Massachusetts Institute of Technology
6.046J/18.410J: Introduction to Algorithms
Professors Piotr Indyk and Madhu Sudan

Handout 8
February 15, 2001

Problem Set 1 Solutions

Problem 1-1. Ranking Functions by Order of Growth

The ranking is based on the following facts:

- exponential functions grow faster than polynomial functions, which grow faster than logarithmic functions;
- the base of a logarithm does not matter asymptotically, but the base of an exponential and the degree of a polynomial do matter.
- Stirling's approximation for $n!$ is useful for dealing with factorials asymptotically.

The ranking is:

$\left(\frac{2}{3}\right)^n$	$(\lg n)!$
1	$n^{\lg\lg n}$
$\lg^* n$	$n^{\lg n}$
$\lg\lg n$	$\left(\frac{3}{2}\right)^n$
$\{\ \ln n\ ,\ \lg n^2\ \}$	e^n
$\left(\sqrt{2}\right)^{\lg n}$	2^{2n}
$\{\ n\ ,\ 2^{\lfloor \lg n \rfloor}\ ,\ 2^{\lceil \lg n \rceil}\ \}$	$(n-1)!$
$\{\ n\lg n\ ,\ \lg(n!)\ \}$	$n!$
$4^{\lg n}$	$(2n)!$
$\{\ n^3\ ,\ n^2+n^3\ \}$	2^{2^n}

Some notes:

- $\left(\frac{2}{3}\right)^n$ approaches zero.
- $n = 2^{\lg n} \geq 2^{\lfloor \lg n \rfloor} \geq 2^{\lg n - 1} = 2^{-1}2^{\lg n} = \frac{1}{2}n$ and so $2^{\lfloor \lg n \rfloor} = \Theta(n)$. Similarly, $n \leq 2^{\lceil \lg n \rceil} \leq 2n$ and so also $2^{\lceil \lg n \rceil} = \Theta(n)$.
- $\left(\sqrt{2}\right)^{\lg n} = 2^{\frac{1}{2}\lg n} = n^{\frac{1}{2}} = \sqrt{n}$ and similarly $4^{\lg n} = 2^{2\lg n} = n^2$.
- $\lg(n!) \approx \lg\left(\sqrt{2\pi e}\left(\frac{n}{e}\right)^{n+\frac{1}{2}}\right) = \Theta(n\lg n)$.
- $(\lg n)! \approx \sqrt{2\pi e}\left(\frac{\lg n}{e}\right)^{\lg n + \frac{1}{2}} = \sqrt{2\pi}\frac{(\lg n)^{\lg n}\sqrt{\lg n}}{n^{\lg e}}$. Now notice that $(\lg n)^{\lg n} = \left(2^{\lg\lg n}\right)^{\lg n} = 2^{\lg\lg n \lg n} = \left(2^{\lg n}\right)^{\lg\lg n} = n^{\lg\lg n}$. Using this we get: $(\lg n)! \approx \sqrt{2\pi}n^{\lg\lg n - \lg e}\sqrt{\lg n}$. This is asymptotically greater than any polynomial in n, since for any polynomial n^d, there is some n for which $\lg\lg n - \lg e > d$. However, it is strictly less than $n^{\lg\lg n}$ since they differ by $\frac{n^{\lg e}}{\sqrt{\lg n}} = \omega(1)$.

Problem 1-2. Asymptotic Notation

(a) **Sometimes true**: For $f(n) = n$ it is true, while for $f(n) = 1/n$ it is not true. (The statement is always true for $f(n) = \Omega(1)$, and hence for most functions with which we will be working in this course, and in particular all time and space complexity functions).

(b) **Sometimes true**: For $f(n) = 1$ and $g(n) = \|n * sin(n)\|$ it is true, while for any $f(n) = O(g(n))$, e.g. $f(n) = g(n) = 1$, it is not true.

(c) **Always true**: $\max(f(n), g(n)) \le f(n) + g(n) \le 2\max(f(n), g(n))$.

(d) **Always true**: Consider $f(n) + g(n)$ where $g(n) = o(f(n))$ and let c be a constant such that $g(n) < cf(n)$ for large enough n. Then $f(n) \le f(n) + g(n) \le (1 + c)f(n)$ for large enough n.

(e) **Never true**: If $f(n) = \Omega(g(n))$ then there exists positive constant c_Ω and n_Ω such that for all $n > n_\Omega$, $cg(n) \le f(n)$. But if $f(n) = o(g(n))$, then for any positive constant c, there exists $n_o(c)$ such that for all $n > n_o(c)$, $f(n) < cg(n)$. If $f(n) = \Omega(g(n))$ and $f(n) = o(g(n))$, we would have that for $n > \max(n_\Omega, n_o(c_\Omega))$ it should be that $f(n) < c_\Omega g(n) \le f(n)$ which cannot be.

Problem 1-3. Insertion sort on small arrays in merge sort

(a) Insertion sort takes $O(k^2)$ time per k-element list. Therefore, sorting n/k such k-element lists, takes $O(k^2 n/k) = O(nk)$ time.

(b) Merging requires $O(n)$ work at each level, since we are still working on n elements, even if they are partitioned among sublists. The number of levels, starting 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 $O(n \lg(n/k))$.

(c) The largest asymptotic value of k, for which the modified algorithm has the same asymptotic running time as standard merge sort, is $k = O(\lg n)$. The combined running time is $O(n \lg n + n \lg n - n \lg \lg n)$, which is $O(n \lg n)$. For larger values of k the term nk would be larger than $O(n \lg n)$.

(d) In practice, k should be chosen such that it minimizes the running time of the combined algorithm, which is $cnk + dn \lg(n/k)$, where c and d are some positive constants that determine the actual running time of the two phases. The value $k = d/c$ minimizes the above expression, and thus the best choice of k does not depend on n, but rather on the ratio between the constants d and c.

Problem 1-4. Recurrences

(a) $T(n) = 4T(\frac{n}{2}) + n^3 \Longrightarrow T(n) = \Theta(n^3)$
Using the third case of the master theorem:

$$n^3 = \Omega\left(n^{\log_2 4} + 0.1\right) = \Omega(n^{2.1})$$

(b) $T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + n \Longrightarrow T(n) = \Theta(n)$
Indeed

$$T(n) = T(\frac{n}{2}) + T(\frac{n}{3}) + n \geq n$$

so $T(n) = \Omega(n)$. Now let $c > 6$ and suppose by induction that $T(k) \leq ck\ \forall k < n$ (the base case $T(1) = 1 < c$ is true). Now

$$T(n) \leq c\frac{n}{2} + c\frac{n}{3} + n = (1 + \frac{5c}{6})n \leq cn$$

given the assumption on c. So we have $T(n) = O(n)$ as well.

(c) $T(n) = 3T(n^{\frac{1}{3}}) + \log_3 n \Longrightarrow T(n) = \Theta(\lg n \lg\lg n)$
Let $n = 3^k$ we have $T(3^k) = 3T(3^{\frac{k}{3}}) + k$. Let $S(k) = T(3^k)$, $S(1) = 1$ and $S(k) = 3S(\frac{k}{3}) + k$ so the second case of the master theorem ($k = \Theta(k^{\log_3 3})$) gives $S(k) = \Theta(k \lg k)$. Going back to n we get $T(n) = S(\log_3 n) = \Theta(\log_3 n \lg \log_3 n) = \Theta(\lg n \lg\lg n)$.

(d) $T(n) = T(n-1) + n^4 \Longrightarrow T(n) = \Theta(n^5)$
By iteration $T(n) = \sum_{i=1}^{n} i^4$. Then use approximation by integrals (page 50 in textbook).

(e) $T(n) = T(n/2 + 5) + n^2 \Longrightarrow T(n) = \Theta(n^2)$ It is reasonable to guess that $T(n)$ has the same solution of $S(n) = S(n/2) + n^2$ as the difference between $n/2$ and $n/2 + 5$ becomes negligible for $n \to \infty$. By case 3 of the Master Theorem, we have that $S(n) = \Theta(n^2)$.
Thus we guess $T(n) = \Omega(n^2)$ and we prove it by substitution. Assume that $T(m) \geq cm^2$ for an appropriate constant c and for all $m < n$. Then we have that

$$T(n) \geq c(n^2/4 + 5n + 25) + n^2.$$

With simple algebraic manipulations we have $c(n^2/4 + 5n + 25) + n^2 \geq cn^2$ is equivalent to $(1 - 3/4c)n^2 + c(5n + 25) \geq 0$. Since $c(5n + 25)$ is always positive, it will be enough to find a value of c for which $(1 - 3/4c)n^2$ is positive. Any $c < 4/3$ makes the inequality true (notice that it may be true also for some $c > 4/3$). Hence we have $T(n) = \Omega(n^2)$.
Now, we guess $T(n) = O(n^2)$ and we prove it by substitution. Assume that $T(m) \leq am^2$ for an appropriate constant a and for all $m < n$. Then

$$T(n) \leq a(n^2/4 + 5n + 25) + n^2.$$

Thus $T(n) \leq an^2$ whenever $a(n^2/4 + 5n + 25) + n^2 \leq an^2$, which is equivalent to $a(n^2/4 + 5n + 25) \leq a(n^2 - 1)$. This inequality is true, given any value of a, for n sufficiently large, e.g. for $n \geq 100$. Thus, we have proved that, for any constant a, $T(n) \geq an^2$, for $n \geq 100$. Hence $T(n) = O(n^2)$.

Combining $T(n) = O(n^2)$ and $T(n) = \Omega(n^2)$, we get $T(n) = \Theta(n^2)$.

(f) $T(n) = 2T(n/2) + n \lg n \Longrightarrow T(n) = n \lg^2 n$
Unfortunately, the Master Method cannot be used since the added value $n \lg n$ is within a logarithmic factor of $n^{\log_2 2} = n$. Instead, by iteration:

$$\begin{aligned}
T(n) &= n \lg n + 2\frac{n}{2} \lg \frac{n}{2} + 4\frac{n}{4} \lg \frac{n}{4} + \cdots \\
&= n \left(\lg n + \lg \frac{n}{2} + \lg \frac{n}{4} + \cdots \right) \\
&= n \sum_{k=0}^{\lg n} \lg 2^k = n \sum_{k=0}^{\lg n} k \\
&= n \frac{\lg n (\lg n + 1)}{2} = \Theta(n \lg^2 n)
\end{aligned}$$

(g) $T(n) = 4T(n/2) + n^2 + n \Longrightarrow T(n) = n^2 \lg n$
This time its fine to use the second case of the Master Method, since $n^2 + n = \Theta(n^{\log_2 4}) = \Theta(n^2)$.