CSCE270

Dr. Wolff

Sorting Report

Qing-Xiang Jia

Due date: 5/20/2011

There five sorting algorithms have been tested, they are: Selection Sort, Insertion Sort, Shell Sort, Heap Sort and Quick Sort. Theoretically, the first two sorting methods have the efficiency of . The next one has the efficiency of . The last two sorting methods’ efficiency is .

For the algorithms with the efficiency of , they were tested with initial array size 1000, size increment 100 and 40 points on the graph. The elements of the array are random numbers of type long. The reason for the initial array size to be 1000 is that if the size is small, it may not take a second to finish the sorting. Assume that it takes 0.00017s to sort an array with size 5 while it takes 0.00019s to sort an array with size 10. Then the two results will be considered as 0.0002 when the accuracy is 4 digits beyond the decimal point. In the worst case, it may be considered 0.0 when the accuracy is low. In this situation, the result may have relatively higher inaccuracy compared with the larger array size. For the same reason, the increment of the array is 100. The number of the points is 40. For these three values, the higher the values, the better results can be obtained. But it is equally important to consider the speed of the test program.

The figure below is the graph of Selection Sort.

Theoretically it should be a polynomial, to be more specific, quadratic function. According to the result of the experiment, it is a quadratic function because the equation of the trendline is a quadratic function and the trendline perfectly fits the points. The coefficient of determination is 1 which means the trendline matches the outcome of the experiment entirely.

For the experiment of Insertion Sort, the figure is shown below.

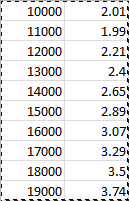
In the theory, it should be quadratic function. From the experiment with the coefficient of determination equal to 0.9997, it is plausible to claim that the outcome justifies the theory. The first two experiments have exactly the same initial array size, increment of the array size and number of the points. The 0.0003 error may be due to at least three causes. First of all, the elements of the array are randomly generated. However, it is possible that the random numbers are not evenly random. Another cause would be that the timer is implemented by System.currentTimeMillis() and it is not reliably accurate. The last cause maybe comes from the computer itself. The high temperature of the hardware may slightly slow down the computer. As a result, parts of the data were processed with normal speed while another part of them were processed with lower speed. Hence the inconsistency formed.

In the experiment for Shell Sort the initial array size is 10000, and the increment of the array size is 1000. The quantity of the points on the graph is 50.

From the graph, the equation of the trendline is a polynomial and the coefficient determination is 0.9987. The value is close to 1 enough to demonstrate that the experimental outcome fits the theoretical ones. The possible causes of the error are already mentioned in the report of the first two experiments.

For the last two experiments, the initial array size, increment of the array size and the number of the points are the same as those used in the experiment of Shell Sort. The outcomes of Heap Sort and Quick Sort are shown below.

The results look like linear but it essentially is not linear because from the theory, both Heap Sort and Quick Sort belong to the category of . There is no trendline for it but with a little bit manipulation, linear trandline can be used to demonstrate the theory.

­­

The colum shown above is part of the result from the experiment. The left colum represents the size of the array, also known as n. The right colum represents the time the program took to sort the colum, known as . The manipulation is to create a new colum with the values from the left colum mulplied by the logarithm of the values. The excel expression for this operation is $A1\*LOG(A1,2). Then select the new colum and the right colum. If the theory is true, the trendline should be linear. The reason is that they are both . From the two graphs, the trendlines are linear and the coefficient determination for Heap Sort is 0.9979 and 0.999 for Quick Sort. So they demonstrate the theory. The causes of the error are already discussed in the first two experiments.

**Creativity Part**

1. Selection Sort and Insertion Sort

From the graph, Insertion Sort is faster than Selection Sort especially when the size of the array is large. The operation of insertion only happens when the elements are unsorted or parts of the elements are unsorted. However, in Selection Sort, the operation happens no matter whether the elements are partially sorted or entirely sorted. So there are best case (in which all elements are sorted) and worst case (in which all elements are unsorted). It was tested by random data, so for Insertion Sort, the efficiency is between best case and worst case. For Selection Sort, each case can be considered as worst case so it has the lowest efficiency.

1. The experimental results are displayed below.

Both Heap Sort and Quick Sort are recursive methods. For heap, it first builds a tree then does deletion. Each time of the deletion, it will remove the largest item. In this way, it sorts data. For Quick Sort, it first picks an element as pivot and recursively splits an array of data into two parts. One part of data is larger than the pivot and another part of data is less than the pivot. For heap, its efficiency is while the efficiency of Quick Sort is for most of time and occasionally it is . From the theory, it seems that on average Heap Sort should be faster than Quick Sort. However, from the outcome of the experiment, although both of them fit the category of , Heap Sort is slower than Quick Sort. On Wikipedia[[1]](#footnote-1) (http://en.wikipedia.org/wiki/Heapsort), it claims that Quick Sort is faster than Heap Sort because of “its better cache performance and other factors.”

1. "Heapsort." Wikipedia. Web. 20 May 2011. <http://en.wikipedia.org/wiki/Heapsort>. [↑](#footnote-ref-1)