COMP2521 Sort Detective Lab Report

by Yiming He (z5528914)

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 input: random input, sorted input and reverse input.

We used these test cases because: different size of random input can test the time complexity of the sorting algorithms. Sorted input can test the algorithms are adaptive or not. Reverse input can be the worst case for some algorithms.

Because of the way timing works on Unix/Linux, it was necessary to repeat the same test multiple times in order to get average data and make sure the sort is stable.

We also investigated the stability of the sorting programs by using lots of data with equal keys.

We also investigated the adaptability of the sorting algorithms by sorted input.

Experimental Results

For Program A, we observed that when the size of input double, the sorting time become four times of previous sorting time, so I find the time complexity should be O(n^2). I also noticed when sortA is sorting sorted input, the time is always 0.00, so I found sortA is adaptive. Then, I use a bunch of data which has the same key and its output shows no difference with the input, so we found it is stable. And sortA dealing sorted input really quick.

These observations indicate that the algorithm underlying the program sortA has the following characteristics:

- Average Time Complexity: O(n^2)
- Adaptive
- Stability

For Program B, we observed that when we double the input size of data, the sorting time become more than 2 times of the previous input, but less than 4 times (2^2), so we can find the time complexity is O(n*log(n)). The algorithms sorts real fast when it dealing random input, but it takes more time dealing with sorted input, so we can find it is not adaptive. We use the same trick in sortA stability testing to test sortB, and I found there have lots of differences between the origin data input and output. We can know that sortB is not stable. Takes much more time when dealing with sorted and reversed input.

These observations indicate that the algorithm underlying the program sortB has the following characteristics:

- Average Time Complexity: O(n*log(n))
- Not Adaptive
- Not Stability

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 Quick Sort` sorting algorithm

Appendix

Any large tables of data that you want to present ...

Average Time Of SortA For Different Kinds Of Output

SortA: Input Size	Random Input	Sorted Input	Reverse Input	SortB: Input Size	Random Input	Sorted Input	Reverse Input
10000	0.15s	0.00s	0.24s	10000	0.00s	0.05s	0.05s
20000	0.67s	0.00s	0.97s	20000	0.00s	0.20s	0.19s
40000	3.40s	0.00s	3.90s	40000	0.00s	0.84s	0.80s
80000	15.89s	0.00s	15.65s	80000	0.01s	3.38s	3.20s
160000	67.24s	0.00s	62.47s	160000	0.03s	13.52s	12.75s
-	-	-	-	320000	0.07s	-	-
-	-	-	-	640000	0.14s	-	-