CSC 505 Project Report on Program 1

Experimental Set UP:

System Architecture :

Operating System : Linux (Ubuntu)

Memory : 1.5 GiB

Processor : Intel® Core ™ i7-6500U CPU : 2.50GHz

Cache : 4 MB SmartCache

Memory Types : DDR4-2133, LPDDR3-1866, DDR3L-1600

OS type : 64-bitDisk Memory 19.4GB

Graphics : Gallium 0.4 on SVGA 3D

Language Used : 'C'

Compiler version : gcc (Ubuntu 5.4.0-6ubuntu1~16.04.4) 5.4.0 20160609

Choice of inputs :

The minimum input size to record an observation of 10ms of runtime was 70,000 records. (random order- On Merge sort)

So, considering this as first input size, we have almost doubled the input size for 8 iterations.

We have applied the shell script for the purpose of experimenting on the 3 algorithms : Mergesort, Heapsort and Utility sort.

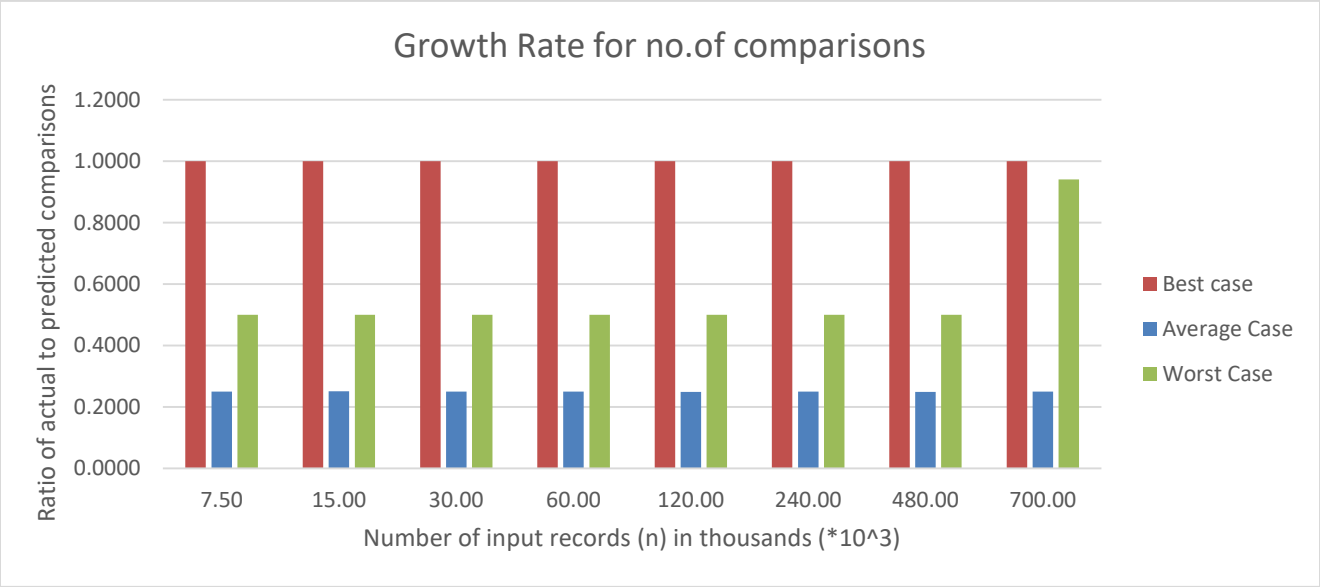
We considered the seed from the numbers 1 to 10.

We took average of the runtime and comparisons for these 10 seeds.

We considered 3 input types , i.e random, sorted and reverse sorted.

Tables and Charts :

1) The growth rate in number of comparisons and how accurately this is predicted by the worst case theoretical analysis for Insertion sort



Average case (Predicted) in millions (*10^6)	Number of comparisons in millions (*10^6)	Ratio of actual to predicted comparisons (Random Input)	Best case (Predicted) in millions (*10^6)	Number of comparisons in millions (*10^6)	Ratio of actual to predicted comparisons (Sorted Input)	Worst case (Predicted) in millions (*10^6)	Number of comparisons in millions (*10^6)	Ratio of actual to predicted comparisons (Reverse Sorted Input)
56.2500	14.0769	0.2503	0.0075	0.0075	0.9999	56.2500	28.1213	0.4999
225.0000	56.4808	0.2510	0.0150	0.0150	0.9999	225.0000	112.4925	0.5000
900.0000	224.7485	0.2497	0.0300	0.0300	1.0000	900.0000	449.9850	0.5000
3600.0000	899.4074	0.2498	0.0600	0.0600	1.0000	3600.0000	1799.9700	0.5000
14400.0000	3591.3351	0.2494	0.1200	0.1200	1.0000	14400.0000	7199.9400	0.5000

57600.0000	14416.6320	0.2503	0.2400	0.2400	1.0000	57600.0000	28799.8800	0.5000
230400.0000	57476.6000	0.2495	0.4800	0.4800	1.0000	230400.0000	115199.7600	0.5000
490000.0000	122414.6480	0.2498	0.7000	0.7000	1.0000	490000.0000	460799.0400	0.9404

Merge Sort

Average case (Predicted) in millions (*10^6)	Number of comparisons in millions (*10^6)	Ratio of actual to predicted comparisons (Random Input)	Best case (Predicted) in millions (*10^6)	Number of comparisons in millions (*10^6)	Ratio of actual to predicted comparisons (Sorted Input)	Worst case (Predicted) in millions (*10^6)	Number of comparisons in millions (*10^6)	Ratio of actual to predicted comparisons (Reverse Sorted Input)
1.1267	0.935108	0.8300	1.1267	0.5552	0.4928	1.1267	0.5737	0.5092
1.8786	1.7388325	0.9256	1.8786	0.9220	0.4908	1.8786	0.9629	0.5126
3.1273	2.9049882	0.9289	3.1273	1.5288	0.4889	3.1273	1.6138	0.5161
5.1980	4.8398531	0.9311	5.1980	2.5600	0.4925	5.1980	2.6501	0.5098
8.6279	8.0548266	0.9336	8.6279	4.2599	0.4937	8.6279	4.3909	0.5089
14.3023	13.3912198	0.9363	14.3023	7.0080	0.4900	14.3023	7.3575	0.5144
23.6800	22.2144851	0.9381	23.6800	11.6681	0.4927	23.6800	12.0717	0.5098
41.8631	39.3488493	0.9399	41.8631	20.7705	0.4962	41.8631	21.1329	0.5048

Heap Sort

Average case (Predicted) in millions (*10^6)	Number of comparisons in millions (*10^6)	Ratio of actual to predicted comparisons (Random Input)	Best case (Predicted) in millions (*10^6)	Number of comparisons in millions (*10^6)	Ratio of actual to predicted comparisons (Sorted Input)	Worst case (Predicted) in millions (*10^6)	Number of comparisons in millions (*10^6)	Ratio of actual to predicted comparisons (Reverse Sorted Input)
1.1267	2.0384735	1.8093	1.1267	2.0992	1.8632	1.1267	1.9786	1.7562
1.8786	3.4168997	1.8189	1.8786	3.5178	1.8726	1.8786	3.3200	1.7673
3.1273	5.7126265	1.8267	3.1273	5.8831	1.8812	3.1273	5.5526	1.7756
5.1980	9.5177885	1.8310	5.1980	9.7779	1.8811	5.1980	9.2689	1.7831
8.6279	15.8598557	1.8382	8.6279	16.2790	1.8868	8.6279	15.4760	1.7937

14.3023	26.3865251	1.8449	14.3023	27.0982	1.8947	14.3023	25.7529	1.8006
23.6800	43.769838	1.8484	23.6800	44.8939	1.8959	23.6800	42.8083	1.8078
41.8631	77.5884579	1.8534	41.8631	79.4723	1.8984	41.8631	75.9279	1.8137

Utility Sort

Average case (Predicted) in millions (*10^6)	Number of comparisons in millions (*10^6)	Ratio of actual to predicted comparisons (Random Input)	Best case (Predicted) in millions (*10^6)	Number of comparisons in millions (*10^6)	Ratio of actual to predicted comparisons (Sorted Input)	Worst case (Predicted) in millions (*10^6)	Number of comparisons in millions (*10^6)	Ratio of actual to predicted comparisons (Reverse Sorted Input)
1.1267	1.0390063	0.9222	1.1267	0.5552	0.4928	1.1267	0.5737	0.5092
1.8786	1.7388325	0.9256	1.8786	0.9220	0.4908	1.8786	0.9629	0.5126
3.1273	2.9049882	0.9289	3.1273	1.5288	0.4889	3.1273	1.6138	0.5161
5.1980	4.8398531	0.9311	5.1980	2.5600	0.4925	5.1980	2.6501	0.5098
8.6279	8.0548266	0.9336	8.6279	4.2598	0.4937	8.6279	4.3909	0.5089
14.3023	13.3912198	0.9363	14.3023	7.0080	0.4900	14.3023	7.3575	0.5144
23.6800	22.2144851	0.9381	23.6800	11.6679	0.4927	23.6800	12.0719	0.5098
41.8631	39.3488493	0.9399	41.8631	20.7700	0.4961	41.8631	21.1333	0.5048

2)The growth rate in runtime (ratio between runtime and n lg n for Insertion Sort)

		Random input	Sorted Input	Reverse Sorted Input
Number of records in thousands (* 10^3)	n lg n	time in ms/n lg n	time in ms/n lg n	time in ms/n lg n
7.50	21.80168	3.09150495	0	6.265572348

15.00	58.60336	4.586767805	0	9.170122836
30.00	147.2067	7.245593241	0	14.51971779
60.00	354.4134	12.06049085	0	24.12211033
120.00	828.8269	20.6528053	0	41.66853319
240.00	1897.654	36.16255086	0.000526967	72.9079267
480.00	4275.307	64.26354149	0.000467803	129.5576989
700.00	6615.848	88.93538965	0.000453457	334.4340853

For merge sort

Merge Sort				
		Random input	Sorted Input	Reverse Sorted Input
Number of records in thousands (* 10^6)	n lg n	time in ms/n lg n (*10^-6)	time in ms/n lg n (*10^-6)	time in ms/n lg n
0.07	1126654.71	20.50	8.88	11.63
0.11	1878591.59	23.79	9.48	10.59
0.18	3127257.03	20.72	13.72	13.08
0.29	5198028.03	21.10	9.12	8.85
0.46	8627911.69	16.15	8.02	7.94
0.73	14302298.67	14.97	8.05	7.94
1.17	23680005.52	15.13	8.08	7.98
2.00	41863137.14	15.51	8.04	7.94

For Heap sort

Heap Sort				
		Random input	Sorted Input	Reverse Sorted Input

Number of records in thousands (* 10^6)	n lg n	time in ms/n lg n (*10^-6)	time in ms/n lg n (*10^-6)	time in ms/n lg n
0.07	1126654.71	36.12	22.63	28.76
0.11	1878591.59	33.80	20.39	18.95
0.18	3127257.03	39.14	34.06	41.22
0.29	5198028.03	33.38	20.08	20.03
0.46	8627911.69	25.68	19.08	18.47
0.73	14302298.67	24.80	18.58	18.40
1.17	23680005.52	25.83	18.75	18.97
2.00	41863137.14	26.30	18.31	18.16

For Utility sort :

Utility Sort				
		Random input	Sorted Input	Reverse Sorted Input
Number of records in thousands (* 10^6)	n lg n	time in ms/n lg n (*10^-6)	time in ms/n lg n (*10^-6)	time in ms/n lg n
0.07	1126654.71	10.65	2.04	3.02
0.11	1878591.59	12.35	3.19	4.42
0.18	3127257.03	15.32	2.97	3.58
0.29	5198028.03	19.14	3.04	3.39
0.46	8627911.69	11.20	2.02	2.63
0.73	14302298.67	10.29	1.99	2.71
1.17	23680005.52	10.45	2.17	3.63
2.00	41863137.14	10.25	2.04	2.79

3)Time per comparison for

1.Insertion Sort

	Random Input	Sorted Input	Reverse Sorted Input
Number of records in thousands (* 10^3)	Time per Comparison in Milliseconds	Time per Comparison in Milliseconds	Time per Comparison in Milliseconds
7.50	4.7880	0	4.85753656
15.00	4.7591	0	4.777207369
30.00	4.7457	0	4.749936109
60.00	4.7525	0	4.749634716
120.00	4.7664	0	4.796706639
240.00	4.7601	4.166684028	4.80397835
480.00	4.7801	4.166675347	4.808161059
700.00	4.8065	4.285720408	4.801583354

2) Merge Sort

Merge Sort				
		Random input	Sorted Input	Reverse Sorted Input
Number of records in thousands (* 10^6)	n lg n	time in ms/n lg n (*10^-6)	time in ms/n lg n (*10^-6)	time in ms/n lg n
0.07	1126654.71	20.50	8.88	11.63
0.11	1878591.59	23.79	9.48	10.59
0.18	3127257.03	20.72	13.72	13.08
0.29	5198028.03	21.10	9.12	8.85
0.46	8627911.69	16.15	8.02	7.94
0.73	14302298.67	14.97	8.05	7.94
1.17	23680005.52	15.13	8.08	7.98
2.00	41863137.14	15.51	8.04	7.94

For Heap Sort :

Heap Sort				
		Random input	Sorted Input	Reverse Sorted Input
Number of records in thousands (* 10^6)	n lg n	time in ms/n lg n (*10^-6)	time in ms/n lg n (*10^-6)	time in ms/n lg n
0.07	1126654.71	36.12	22.63	28.76
0.11	1878591.59	33.80	20.39	18.95
0.18	3127257.03	39.14	34.06	41.22

0.29	5198028.03	33.38	20.08	20.03
0.46	8627911.69	25.68	19.08	18.47
0.73	14302298.67	24.80	18.58	18.40
1.17	23680005.52	25.83	18.75	18.97
2.00	41863137.14	26.30	18.31	18.16

Utility Sort

		Random input	Sorted Input	Reverse Sorted Input
Number of records in thousands (* 10^6)	n lg n	time in ms/n lg n (*10^-6)	time in ms/n lg n (*10^-6)	time in ms/n lg n
0.07	1126654.71	10.65	2.04	3.02
0.11	1878591.59	12.35	3.19	4.42
0.18	3127257.03	15.32	2.97	3.58
0.29	5198028.03	19.14	3.04	3.39
0.46	8627911.69	11.20	2.02	2.63
0.73	14302298.67	10.29	1.99	2.71
1.17	23680005.52	10.45	2.17	3.63
2.00	41863137.14	10.25	2.04	2.79

4) The impact of different types of inputs on runtime and number of comparisons (Comparison between Insertion and Merge Sort)

	Random Input		Sorted Input		Reverse Sorted Input	
	Merge Sort	Insertion Sort	Merge Sort	Insertion Sort	Merge Sort	Insertion Sort
Number of records in thousands (* 10^3)	Time in seconds	Time in seconds	Time in seconds	Time in seconds	Time in seconds	Time in seconds
7.50	0.001	0.067	0.001	0.000	0.001	0.137
15.00	0.004	0.269	0.002	0.000	0.002	0.537
30.00	0.006	1.067	0.003	0.000	0.003	2.137
60.00	0.012	4.274	0.007	0.000	0.007	8.549
120.00	0.027	17.118	0.014	0.000	0.014	34.536
240.00	0.057	68.624	0.031	0.001	0.030	138.354
480.00	0.121	274.746	0.065	0.002	0.064	553.899
700.00	0.181	588.383	0.097	0.003	0.096	2212.565

	Random Input		Sorted Input		Reverse Sorted Input	
	Merge Sort	Insertion Sort	Merge Sort	Insertion Sort	Merge Sort	Insertion Sort
Number of records in thousands (* 10^3)	Number of comparisons in millions (*10^6)	Number of comparisons in millions (*10^6)	Number of comparisons in millions (*10^6)	Number of comparisons in millions (*10^6)	Number of comparisons in millions (*10^6)	Number of comparisons in millions (*10^6)
7.50	0.087	14.077	0.047	0.007	0.049	28.121
15.00	0.189	56.481	0.102	0.015	0.106	112.493
30.00	0.409	224.749	0.220	0.030	0.228	449.985
60.00	0.877	899.407	0.469	0.060	0.485	1799.970

120.00	1.874	3591.335	0.998	0.120	1.031	7199.940
240.00	3.989	14416.632	2.116	0.240	2.182	28799.880
480.00	8.458	57476.600	4.470	0.480	4.604	115199.760
700.00	12.724	122414.648	6.650	0.700	7.000	460799.040

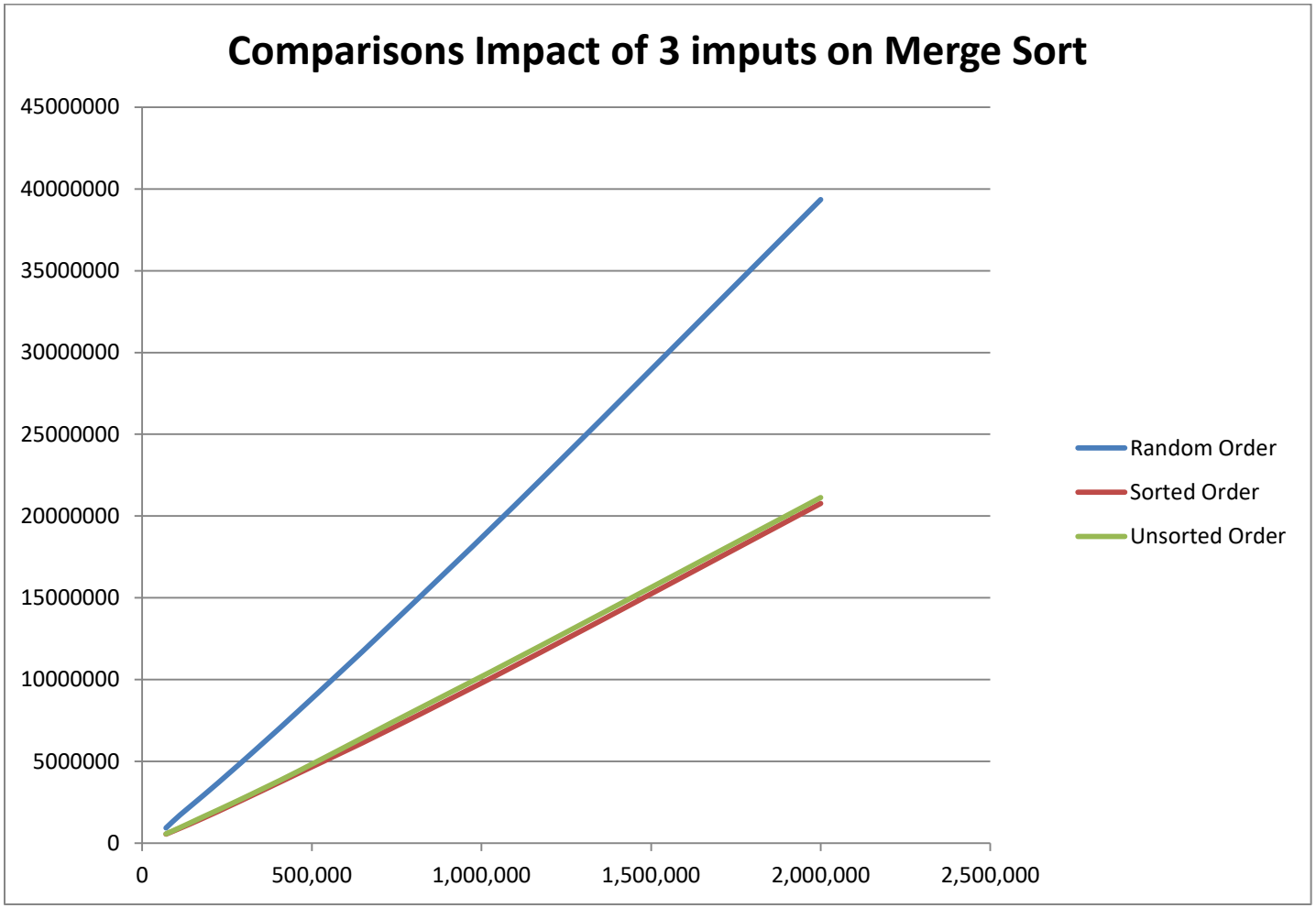
Note :

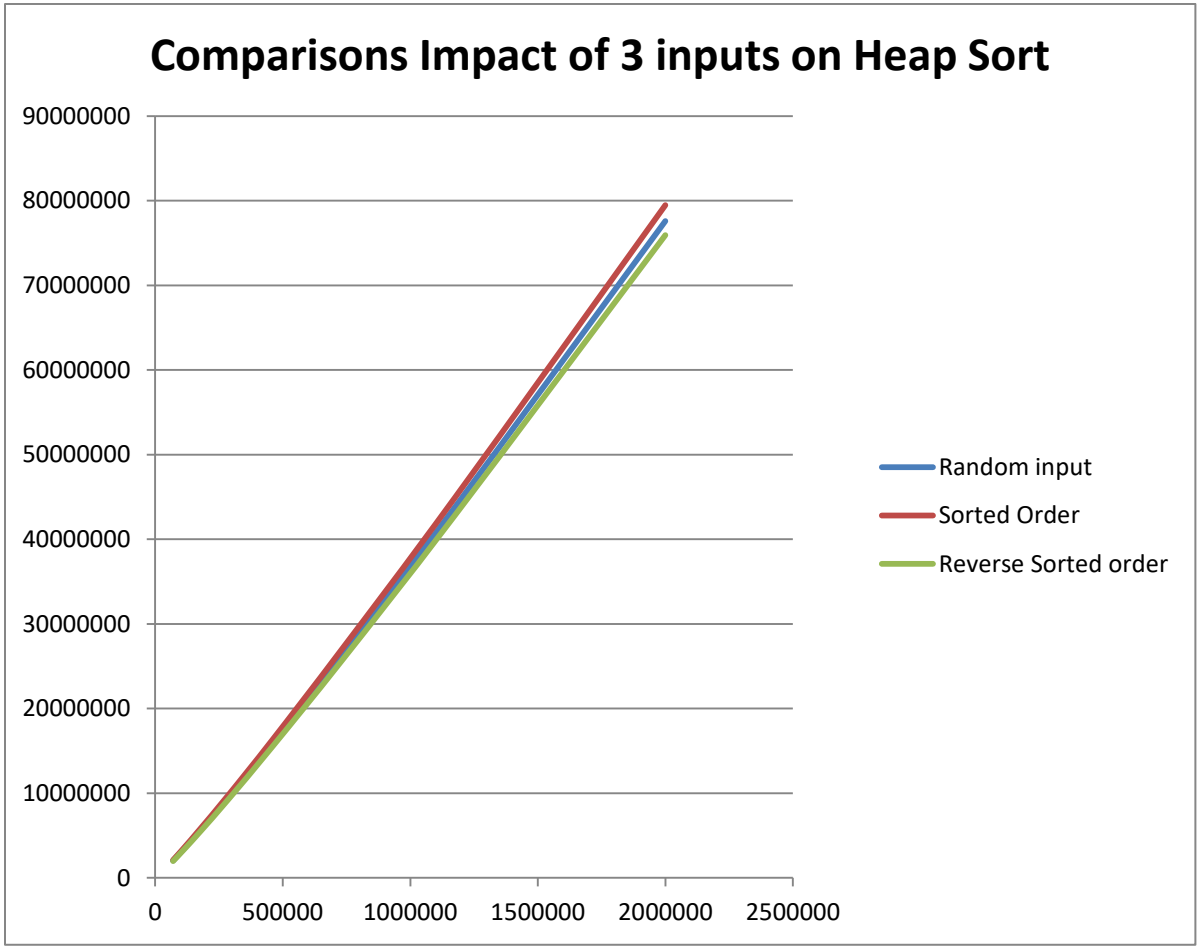
- Insertion sort values are too big, outperforming the curve of Merge sort.
- Hence we couldn't show the graph.

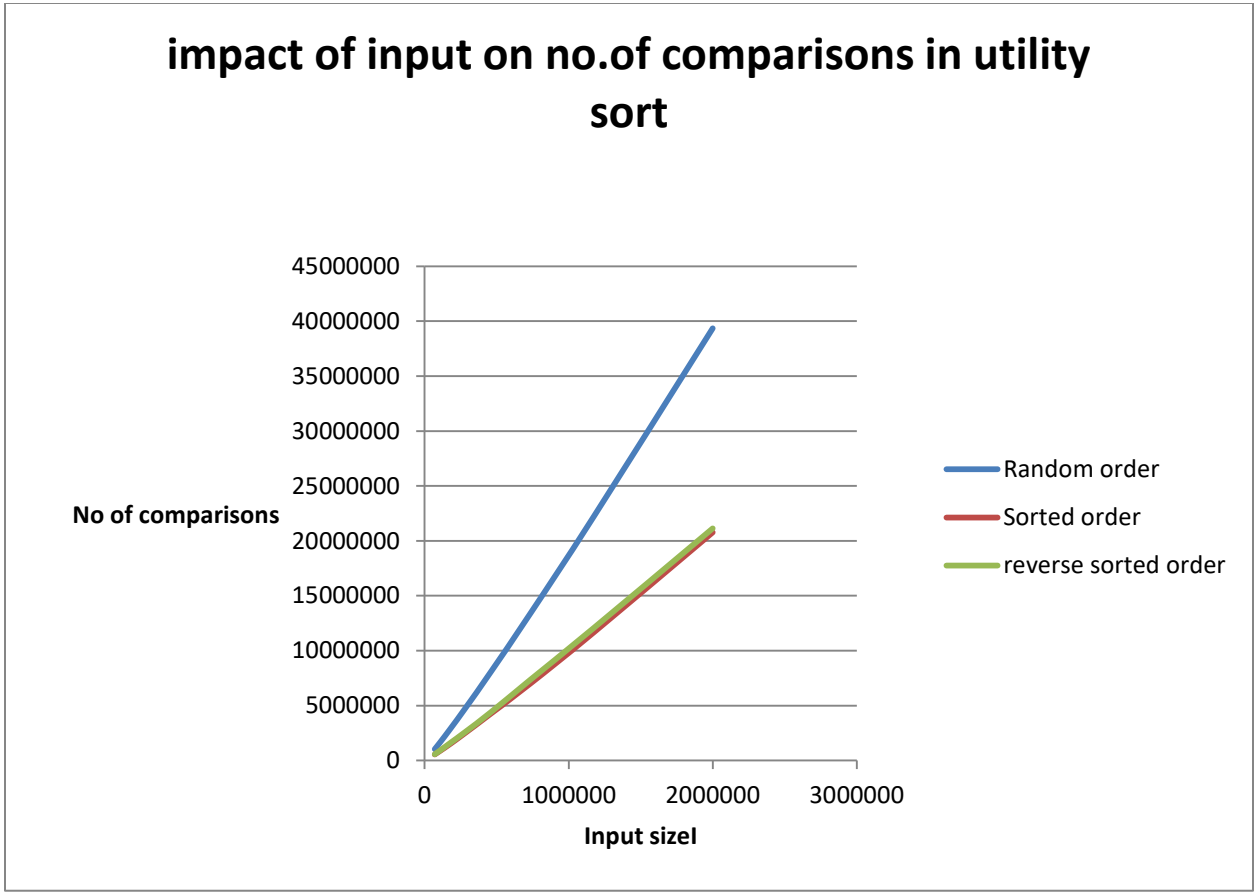
The table for the Number of Comparisons for the other 3 algorithms :

	Random Input			**Sorted Input**			**Reverse Sorted Input**		
	Merge Sort	**Heap Sort**	**Utility Sort**	**Merge Sort**	**Heap Sort**	**Utility Sort**	**Merge Sort**	**Heap Sort**	**Utility Sort**
Number of records in thousands (* 10^6)	**Number of comparisons in millions (*10^6)**	**Number of comparisons in millions (*10^6)**	**Number of comparisons in millions (*10^6)**	**Number of comparisons in millions (*10^6)**	**Number of comparisons in millions (*10^6)**	**Number of comparisons in millions (*10^6)**	**Number of comparisons in millions (*10^6)**	**Number of comparisons in millions (*10^6)**	**Number of comparisons in millions (*10^6)**
0.07	0.94	2.04	1.04	0.56	2.10	0.56	0.57	1.98	0.57
0.11	1.74	3.42	1.74	0.92	3.52	0.92	0.96	3.32	0.96
0.18	2.90	5.71	2.90	1.53	5.88	1.53	1.61	5.55	1.61
0.29	4.84	9.52	4.84	2.56	9.78	2.56	2.65	9.27	2.65
0.46	8.05	15.86	8.05	4.26	16.28	4.26	4.39	15.48	4.39
0.73	13.39	26.39	13.39	7.01	27.10	7.01	7.36	25.75	7.36
1.17	22.21	43.77	22.21	11.67	44.89	11.67	12.07	42.81	12.07

2.00	39.35	77.59	39.35	20.77	79.47	20.77	21.13	75.93	21.13

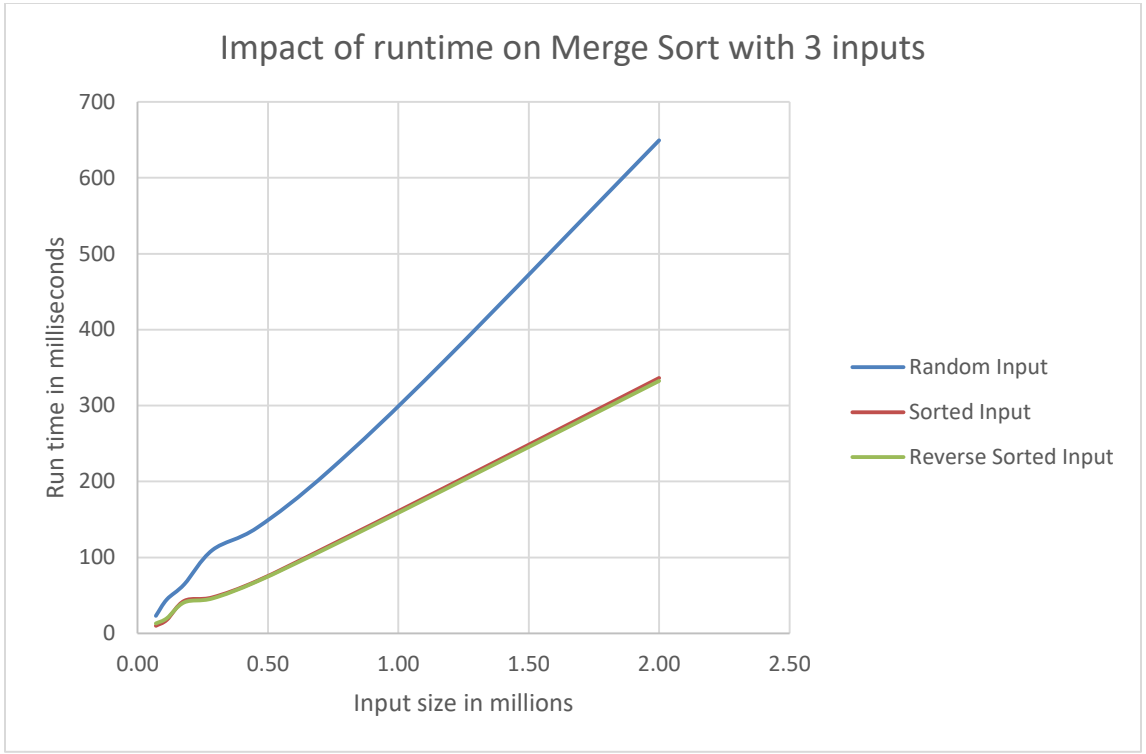


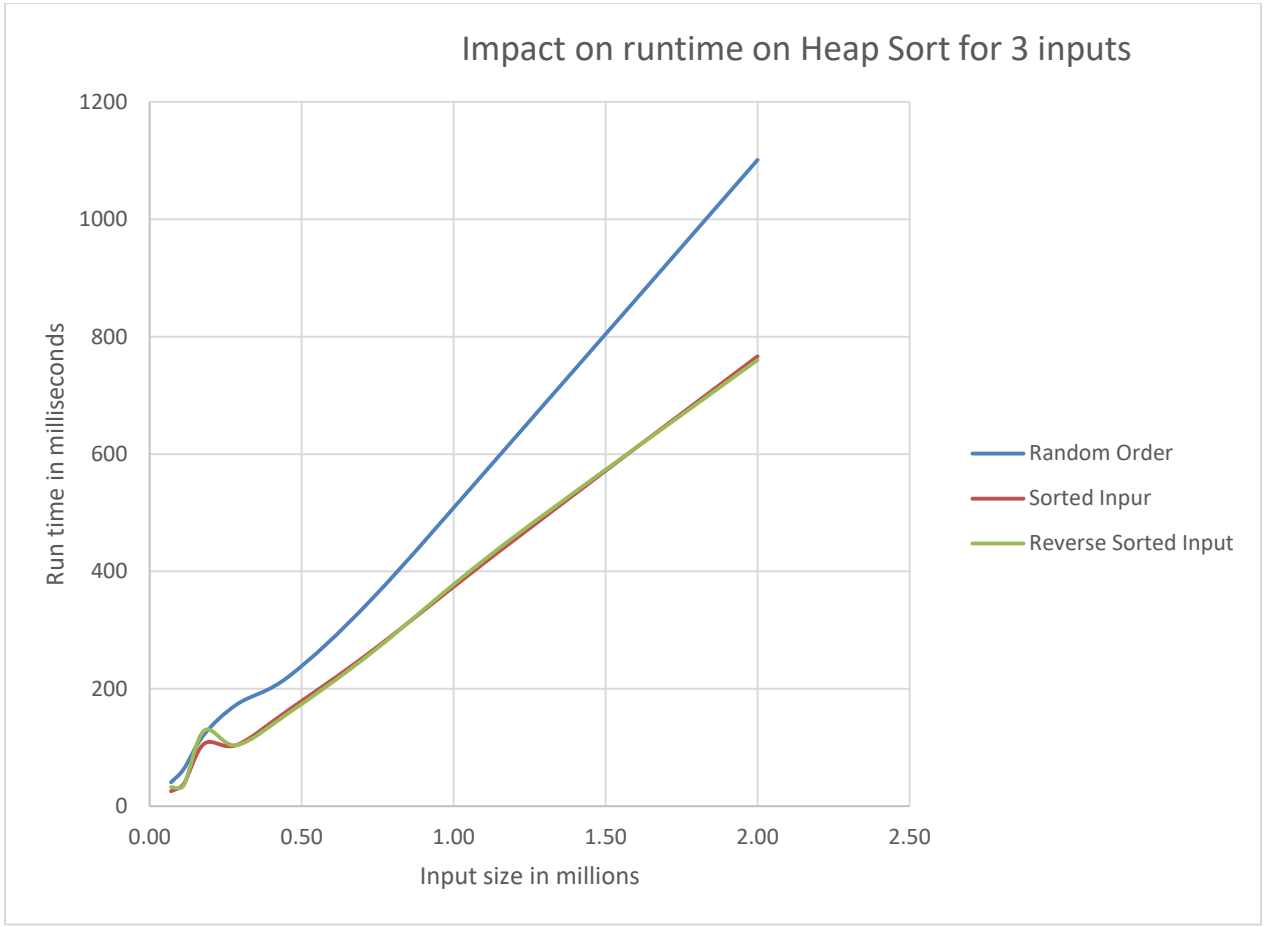

Comparisons Impact of 3 inputs on Heap Sort
90000000
80000000
70000000
60000000
50000000
40000000
30000000
20000000
10000000
0
0
500000
1000000
1500000
2000000
2500000
Random input
Sorted Order
Reverse Sorted order

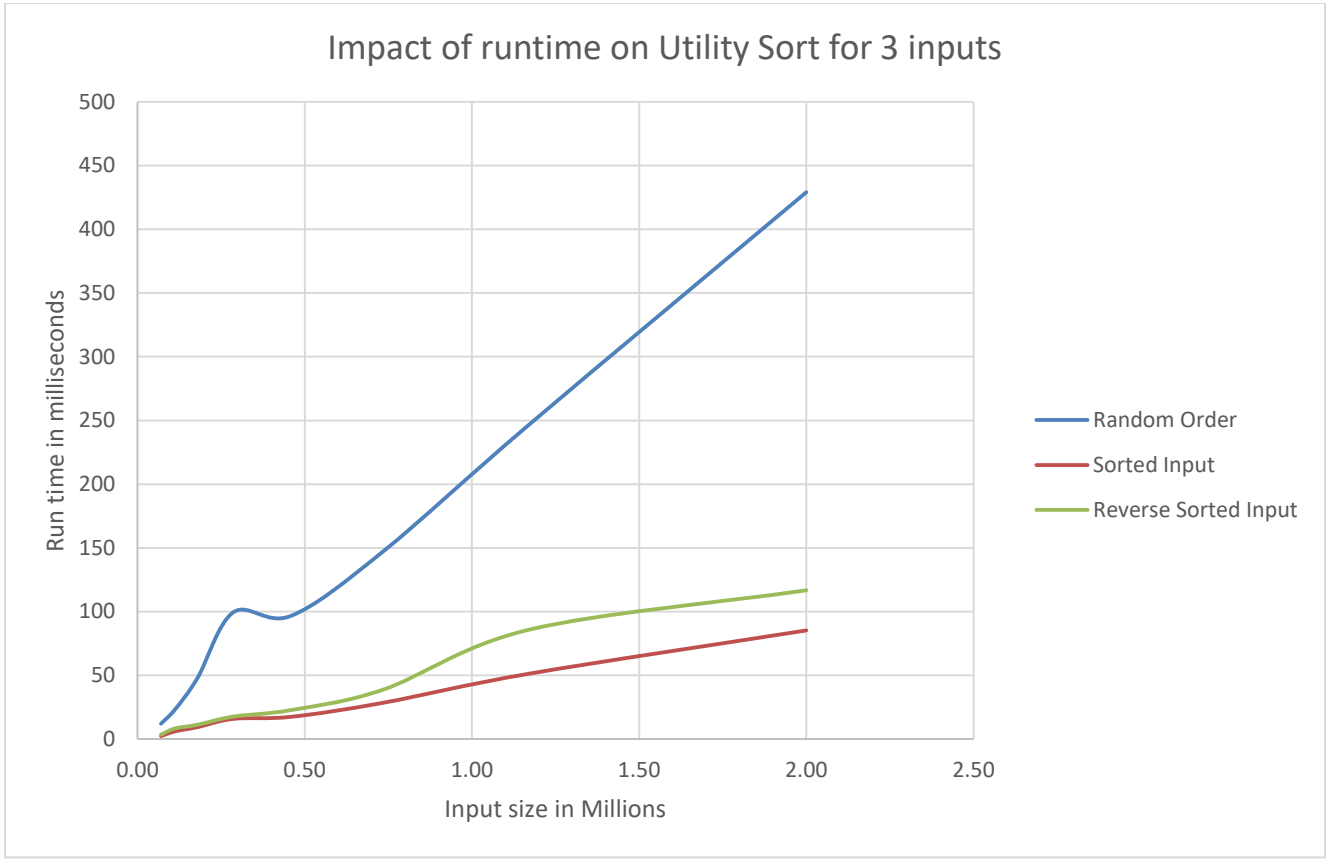


The Table for the Runtime for the other 3 algorithms :

	Random Input			Sorted Input			Reverse Sorted Input		
	Merge Sort	Heap Sort	Utility Sort	Merge Sort	Heap Sort	Utility Sort	Merge Sort	Heap Sort	Utility Sort
Number of records in thousands (* 10^3)	Time in miliseconds	Time in seconds	Time in seconds	Time in seconds	Time in seconds	Time in seconds	Time in seconds	Time in seconds	Time in seconds
0.07	23.1	40.7	12	10	25.5	2.3	13.1	32.4	3.4
0.11	44.7	63.5	23.2	17.8	38.3	6	19.9	35.6	8.3
0.18	64.8	122.4	47.9	42.9	106.5	9.3	40.9	128.9	11.2
0.29	109.7	173.5	99.5	47.4	104.4	15.8	46	104.1	17.6
0.46	139.3	221.6	96.6	69.2	164.6	17.4	68.5	159.4	22.7
0.73	214.1	354.7	147.1	115.2	265.8	28.4	113.5	263.2	38.7
1.17	358.3	611.6	247.5	191.4	443.9	51.4	188.9	449.3	86
2.00	649.4	1101.1	429.1	336.4	766.5	85.2	332.5	760.3	116.7


Impact of runtime on Merge Sort with 3 inputs
Run time in milliseconds
700
600
500
400
300
200
100
0
0.00
0.50
1.00
1.50
2.00
2.50
Input size in millions
Random Input
Sorted Input
Reverse Sorted Input


Impact on runtime on Heap Sort for 3 inputs
Run time in milliseconds
1200
1000
800
600
400
200
0
0.00
0.50
1.00
1.50
2.00
2.50
Input size in millions
Random Order
Sorted Inpur
Reverse Sorted Input


Impact of runtime on Utility Sort for 3 inputs
Run time in milliseconds
500
450
400
350
300
250
200
150
100
50
0
0.00
0.50
1.00
1.50
2.00
2.50
Input size in Millions
Random Order
Sorted Input
Reverse Sorted Input

Summary and Conclusion :

We have made many conclusions through this Programming task. We have listed a few below:

NOTE :

- The input size for every iteration is not doubled, rather they are increased by a factor of 1.6 and nor 2.
- This is because we have used 'C' langue for programming and the algorithms are implemented using recursion mechanisms rather than iterative.
- So any input size above 2 million, we are not able to process due to segmentation fault error.
- This is because stack is continuously busy with recursive calls.

The impact on input i.e (sorted and reverse sorted order) on Heap and Merge sort algorithms is not there.

If the elements are sorted either in ascending or descending, the algorithms runtime and number of comparisons are not effected.

Insertion sort outperforms merge sort for sorted inputs.

But for any other case, merge sorts performs better.

Heap sort takes more number of comparisons than Merge sort (actually double for many input sizes).

This is because the heapify step is costly and in every comparison, we are actually making 2 comparisons :

Once for left child and once for right child for each parent.

The actual/predicted number of comparisons for insertion sort on the sorted order, reverse sorted and random order is pretty much similar which implies the deviation for different inputs is not that much.