Solutions to CS / MCS 401 Week #1 Exercises (Spring 2008)

Exercise 1.2-2 (page 13)

We want the positive integer values of n for which $8n^2 < 64n\lg(n)$, or equivalently for which $n < 8\lg(n)$, or for which $n - 8\lg(n) < 0$.

If $f(x) = x - 8\lg(x)$, then $f'(x) = 1 - 8\lg(e) / x$. So $f'(x) < 0$ for $x < 8\lg(e) \approx 11.54$, and $f'(x) > 0$ for $x > 11.54$. In other words, $f(x)$ decreases for $1 < x < 11.54$ and increases for $x > 11.54$. We know that $f(1) = 1 > 0$ but $f(2) = 2 - 8\lg(2) = 2 - 8 = -6 < 0$. This tells that, for some integer r, $f(n) < 0$ for $n = 2, 3, ..., r$ and $f(n) > 0$ for $n \geq r$. We can find r by binary search.

n	$n - 8\lg(n)$
2	–6
64	16
32	–8
48	3.320
40	–2.575
44	0.325
42	–1.139
43	–0.410

Thus $f(n)$ is negative for $\boldsymbol{n = 2, 3, \ldots, 43}$ and positive for all larger integers n. (Also, $f(n)$ is negative for $n = 1$.)

So under the assumptions of this problem, insertion sort beats merge sort when $n \leq 43$. (Note, however, that "beats" means performs fewer steps. The time per step might differ for the two algorithms. The results for $n = 1$ aren't significant.)

Exercise 1-1 (page 13)

	1 minute = 6.0×10^7 μs	**1 day = 8.64×10^{10} μs**	**1 year = 365 days = 3.15×10^{13} μs**
$\boldsymbol{n}$	6.0×10^7	8.64×10^{10}	3.15×10^{13}
$\boldsymbol{n\lg(n)}$	2.80×10^6	2.75×10^9	7.97×10^{11}
$\boldsymbol{n^2}$	7.75×10^3	2.94×10^5	5.61×10^6
$\boldsymbol{n^3}$	391	4420	3.16×10^4
$\boldsymbol{2^n}$	25	36	44
$\boldsymbol{n!}$	11	13	15

For the $n\lg(n)$ case, you may use bisection to obtain an approximate solution to $n\lg(n) = T$, stopping when the desired degree of accuracy is reached. (I stopped after obtaining three significant figures.)

Note that, when the running time is exponential (as in 2^n and $n!$), allowing the program to run longer (or equivalently, obtaining a faster computer) doesn't help much.

Note: In exercises A and B, all multiplications are mod m. For simplicity, I have omitted the mod m.

Exercise A. $109 = (1101101)_2 = 64 + 32 + 8 + 4 + 1$.

With 6 multiplications, we can compute

$a_0 = a$,
$a_1 = a_0^2 = a^2$,
$a_2 = a_1^2 = a^4$,
$a_3 = a_2^2 = a^8$,
$a_4 = a_3^2 = a^{16}$,
$a_5 = a_4^2 = a^{32}$,
$a_6 = a_5^2 = a^{64}$.

Now $a^{125} = a^{64+32+8+4+1} = a^{64}a^{32}a^8a^4a = a_6 \cdot a_5 \cdot a_3 \cdot a_2 \cdot a_0$. So with another 4 multiplications (a total of 10) we can compute a^{109}.

Exercise B. We can compute b^5 using 3 modular multiplications like this:

$$b^5 = (b^2)^2 \cdot \mathrm{b}$$

Now starting with a, we can compute a^5 with 3 modular multiplications. Once we have a^5, we can compute $(a^5)^5 = a^{25}$ using another 3 modular multiplications. Finally, once we have a^{25}, we can compute $(a^{25})^5 = a^{125}$ using still another 3 modular multiplications. This uses 9 modular multiplications, fewer than the 11 used by the fast algorithm given in class.