CMPSC 623 Problem Set 1
by Prof. Honggang Zhang

Problem 1. Exercise 1.2-2 on page 13. In addition, how might one rewrite the merge sort pseudocode to make it even faster on small inputs?

Solution:

Insertion sort beats merge sort when $8n^2 < 64n \lg n$, so we know that $n < 8 \lg n$, and $2^{n/8} < n$, which is true when $2 \leq n \leq 43$.

Merge sort can be rewritten so that it does insertion sort on inputs of size 43 or less. The modified merge sort now takes fewer steps than the straight merge sort.

Problem 2. Rank the following functions by order of growth; that is, find an arrangement $g_1, g_2, ..., g_{23}$ of the functions satisfying $g_1 = \Omega, g_2 = \Omega(g_3), ..., g_{22} = \Omega(g_{23})$. Partition your list into equivalent classes such that $f(n)$ and $g(n)$ are in the same class if and only if $f(n) = \Theta(g(n))$.

$$(3/2)^n, (\sqrt{2})^{\lg n}, \lg^* n, n^2, n^3, \lg^2 n, \lg(n!), 2^{2^n}, n^{1/\lg n}, \lg\lg n, n \cdot 2^n, n^{\lg\lg n}$$

$$\ln n, 2^n, 2^{\lg n}, (\lg n)^{\lg n}, 4^{\lg n}, (n+1)!, \sqrt{\lg n}, n!, n, n \lg n, 1$$

Solution:

$$2^{2^n} \quad (n+1)! \quad n! \quad n \cdot 2^n \quad 2^n \quad (3/2)^n \quad (\lg n)^{\lg n} = n^{\lg\lg n} \quad n^3 \quad n^2 = 4^{\lg n}$$

$$n \lg n = \Theta(\lg(n!)) \quad n = 2^{\lg n} \quad (\sqrt{2})^{\lg n} \quad \lg^2 n \quad \ln n \quad \sqrt{\lg n} \quad \lg\lg n \quad \lg^* n \quad n^{1/\lg n} = \Theta(1)$$

Note, $(\lg n)^{\lg n} = n^{\lg\lg n}$. $n^{1/\lg n} = 2$ because $2^{\lg n} = n$.

Problem 3. Problem 2-1 on page 37.

Solution:

a)
Insertion sort takes $\Theta(k^2)$ time per k-element list in the worst case. Therefore, sorting n/k such k-element lists takes $\Theta(k^2 n/k) = \Theta(nk)$ worst-case time.

b)

The key idea is to merge the lists pairwise, then merge the resulting lists pairwise, and so on, until there is just one list. The pairwise merging requires $\Theta(n)$ work at each level, since we are sill working on n elements, even if they are partitioned among sublists. The number of levels, staring with n/k k-element lists and finishing with 1 n-element list, is $\lceil \lg(n/k) \rceil$. Therefore, the total running time for the merging is $\Theta(n \lg(n/k))$.

c)
The modified algorithm has the same asymptotic running time as standard merge sort when

$$\Theta(nk + n \lg(n/k)) = \Theta(n \lg n).$$

The largest asymptotic value of k as a function of n that satisfies this is $k = \Theta(\lg n)$.

k can't be more than $\Theta(\lg n)$ or the left-hand expression would not be $\Theta(n \lg n)$ (because it would have a higher-order term than $n \lg n$). So all we need to do is verify that $k = \Theta(\lg n)$ works, which we can do by plugging $k = \lg n$ into

$$\Theta(nk + n\lg(n/k)) = \Theta(nk + n\lg n - n\lg k)$$

to get

$$\Theta(n\lg n + n\lg n - n\lg\lg n) = \Theta(2n\lg n - n\lg\lg n)$$

Then by taking just the high-order term and ignoring the constant we can get $\Theta(n \lg n)$.

d)
In practice, k should be the largest list length on which insertion sort is faster than merge sort.

Problem 4. Exercise 3.1-1 on page 50.

Solution:

Let $h(n) = \max\{f(n), g(n)\}$. Then

$$h(n) = \begin{cases} f(n) & \text{if } f(n) \geq g(n) \\ g(n) & \text{if } f(n) < g(n) \end{cases}$$

As $f(n)$ and $g(n)$ are asymptotically non-negative, there exists n_0 such that $f(n) \geq 0$ and $g(n) \geq 0, \forall n \geq n_0$. Thus, for $n \geq n_0, f(n) + g(n) \geq f(n) \geq 0, f(n) + g(n) \geq g(n) \geq 0$. Because $h(n)$ is either $f(n)$ or $g(n)$, we have

$$f(n) + g(n) \geq h(n) \geq 0$$

which shows that

$$h(n) = \max(f(n), g(n)) = O(f(n) + g(n)).$$

Similarly, since $h(n)$ is the larger of $f(n)$ and $g(n)$, we have $\forall n \geq n_0, 0 \leq f(n) \leq h(n)$ and $0 \leq g(n) \leq h(n)$. Thus, we have

$$0 \leq f(n) + g(n) \leq 2h(n)$$

or

$$0 \leq 1/2 \cdot (f(n) + g(n)) \leq h(n)$$

This means that $h(n) = \Omega(f(n) + g(n))$, with constant $c = 1/2$ in the definition of Ω.

$h(n) = \Omega(f(n) + g(n))$ and $h(n) = O(f(n) + g(n))$ indicates that $h(n) = \Theta(f(n) + g(n))$.

Proved.

Problem 5. Exercise 3.1-3 on page 50.

Solution:

Let the running time be $T(n)$. $T(n) \geq O(n^2)$ means that $T(n) \geq f(n)$ for some function $f(n)$ in the set $O(n^2)$. This is true for any $T(n)$ since the function $g(n) = 0, \forall n$ is in $O(n^2)$ and $T(n)$ is always larger than 0. So this statement tells us nothing about the running time.

Problem 6. Exercise 3.1-8 on page 50.

Solution:

$$\begin{aligned}\Omega(g(n,m)) = \{f(n,m) \quad &: \text{there exist positive constants} \quad c, n_0, \quad \text{and} \quad m_0 \\ &\text{such that} \quad 0 \le cg(n,m) \le f(n,m) \\ &\text{for all} \quad n \ge n_0 \quad \text{and} \quad m \ge m_0.\}\end{aligned}$$

$$\begin{aligned}\Theta(g(n,m)) = \{f(n,m) \quad &: \text{there exist positive constants} \quad c_1, c_2, n_0, \quad \text{and} \quad m_0 \\ &\text{such that} \quad 0 \le c_1 g(n,m) \le f(n,m) \le c_2 g(n,m) \\ &\text{for all} \quad n \ge n_0 \quad \text{and} \quad m \ge m_0.\}\end{aligned}$$