Homework03 – Profiling (or Performance Measurement)

Due 11:55 PM Wednesday, March 26, 2014, 3 points(+ 2 optional extra work), Group assignment

Overview

The textbook presents algorithms and analysis of the Selection sort in Example 1.22. and QuickSort Program 7.7, respectively. This assignment seeks to verify empirically the accuracy of those analysis's by measuring performance of each algorithm under specific conditions. Performance measurement or profiling program execution provides detailed empirical data on algorithm performance at different levels of granularity and measures.

The assignment uses a timing program suggested by the textbook to acquire runtimes of the two sorting algorithms under varying problem sizes and initial orderings (sorted or random).

Purpose of Assignment

- Provide exposure to the performance measurement or program profiling.
- Compare the actual running times of two different sorting algorithms.[wish]
- Use algorithm analysis to predict actual running time.

Assignment Software

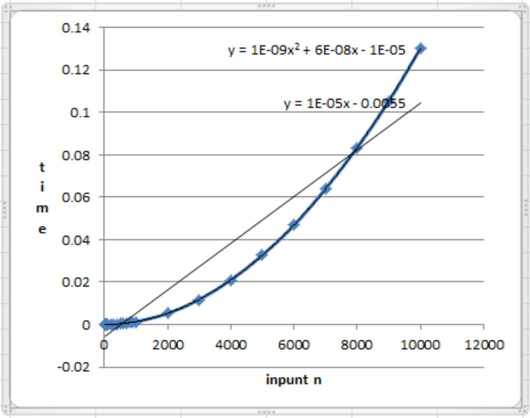
Visual Studio comes with "Performance Profiler" which does what we are trying to do. It is not available in VS Express version. Therefore we will use the elapsed times printed by program execution, even though these times may not be as accurate as a profiler.
With small input data size, all times will likely be 0.0000 because the clock interval is too large to measure the execution times. In that case you will not be able to analyze the output to answer the following questions. You should try to get sufficiently accurate results with various data sets and/or extra lines of code repetitions as described in the textbook **Program1.25.** Our focus on this assignment is to compare and prove the time complexity of sorting algorithms, not the performance itself.

Assignments

1. Read the text related with this topic in Section 1.6.
2. Let's estimate the order of growth of the running time as a function of n using Figure 1.11. Safely, we can assume that the running time obeys a power law $\mathrm{T(n)} \approx a\, n^{\boldsymbol{b}}$ since the selection sort is $\mathrm{O}(n^2)$.
   - For your answer, **compute or estimate** the constant **b** using data from different input sizes of n in Figure 1.11. You must show how you get your answers. I recommend that you use a calculator and compute up to two digits after the decimal separator, e.g., 2.26, 1.46.
3. Complete implementation of the Program 1.24 and 1.25 for selection sort and timing. Of course, you must understand the program thoroughly.
   - Display the running times for n starting at 10 increment by 10 up 100, increment by 100 up to 1,000, and increment by 1,000 to 10,000. You may keep the repetition count in the program, but do not print it which makes you confused.

- Once it seems working, save the output by redirecting it (n and the running times) to a file. **You must specify the output file name as a command argument** to save the result. (e.g. > selectionSortProfile.txt)
- Get the data from selectionSortProfile.txt and **plot** the output using an Excel spread sheet. **Place** a best fit curve in polynomial (second order) and its equation estimated as shown below. (I also put a linear line as well for a test.)

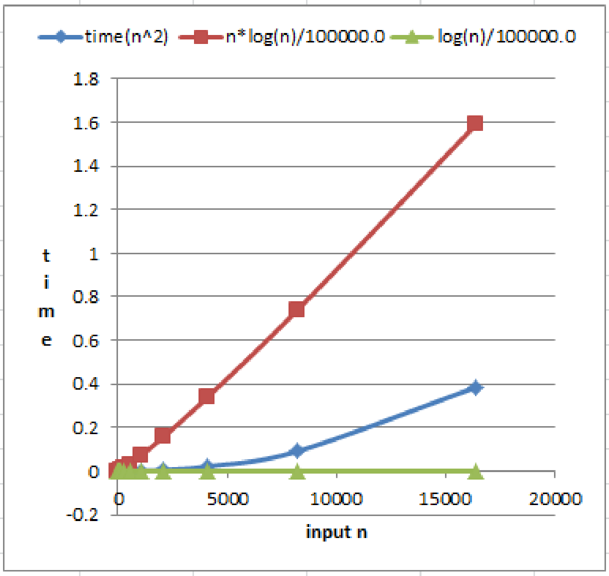
	A	B
1	n	time
2	10	0
3	20	0.000001
4	30	0.000002
5	40	0.000003
6	50	0.000004
7	60	0.000005
8	70	0.000007
9	80	0.00001
10	90	0.000012
11	100	0.000014
12	200	0.000055
13	300	0.000122
14	400	0.000214
15	500	0.000335
16	600	0.000479
17	700	0.000646
18	800	0.00084
19	900	0.001063
20	1000	0.001309
21	2000	0.005246
22	3000	0.011756



4. Let's assume that you implemented QuickSort algorithm and ran it for the performance measurement. It will be something like below.

F22

	A	B	C	D
1	n	time(n^2)	n*log(n)/100000.0	log(n)/100000.0
2		SelectionSort	QuickSort(average)	?
3	1	0	0	0
4	2	0	0.000014	0.000007
5	4	0	0.000055	0.000014
6	8	0	0.000166	0.000021
7	16	0	0.000444	0.000028
8	32	0.000002	0.001109	0.000035
9	64	0.000006	0.002662	0.000042
10	128	0.000023	0.006211	0.000049
11	256	0.000089	0.014196	0.000055
12	512	0.000364	0.03194	0.000062
13	1024	0.001427	0.070978	0.000069
14	2048	0.005464	0.156152	0.000076
15	4096	0.022022	0.340696	0.000083
16	8192	0.09075	0.738174	0.00009
17	16384	0.384667	1.589913	0.000097
18				



This time, run the SelectionSort with different input size increments by the power of 2 such as 1, 2, 4, 8, 16, … 16384 and so on. **Create** an Excel table as shown while you run the SelectionSort. Once it is running as expected, then redirect the output to a file, not to a terminal as described above. Since we have not studied the quicksort, we simply compute its QuickSort performance, O(n*log(n)), by computing n*log(n) and divide by 100,000.0. Our focus is to compare the **growth rate** of two algorithms.

5. As you did in 2 above using Figure 1.11, compute or estimate the order of growth rate **b** in $\mathrm{T(n)} \approx a\, n^{\boldsymbol{b}}$ of the running time as a function of n using SelectionSort in the Excel table you created in 4. This time, you may have enough data set such that you can increase the accuracy of the value of **b**. It is very convenient to use a pair of data set something like n = 8192, n = 16384 together. Why?

Checklist

Note: *a significant portion of the grade for this project depends on what you write in response to these questions and instructions.*

1. Include all your work in a word document, hwp or pdf file. You may use "screen capture" or "cut and paste" to include your necessary code, data, graph and so on.
   Don't turn in your source code.
2. Even though this is a group assignment, each one of you is responsible for turning it in with your partner clearly credited.
3. Include what you are asked to do in ***Assignment questions in 2, 3, 4, and 5.*** Specially pay attention to questions underlined and including the followings;
   Make observations and explanations of:
   - For each graph, discuss and explain why you see what you see in the graphs.
   - ***Output analysis:*** What surprised you the most about this homework?
   - ***Group work:*** Describe what you did and what your partner did for this assignment, how you contributed or get helped each other.
     This is the only section that could be different from your partner.
4. Name your file as HW03_StudentID_Name.
   If I can't find your file because of your filename, I can't grade it, and your grade will suffer.
5. Include the following line at the top of your every file with your name signed.
   On my honor, I pledge that I have neither received nor provided improper assistance in the completion of this assignment. Signed: ______________

Extra 2 miles (points) to go optionally:
Develop this homework further such that QuickSort is incorporated. **Theoretically**, QuickSort algorithm is known to make O(n log n) comparison to sort n items. In the worst case, it makes O(n^2) comparison. Also it can be improved how we select pivots and whether input is sorted or not. **Empirically** show the performance measurements of QuickSort such that it makes either O(n log n) or O(n^2). Things to turn in for this extra work is that a complete set of homework assignment and its answers, separately.