CS 395 Homework 1

Colby Blair

January 21st, 2011

Due January 25th, 2012
Grade: ____________________

PROBLEMS

1.

First, let run times for **insertion sort** and **merge sort** be denoted by T_i and T_m, respectively. $T_i = 8n^2$, and $T_m = 64nlg(n)$, where $lg(n) = log_2(n)$. We can then find where T_i beats T_m by using them in an equality:

$$
\begin{aligned}
T_i = T_m & & (1)\\
=> \quad 8n^2 = 64nlog_2(n) & & (2)\\
=> \quad 8n^2 = 64\frac{ln(n)}{ln(2)} & \quad \text{Definition of Natural Logarithm} & (3)\\
=> \quad n = 8\frac{ln(n)}{ln(2)} & & (4)\\
=> \quad \frac{n}{ln(n)} = \frac{8}{ln(2)} & & (5)\\
=> \quad n = e^{-W(-\frac{ln(n)}{8})} & \quad \text{Product Log Function} & (6)
\end{aligned}
$$

This may be a bit complicated, as I am no expert with the Product Log Function. But the values of n where $T_i = T_m$ is ≈ 43.5593, as seen below on a graph:

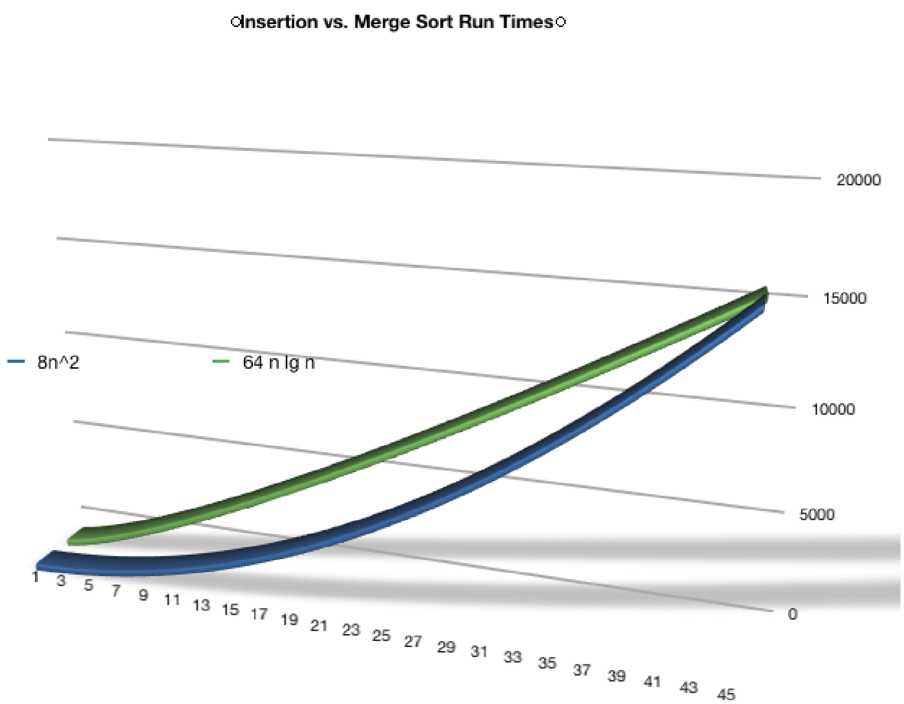

Figure 1: Insertion vs. Merge Sort Run Times

As seen by the graph, **insertion sort run time beats merge sort run time for values of** $0 < n < 43$.

2.

The psuedocode for **linear search**:

line				cost	times
1	found = false			C_1	1
2	for i = 1 to length(A)):			C_2	n
3		if A[i] == some search val:		C_3	$n-1$
4			found = true	C_4	$n-1$

We can show that the properties for a **loop invariant** hold as follows:

- **Initialization:** It can be shown that if $A = [1]$, the array is already sorted and is trivial, and the loop invariant holds prior to the first interation of the loop.
- **Maintenance:** Consider the array $A = [2, 5, 4]$ and $val = 4$. At line 2, $i = 1$. When $i == 3$, on line 3 $A[3] == 4$. We then branch to line 4.

  By Proof by Induction, since we branch into the inner 'if' statement and have the correct found value before leaving the 'for' loop, the second invariant loop property holds.
- **Termination:** Refering to the final output in the 'found' value above, the correct result exists on termination, and the third invariant loop property holds.

3.

Considering the equation $n^3/1000 - 100n^2 - 100n + 3$, to put it into Θ-notation, consider the leading term n^3. During the growth of n, n^3 grows much quicker than the rest of the terms. So $\Theta(n^3)$.

4.

Pseudo code for **selection sort** is as follows:

				cost	time
A = [a1, a2, ... an]				C_1	1
for j = 1 to n - 1				C_2	$n-1$
	min_i = j;			C_3	$n-2$
	for i = j +1 to n			C_4	$\sum_{j=1}^{n} t_j$
		if (A[i] ¡ A[min_i])		C_5	$\sum_{j=1}^{n}(t_j-1)$
			min_i = i;	C_6	$\sum_{j=1}^{n}(t_j-1)$
		i = i + 1		C_7	$\sum_{j=1}^{n}(t_j-1)$
	j = j + 1			C_8	$\sum_{j=1}^{n} t_j$
	if min_i != j			C_9	$n-2$
		swap(a, j, min_i);		C_{10}	$n-2$

The algorithm maintains the Termination property. The algorithm needs to look and check whether to swap many elements, and to compare them all, it only needs to run $(n-1)$ times.

Total time is denoted as $T(n)$:
$T(n) = C_1 + C_2(n-1) + C_3(n-2) + C_4\sum_{j=1}^{n}(t_j-1) + C_5\sum_{j=1}^{n} t_j + C_6\sum_{j=1}^{n} t_j + C_7\sum_{j=1}^{n} t_j + C_8\sum_{j=1}^{n}(t_j-1) + C_9(n-2) + C_{10}(n-2)$

To find Θ for the **worst case**, the array A is in the reverse order. Thus, $t_j = j$ for $j = 1, 2, 3...n$. Thus:

$$\sum_{j=1}^{n} j = \frac{n(n+1)}{2} - 1 \tag{7}$$

$$\sum_{j=1}^{n}(j-1) = \frac{n(n+1)}{2} \tag{8}$$

$T(n)$ can therefore be denoted as:
$T(n) = C_1 + C_2(n-1) + C_3(n-2) + C_4(\frac{n(n+1)}{2} - 1) + C_5(\frac{n(n+1)}{2})C_6(\frac{n(n+1)}{2}) + C_7(\frac{n(n+1)}{2}) + C_8(\frac{n(n+1)}{2} - 1) + C_9(n-2) + C_{10}(n-2)$
$= (\frac{C_4}{2} + \frac{C_5}{2} + \frac{C_6}{2} + \frac{C_7}{2} + \frac{C_8}{2})n^2 + (C_2 + C_3 + \frac{C_4}{2} + \frac{C_5}{2} + \frac{C_6}{2} + \frac{C_7}{2} + \frac{C_8}{2} + C_{10})n + (C_1 - C_2 - 2C_3 - C_4 - C_8 + 3C_9 - 2C_{10})$
$= an^2 + bn + c$

To find Θ, we take the leading term, or an^2. Getting rid of the constant a, we find that Θ is $\Theta(n^2)$.

To find the **best case**, find that $t_j = 1$ for $j = 1, 2, 3, ...n$. Therefore:
$T(n) = C_1 + C_2(n-1) + C_3(n-2) + C_4(n-2) + C_8(n-2) + C_9(n-2) + C_{10}(n-2)$
$= (C_2 + C_3 + C_4 + C_8 + C_9 + C_{10})n + (C_1 - C_2 - 2C_3 - 2C_4 - 2C_8 - 2C_9 - 2C_{10})$
$= an + b$

To find Θ, we take the leading term, or an. Getting rid of the constant a, we find that Θ is $\Theta(n)$.