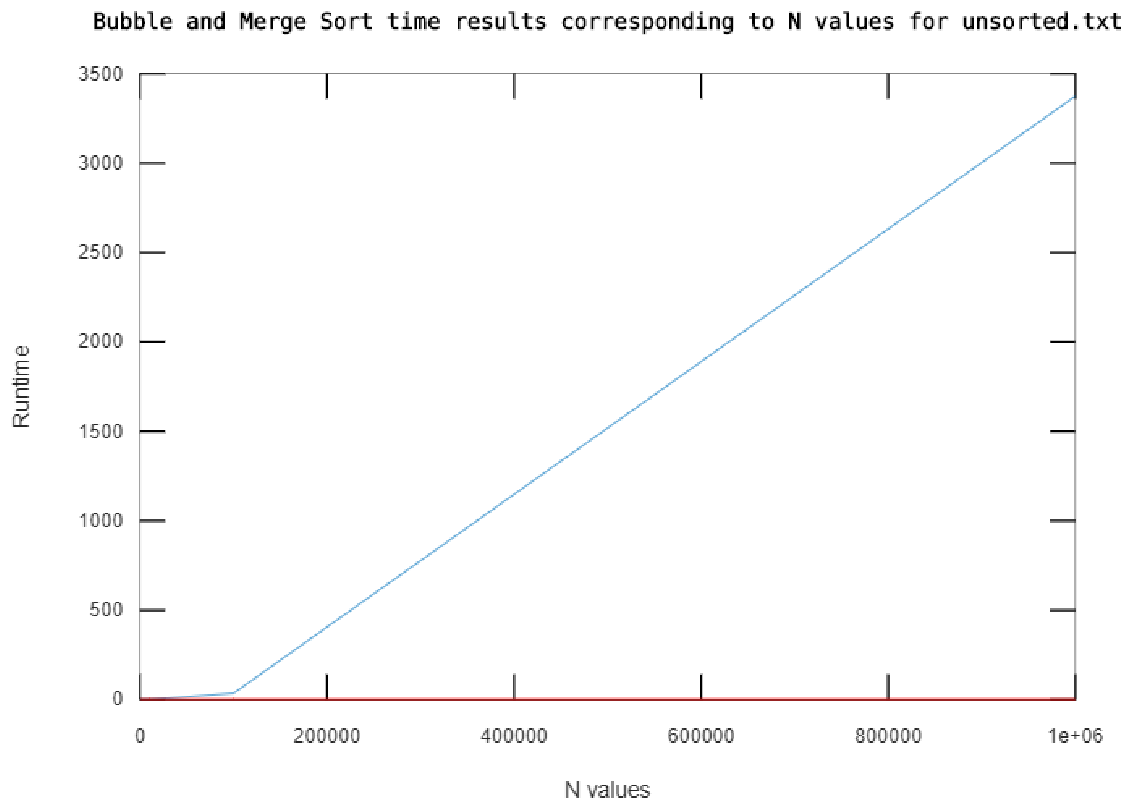
- Hence, average execution of bubble sort for: (*both sorted and unsorted*)
  - **<u>1000 elements:</u>**
    - **<u>unsorted.txt:</u>**
      (0.002609+0.001713+0.001875+0.001653)/4= **0.0019625 seconds**
    - **<u>sorted.txt:</u>**
      (3e-06+3e-06+2e-06+2e-06)/4 = **0.000025 seconds**
  - **<u>10000 elements:</u>**
    - **<u>Unsorted.txt:</u>**
      (0.330567+0.219684+0.21839+0.217751)/4=**0.246598 seconds**
    - **<u>Sorted.txt:</u>**
      (3.2e-05+2e-05+1.9e-05+2e-05)/4=**0.00002275 seconds**
  - **<u>100000 elements:</u>**
    - **<u>Unsorted.txt:</u>**
      (55.816+26.8219+26.1084+26.0659)/4=**33.70305 seconds**
    - **<u>Sorted.txt:</u>**
      (0.000217+0.000191+0.000196+0.00019)/4= **0.0001985 seconds**
  - **<u>1000000 elements:</u>**
    - **<u>Unsorted.txt:</u>**
      3373.04 seconds
    - **<u>Sorted.txt:</u>**
      (0.002104+0.001993+0.001967+0.001988)/4=**0.002013 seconds**

(c)

Results for sorted.txt (red for Merge Sort, blue for Bubble Sort)



Results for unsorted.txt (red for Merge Sort, blue for Bubble Sort)



Conclusion:

- In sorted case, Bubble Sort gives better results since it completes in a shorter time in all values of N. This can also be obtained by comparing the Ω's of the two algorithms, in which best time complexity of Merge Sort is Ω(nlogn), whereas best time complexity of Bubble Sort is Ω(n), and superlinear (nlogn) grows faster than linear (n), which shows that our results and asymptotic bounds are consistent.

- In unsorted case, Merge Sort gives better results since it completes in shorter time span. This can also be found from comparison between their worst-case time complexity O. Merge sort has a worst-case time complexity of O(nlogn), whereas Bubble Sort has $O(n^2)$, and quadratic grows faster than superlinear. This also shows that our results and asymptotic bounds are again consistent.

- We can also say that Merge Sort is more consistent since its best and worst case complexities are the same, so if we don't know whether given data is sorted or unsorted, it would be better to use Merge Sort since if it's unsorted and large data, Bubble Sort would be quite inefficient.

Sinem Elif Haseki
150160026

(d)

```
1. Algorithm Mystery(n)
2. r <- 0
3. for i <- 1 to n do
4.     for j <- i+1 to n do
5.         for k <- 1 to j do
6.             r <- r+1;
7. return r
```

$$\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{k=1}^{j}1 = \sum_{i=1}^{n}\sum_{j=i+1}^{n} j = \sum_{i=1}^{n}\left(\sum_{j=1}^{n} j - \sum_{j=1}^{i} j\right) = \sum_{i=1}^{n}\frac{n(n+1)}{2}-\frac{i(i+1)}{2}$$

$$= {}^{1}\!/_{2}\sum_{i=1}^{n} n^2 + n - i^2 - i$$

$$= {}^{1}\!/_{2}\left(n^2(n) + n(n) - \left(\frac{n(n+1)(2n+1)}{6}\right) - \frac{n(n+1)}{2}\right)$$

$$= {}^{1}\!/_{2}\left(n^3 + n^2 - \left(\frac{n(n+1)(2n+1)}{6}\right) - \frac{n(n+1)}{2}\right) = \frac{n(n^2-1)}{3}$$

❖ Based on this calculation, first 7 results (from 0 to 6) of calculated n values can be found below:

n	**0**	**1**	**2**	**3**	**4**	**5**	**6**
Result	0	0	2	8	20	40	70

	Statement	**Steps/ execution**	**Frequency**		**Total Steps**	
			if-true	if-false	if-true	if-false
1.	Algorithm Mystery(n)	0	-		-	
2.	r <- 0	1	1	1	1	1
3.	for i <- 1 to n do	1	n+1	n+1	n+1	n+1
4.	for j <- i+1 to n do	1	n(n)	n(n)	n(n)	n(n)
5.	for k <- 1 to j do	1	$(n^2-1)(n-1)$	$(n^2-1)(n-1)$	$(n^2-1)(n-1)$	$(n^2-1)(n-1)$
6.	r <- r+1;	1	$(n^2-1)(n-1)-1$	$(n^2-1)(n-1)-1$	$(n^2-1)(n-1)-1$	$(n^2-1)(n-1)-1$
7.	return r	1	1	1	1	1
Total (T(n))		$2n^3 - n^2 - n + 4$				

❖ Since T(n) = $2n^3 - n^2 - n + 4$, thus it is in order of O(n^3), and also it's best case time complexity is Ω(n^3), since there are no differences between if-true and if-false statements.