CS 325 Spring 2019, Homework 1

April 7, 2019

1. For each of the following pairs of function, select the best relationship from the options:
   - $f(n)$ is $O(g(n))$
   - $f(n)$ is $\Theta(g(n))$
   - $f(n)$ is $\Omega(g(n))$

   and give a brief explanation.

   a. $f(n) = n^{0.25}; g(n) = n^{0.5}$

   $$\lim_{n\to\infty} \frac{n^{0.25}}{n^{0.5}} = \lim_{n\to\infty} \frac{1}{n^{0.25}} = 0 \tag{1}$$

   $f(n)$ is $O(g(N))$

   b. $f(n) = \log n^2; g(n) = \ln n$

   $$\begin{aligned}
   \lim_{n\to\infty} \frac{\log_{10} n^2}{\log_e n} &= \lim_{n\to\infty} \frac{\log_e n^2}{\log_e 10 \cdot \log_e n} && (2)\\
   &= \log_{10} e \cdot \lim_{n\to\infty} \frac{2\log_e n}{\log_e n} && (3)\\
   &= 2\log_{10} e, \text{ which is a constant > 0} && (4)\\
   & && (5)
   \end{aligned}$$

   $f(n)$ is $\Theta(g(n))$

   c. $f(n) = n\log n + n^2; g(n) = n\sqrt{n}$

   $$\begin{aligned}
   \lim_{n\to\infty} \frac{n\log n + n^2}{n\sqrt{n}} &= \lim_{n\to\infty} \frac{n\log n}{n\sqrt{n}} + \lim_{n\to\infty} \frac{n^2}{n\sqrt{n}} && (6)\\
   &= \lim_{n\to\infty} \frac{\log n}{\sqrt{n}} + \lim_{n\to\infty} \sqrt{n} && (7)\\
   &= \lim_{n\to\infty} \frac{\log n}{\sqrt{n}} + \infty && (8)\\
   &= \infty && (9)
   \end{aligned}$$

   $f(n)$ is $\Omega(g(n))$

d. $f(n) = e^n; g(n) = 2^n$

$$\lim_{n\to\infty} \frac{e^n}{2^n} = \lim_{n\to\infty} \left(\frac{e}{2}\right)^n \tag{10}$$
$$= \infty \tag{11}$$

$f(n)$ is $\Omega(g(n))$

e. $f(n) = 2^n; g(n) = 2^{n+1}$

$$\lim_{n\to\infty} \frac{2^n}{2^{n+1}} = \lim_{n\to\infty} \frac{2^n}{2 \cdot 2^n} \tag{12}$$
$$= \frac{1}{2} \tag{13}$$

$f(n)$ is $\Theta(g(n))$

f. $f(n) = n^n; g(n) = n!$

$$\lim_{n\to\infty} \frac{n^n}{n!} = \infty \tag{14}$$

$f(n)$ is $\Omega(g(n))$

2. Let f_1 and f_2 be asymptotically positive non-decreasing functions. Prove or disprove each of the following conjectures. To disprove, give a counter example.

a. If $f_1(n) = \Theta(g(n))$ and $f_2(n) = \Theta(g(n))$ then $f_1(n) = \Theta(f_2(n))$.
Assuming $f_1(n) = \Theta(g(n))$ and $f_2(n) = \Theta(g(n))$ are true

$$\lim_{n\to\infty} \frac{f_1(n)}{g(n)} = c_1 > 0 \tag{15}$$
$$\lim_{n\to\infty} \frac{f_2(n)}{g(n)} = c_2 > 0 \tag{16}$$

We want to prove that $f_1(n) = \Theta(f_2(n))$ is true

$$\lim_{n\to\infty} \frac{f_1(n)}{f_2(n)} = \lim_{n\to\infty} \frac{f_1(n)}{g(n)} \cdot \frac{g(n)}{f_2(n)} \tag{17}$$
$$= \lim_{n\to\infty} \frac{f_1(n)}{g(n)} \cdot \lim_{n\to\infty} \frac{g(n)}{f_2(n)} \tag{18}$$
$$= \lim_{n\to\infty} \frac{f_1(n)}{g(n)} \cdot \lim_{n\to\infty} \frac{1}{\frac{f_2(n)}{g(n)}} \tag{19}$$
$$= \lim_{n\to\infty} \frac{f_1(n)}{g(n)} \cdot \frac{1}{\lim_{n\to\infty} \frac{f_2(n)}{g(n)}} \tag{20}$$
$$= c_1 \cdot \frac{1}{c_2} \tag{21}$$
$$= c_3 > 0 \tag{22}$$

Therefore, $f_1(n) = \Theta(f_2(n))$ is true

b. If $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ then $\frac{f_1(n)}{f_2(n)} = O\left(\frac{g_1(n)}{g_2(n)}\right)$
Assuming $f_1(n) = O(g_1(n))$ and $f_2(n) = O(g_2(n))$ are true

$$\lim_{n\to\infty} \frac{f_1(n)}{g_1(n)} = 0 \tag{23}$$
$$\lim_{n\to\infty} \frac{f_2(n)}{g_2(n)} = 0 \tag{24}$$

We want to prove that $\frac{f_1(n)}{f_2(n)} = O\left(\frac{g_1(n)}{g_2(n)}\right)$ is true.

$$\lim_{n\to\infty} \frac{\frac{f_1(n)}{f_2(n)}}{\frac{g_1(n)}{g_2(n)}} = 0 \tag{25}$$

Suppose:

- $f_1(n) = n$
- $f_2(n) = n^2$
- $g_1(n) = n^3$
- $g_2(n) = n^5$

Then,

$$\lim_{n\to\infty} \frac{\frac{f_1(n)}{f_2(n)}}{\frac{g_1(n)}{g_2(n)}} = \lim_{n\to\infty} \frac{\frac{n}{n^2}}{\frac{n^3}{n^5}} \tag{26}$$
$$= \lim_{n\to\infty} \frac{\frac{1}{n}}{\frac{1}{n^2}} \tag{27}$$
$$= \lim_{n\to\infty} \frac{1}{n} \cdot \frac{n^2}{1} \tag{28}$$
$$= \lim_{n\to\infty} n = \infty \tag{29}$$

Which means that $\frac{f_1(n)}{f_2(n)} = \Omega\left(\frac{g_1(n)}{g_2(n)}\right)$, which contradicts our original statement.

2. Merge Sort vs Insertion Sort Running Time Analysis

   Now that you have verified that your code runs correctly using the data.txt input file, you can modify the code to collect running time data. Instead of reading arrays from the file data.txt and sorting, you will now generate arrays of size n containing random integer values from 0 to 10,000 to sort. Use the system clock to record the running times of each algorithm for ten different values of n for example:

   $n = 5000, 10000, 15000, 20,000, ..., 50,000$

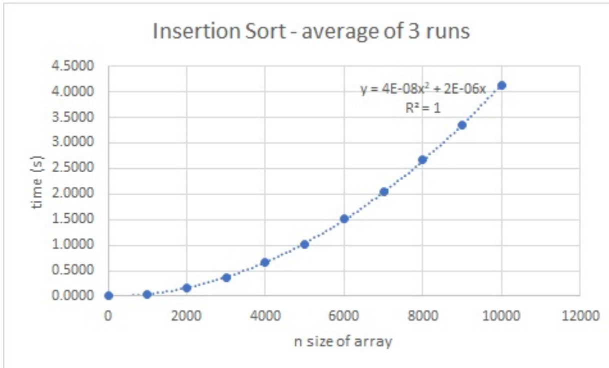
   You may need to modify the values of n if an algorithm runs too fast or too slow to collect the running time data (do not collect times over a minute). Output the array size n and time to the terminal. Name these new programs insertTime and mergeTime.

   Collect running times - Collect your timing data on the engineering server. You will need at least eight values of t (time) greater than 0. If there is variability in the times between runs of the same algorithm you may want to take the average time of several runs for each value of n. Create a table of running times for each algorithm.

n	Merge Sort	Insertion Sort
0	0.0000	0.0000
1000	0.0033	0.0433
2000	0.0100	0.1600
3000	0.0200	0.3733
4000	0.0200	0.6633
5000	0.0267	1.0267
6000	0.0300	1.5167
7000	0.0400	2.0400
8000	0.0467	2.6600
9000	0.0500	3.3500
10000	0.0533	4.1400

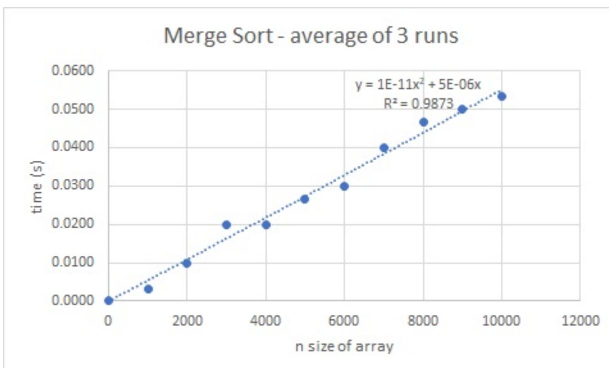
Running Time of Insertion Sort and Merge Sort.

Plot data and fit a curve - For each algorithm plot the running time data you collected on an individual graph with n on the x-axis and time on the y-axis. You may use Excel, Matlab, R or any other software. What type of curve best fits each data set? Give the equation of the curves that best "fits" the data and draw that curves on the graphs.



Insertion Sort.

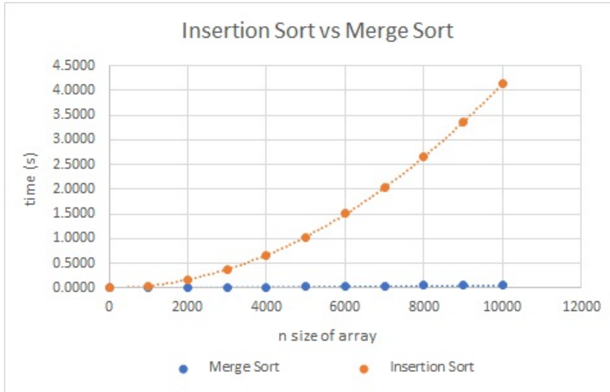
The curve that best fits the data set is a second degree polynomial.



Merge Sort.

The curve that best fits the data set is a first degree polynomial.

Combine - Plot the data from both algorithms together on a combined graph. If the scales are different you may want to use a log-log plot.



Insertion Sort vs Merge Sort.

Comparison - Compare your experimental running times to the theoretical running times of the algorithms? Remember, the experimental running times were the "average case" since the input arrays contained random integers.

The experimental running time of insertion sort and merge sort closely resembles the theoretical running time. Insertion Sort has a theoretical average case of $O(n^2)$. Merge Sort has a theoretical average case of $O(n \log n)$. n^2 grows a lot faster than $n \log n$.