COMP2521 Sort Detective Lab Report

by Jasmine Tran z5394841, Bryan Le z5361001

[using z5394841 sort programs]

In this lab, the aim is to measure the performance of two sorting programs, without access to the code, and determine which sorting algorithm each program uses.

Experimental Design

We measured how each program's execution time varied as the size and initial sortedness of the input varied. We used the following kinds of inputs generated by the *gen* program: for each input type ascending, descending, random, we tested the execution time for input sizes from 1..10 000 000.

We used these test cases because …

- Adaptability can be determined by analysing whether or not the time increases as the input size grows for ascending input data.
- Average Time complexity can be determined by analysing the growth rate in time as the input size grows for all types of inputs.

Because of the way timing works on Unix/Linux, it was necessary to **repeat** the same test multiple times…and so each input size and input type combination were repeated at least 10 times.

We also investigated the **stability** of the sorting programs by having data with equivalent numeric keys but differing alphabetical keys to see if the alphabetical order is maintained. For example, using the input:

- 1 ddd
- 2 eee
- 1 ccc
- 3 fff
- 1 bbb
- 4 ggg
- 1 aaa

We also investigated the difference between Bubble Sort and Insertion Sort. The following kind of input data will be used,

- 10 aaa
- 1 aaa
- …
- 9 aaa,

where the sequence is ordered in ascending numbers, but the highest number is placed at the start. Bubble Sort will operate much faster because it will have sorted the element “10 aaa” on the first sweep, and stop executing after the second sweep once detected the sorted array. Insertion sort, however, will have to insert all of the elements before the element “10 aaa.”

Summary Information

<table>
<tr><td colspan="2">Stable</td><td colspan="6">Not Stable</td></tr>
<tr><td colspan="2">Adaptive</td><td>Not Adaptive</td><td colspan="6">Not Adaptive</td></tr>
<tr><td colspan="2">$O(n^2)$</td><td>O(nlogn)</td><td>$O(n^2)$</td><td>O(n!)</td><td>W: $O(n^2)$
B: O(nlogn)</td><td>More likely O(nlogn)</td><td>$O(n^2)$ - O(nlogn)</td></tr>
<tr><td>Bubble Sort</td><td>Insertion Sort</td><td>Merge Sort</td><td>Selection Sort</td><td>Bogosort</td><td>Naive Quicksort</td><td>Median-of-3 Quicksort</td><td>Randomised Quicksort</td></tr>
</table>

Experimental Results

Stability		
Input	A	B
1 ddd 2 eee 1 ccc 3 fff 1 bbb 4 ggg 1 aaa	1 ddd 1 ccc 1 bbb 1 aaa 2 eee 3 fff 4 ggg	1 ccc 1 ddd 1 bbb 1 aaa 2 eee 3 fff 4 ggg
Results Analysis:	A is stable	B is not stable

Program's Execution Time vs. Input Size and Input Type **note: '-' are times > 60 seconds*							
SEED = 1	#Repetitions	**Ascending (A)** (average time)		**Descending (D)** (average time)		**Random (R)** (average time)	
		A	**B**	**A**	**B**	**A**	**B**
5	10	0.00	0.00	0.00	0.00	0.00	0.00
10	10	0.00	0.00	0.00	0.00	0.00	0.00
100	10	0.00	0.00	0.00	0.00	0.00	0.00
1 000	10	0.00	0.00	0.01	0.00	0.00	0.00
5 000	10	0.00	0.06	0.07	0.02	0.06	0.00
10 000	10	0.00	0.18	0.047	0.18	0.43	0.00
50 000	10	0.00	3.11	7.65	2.62	8.37	0.00
75 000	10	0.00	6.83	17.37	5.93	18.83	0.00
100 000	10	0.02	12.23	39.72	13.16	42.62	0.00
1 000 000	10	0.08	-	-	-	-	0.35
1 250 000	10	0.10	-	-	-	-	0.44
1 500 000	10	0.13	-	-	-	-	0.60
1 750 000	10	0.14	-	-	-	-	0.70
10 000 000 (max)	10	1.06	-	-	-	-	3.79

Results Analysis		
	A	B
Adaptability	**Adaptable**: Minimal increase in time for ascending inputs suggests A is adaptable.	**Not adaptable**: The sudden significant increase in time for ascending inputs suggests B is not adaptable.
Average Time Complexity	$O(n^2)$	A+D: $O(n^2)$ R: O(nlog(n)
Elimination:	A is bubble sort or insertion sort.	B is one of the quicksorts.

Program A: Bubble Sort OR Insertion Sort?	
Input	Average Time
10 aaa, 1 aaa..9 aaa	0.01
100 aaa, 1 aaa.. 99 aaa	0.02
1000 aaa, 1 aaa..999 aaa	0.01
Results Analysis	
A is bubble sort: Since there is no change in time as input size increases, program A is bubble sort.	

Program B: Which Quick Sort?	
Ascending input = not median-of-3 = not randomised	The $O(n^2)$ for the ascending inputs suggests B is not median-of-3 because the exact middle value would have been chosen each time, which is the best pivot, and we would have seen O(nlog(n)) complexity. If it was randomised, the ascending column would have a much less significant gradient in time increase as there is a much less probability of choosing the smallest or largest value as the pivot.
Random Input = naive quick sort	Random input is the best case scenario for naive quick sort, as choosing the first element would only have a 2/n probability of choosing the worst pivot (the highest or lowest value). This would be close to O(nlog(n)) complexity which is seen in the table

Conclusions

On the basis of our experiments and our analysis above, we believe that

- sortA implements the *Bubble Sort* sorting algorithm
- sortB implements the *Naive Quicksort* sorting algorithm