Massachusetts Institute of Technology
6.042J/18.062J: Mathematics for Computer Science
Professor Albert R. Meyer

Handout 28
March 16, 1998, 9:00pm

Notes for Recitation 6 (3/13/98)

1 Master Theorem

In lecture we described a "Master Theorem" which can be used to solve many recurrences of the form:

$$T(n) = aT(\lceil n/b \rceil) + f(n)$$

where $T, f : \mathbb{N} \to \mathbb{N}, a > 0$, and $b > 1$. Namely, let $\beta ::= \log_b a$. Then

$$T(n) = \begin{cases} \Theta(n^\beta) & \text{if } f(n) = O(n^{\beta-\epsilon}) \text{ for some } \epsilon > 0, \\ \Theta(n^\beta \log n) & \text{if } f(n) = \Theta(n^\beta), \\ \Theta(f(n)) & \text{if } af(n/b) \leq cf(n) \text{ for some } c < 1 \text{ and all large } n. \end{cases}$$

The third condition looks different, but it does not overlap with the others, because if $af(n/b) \leq cf(n)$ for some $c < 1$ and all large n, then

$$f = \Omega(n^{\beta+\epsilon})$$

for $\epsilon = -\log_b c$; note that $\epsilon > 0$ because $c < 1$. (We leave the proof of this claim as an exercise.) The converse, however, does not hold, so we can't replace the stronger condition used in the Master Theorem by this last condition.

Solution to the Exercise:

Assume for simplicity that n is a power of b, say $n = b^k$, and the "smoothness" condition $af(n/b) \leq cf(n)$ holds for all n.

So

$$\begin{aligned} f(n) &\geq (a/c)\, f(n/b) \geq (a/c)\left((a/c)f((n/b)/b)\right) = (a/c)^2 f\left(n/b^2\right) \\ &= \cdots = (a/c)^k f\left(n/b^k\right) = (a/c)^k f(1). \end{aligned}$$

But

$$\begin{aligned} (a/c)^k &= (b^{\log_b a - \log_b c})^k = (b^k)^{(\log_b a - \log_b c)} \\ &= n^{\log_b a - \log_b c}, \end{aligned}$$

so

$$f(n) = \Omega((a/c)^k) = \Omega\left(n^{\log_b a - \log_b c}\right) = \Omega\left(n^{\log_b a + \epsilon}\right)$$

where $\epsilon = -\log_b c$.

The Master theorem applies nicely to the common case when $f(n) = \Theta(n^\alpha)$.

Corollary 1.1 *If* $T(n) = aT(\lceil n/b \rceil) + f(n)$ *where* $a > 0, b > 1$ *and* $f(n) = \Theta(n^\alpha)$, *then*

$$T(n) = \begin{cases} \Theta(n^{\max(\alpha,\beta)}) & \text{if } \alpha \neq \beta, \\ \Theta(n^\beta \log n) & \text{if } \beta = \alpha. \end{cases}$$

Proof.

Case 1: $\beta > \alpha$.

Choose $\epsilon = \beta - \alpha > 0$. We have that $\alpha = \beta - \epsilon$; hence $f(n) = \Theta(n^\alpha) = \Theta(n^{\beta-\epsilon})$ with $\epsilon > 0$. So the first case of the Master Theorem applies, yielding

$$T(n) = \Theta(n^\beta),$$

as desired.

Case 2: $\beta = \alpha$.

Then $f(n) = \Theta(n^\beta)$, so the second case of the Master Theorem applies, yielding

$$T(n) = \Theta(n^\beta \log n),$$

as desired.

Case 3: $\alpha > \beta$.

Let's prove this case first when $f(n) = dn^\alpha$ for some $d > 0$ and all large n.

Let $c ::= a/b^\alpha = b^\beta/b^\alpha = 1/b^{\alpha-\beta}$. We know $b > 1$ and $\alpha - \beta > 0$, so $b^{\alpha-\beta} > 1$. Therefore $c = 1/b^{\alpha-\beta} < 1$. Also,

$$af(n/b) = ad(n/b)^\alpha = (a/b^\alpha)dn^\alpha = cf(n).$$

So the third case of the Master Theorem applies, yielding

$$T(n) = \Theta(f(n)) = \Theta(n^\alpha),$$

as desired.

Now suppose all we know is that $f(n) = \Theta(n^\alpha)$. Then by definition of $\Theta()$, there are constants $n_0, d_1, d_2 > 0$ such that $d_1 n^\alpha \leq f(n) \leq d_2 n^\alpha$ for all $n > n_0$. For $i = 1, 2$, let $f_i(n) = f(n)$ for $n \leq n_0$, and for $n > n_0$, let $f_i(n) = d_i n^\alpha$. So $f_1(n) \leq f(n) \leq f_2(n)$ for *all* $n \in \mathbb{N}$, and also $f_i(n) = d_i n^\alpha$ for all $n > n_0$.

Now define T_1, T_2 by the conditions: $T_i(0) = T(0), T_i(1) = T(1)$, and $T_i(n) = aT_i(\lceil n/b \rceil) + f_i(n)$ for $n > 1$. Since $f_i(n)$ equals a constant times n^α for all large n, our earlier reasoning shows that

$$T_i(n) = \Theta(n^\alpha)$$

for $i = 1, 2$. But since $f_1(n) \leq f(n) \leq f_2(n)$ for all n, it follows (by an easy induction) that

$$T_1(n) \leq T(n) \leq T_2(n),$$

which immediately implies that $T(n) = \Theta(n^\alpha)$.

$\blacksquare$

It's helpful to note that the Master Theorem and its corollary still hold if, instead of rounding n/b up to $\lceil n/b \rceil$, we rounded down to $\lfloor n/b \rfloor$, or even chose to round up or down as a function of n.

2 Examples

We saw in lecture a way to multiply two n bit numbers by performing three $n/2$-bit multiplications plus some fixed number of length n shifts and sums of n-bit numbers. Each such addition or shift takes $O(n)$ bit operations. Thus if $T_1(n)$ is the number of bit operations to multiply two n-bit numbers, we have

$$T_1(n) = 3T_1(\lceil n/2 \rceil) + f(n)$$

where $f(n) = O(n)$.

So $a = 3$, $b = 2$, $\log_b a > 1 + 0.58$. Hence $f(n) = O(n^{\log_b a - 0.58})$, and the first case of the Master Theorem applies, namely,

$$T_1(n) = \Theta(n^{\log_2 3}).$$

This is much better than the $\Theta(n^2)$ operations that ordinary gradeschool multiplication takes on some n-bit multiplications. (This idea generalizes; two n-bit numbers can be multiplied with $O(n \log n)$ bit operations.)

If $T_2(n)$ is the number of pairwise comparisons needed to sort n numbers, then

$$T_2(n) = 2T_2(\lceil n/2 \rceil) + f(n)$$

where $n - 1 \leq f(n) \leq n + 1$. So $a = 2$, $b = 2$, $\log_a b = 1$, $f(n) = \Theta(n) = \Theta(n^{\log_a b})$ and the second case of the Master Theorem applies yielding

$$T_2(n) = \Theta(n \log n).$$

Using an ingenious extension of the idea used to obtain the $O(n^{\log_2 3})$ n-bit multiplication procedure, Strassen in 1972, observed that two $n \times n$ matrices could be multiplied by performing only seven multiplications of linear combinations of their eight $n/2 \times n/2$ submatrices and then computing eight linear combinations of the seven product matrices. Since a linear combination of a fixed number of $n/2 \times n/2$ matrices requires $O(n^2)$ arithmetic operations, the number, $T_3(n)$, of arithmetic operations required by Strassen's procedure is

$$T_3(n) = 7T_3\left(\lceil n/2 \rceil\right) + O(n^2).$$

Again the first case of the Master Theorem applies with $\log_b a = \log_2 7 > 2$, so

$$T_3(n) = \Theta(n^{\log_2 7}) = O(n^{2.81}).$$

This is a considerable improvement over the standard row-by-column method of matrix multiplication, which takes $\Theta(n^3)$ operations.

Strassen's technique has been extended in various *ad hoc* ways, now it is actually known how to multiply $n \times n$ matrices with $O(n^{2.49})$ operations. However, no one believes the currently fastest method is optimal. It remains an open question whether $n \times n$ matrices can be multiplied in $\Theta(n^2)$ operations.

3 Proof of the Master Theorem

We did not go over the proof in recitation. The proof is here for the record.

All the calculations in the proof are much simpler if we assume n is a power of $b \in \mathbb{N}$, so we can ignore the details of repeatedly rounding real numbers to integers. Of course in general, neither b nor any of it powers need be an integer. Nevertheless, the conclusions we reach below when powers of b are integers can be shown to hold for arbitrary real numbers. (We shall not give that argument here, *cf.* Cormen, Leiserson, Rivest, *Introduction to Algorithms*, McGraw-Hill, 1990, §4.4.2.)

Proof. So assume $b, k \in \mathbb{N}, b, k \geq 1$, and $n = b^k$. Now we have,

$$\begin{aligned}
T(n) &= T(b^k) = aT(b^{k-1}) + f(b^k) \\
&= a(aT(b^{k-2}) + f(b^{k-1})) + f(b^k) \\
&= a^2T(b^{k-2}) + a^1 f(b^{k-1})) + a^0 f(b^k) \\
&= a^k T(1) + \textstyle\sum_{i=0}^{k-1} a^i f(b^{k-i}).
\end{aligned}$$

Since the argument above does not depend on the values of $f(j)$ for $j < b$, there's no harm in assuming that $f(1) = T(1)$, so we can say

$$T(b^k) = \sum_{i=0}^{k} a^i f(b^{k-i}).$$

Since the first term in the sum is $f(n)$ and all the terms are nonnegative, this already tells us that

$$T(n) = \Omega(f(n)).$$

Since the last term is $a^k f(1)$ and $a = b^\beta$, we also have

$$T(n) \geq a^k f(1) = (b^\beta)^k f(1) = (b^k)^\beta f(1) = n^\beta f(1),$$

so

$$T(n) = \Omega(n^\beta).$$

If we can match either of these $\Omega()$ bounds on T with $O()$ bounds, then we will have a $\Theta()$ bound. So we now consider the different cases for the behavior of f and derive the needed $O()$ bounds.

Case 1: $f(n) = O(n^{\beta-\epsilon})$ for some $\epsilon > 0$.

So

$$\begin{aligned} T(n) &= T(b^k) = \textstyle\sum_{i=0}^{k} a^i f(b^{k-i}) \\ &= O(\textstyle\sum_{i=0}^{k} a^i (b^{k-i})^{\beta-\epsilon}). \end{aligned}$$

Let $S(k)$ be this last sum, that is, $T(b^k) = O(S(k))$ where

$$S(k) ::= \sum_{i=0}^{k} a^i (b^{k-i})^{\beta-\epsilon}.$$

Let's focus on the ith term of $S(k)$. Using the fact that $a = b^\beta$ and $b^k = n$,

$$\begin{aligned} a^i (b^{k-i})^{\beta-\epsilon} &= (b^\beta)^i (b^{k-i})^{\beta-\epsilon} = b^{i\beta} b^{k\beta - i\beta - \epsilon(k-i)} \\ &= b^{k\beta - \epsilon(k-i)} = (b^k)^\beta / (b^\epsilon)^{i-k} = n^\beta (1/b^\epsilon)^{k-i}. \end{aligned}$$

So

$$\begin{aligned} S(k) &= n^\beta \textstyle\sum_{i=0}^{k} (1/b^\epsilon)^{k-i} \\ &= n^\beta \textstyle\sum_{i=0}^{k} (1/b^\epsilon)^{i} \\ &\leq n^\beta \textstyle\sum_{i=0}^{\infty} (1/b^\epsilon)^{i} \\ &= O(n^\beta) \end{aligned}$$

The final $O()$ bound above follows because $b > 1$, so $1/b^\epsilon < 1$, and therefore the infinite geometric series above is bounded by a constant. But since $T(n) = T(b^k) = O(S(k))$, we can conclude that

$$T(n) = O(n^\beta).$$

Since we already observed that $T(n) = \Omega(n^\beta)$, we have finally that

$$T(n) = \Theta(n^\beta)$$

as was claimed in this case.

Case 2: $f(n) = \Theta(n^\beta)$.

So

$$\begin{array}{rcl} T(n) & = & T(b^k) \quad = \quad \sum_{i=0}^{k} a^i f(b^{k-i}) \\ & = & \Theta(\sum_{i=0}^{k} a^i (b^{k-i})^\beta) \\ & = & \Theta(\sum_{i=0}^{k} (b^\beta)^i (b^\beta)^{k-i}) \\ & = & \Theta(\sum_{i=0}^{k} b^{k\beta}) \\ & = & \Theta(\sum_{i=0}^{k} n^\beta) \quad = \quad \Theta(kn^\beta) \quad = \quad \Theta(n^\beta \log n). \end{array}$$

Case 3: $af(n/b) \leq cf(n)$ for some $c < 1$ and all large n.

So

$$\begin{array}{lcl} f(n/b) & \leq & (c/a)f(n), \\ f(n/b^2) & \leq & (c/a)f(n/b) \quad \leq \quad (c/a)((c/a)f(n)) \quad = \quad (c/a)^2 f(n), \\ f(b^{k-i}) = f(n/b^i) & \leq & (c/a)^i f(n), \end{array}$$

providing b^{k-i} is sufficiently large. In particular, there is an $n_0 > 0$ such that the last inequality holds providing $b^{k-i} \geq n_0$. So it holds when $i \leq k - d$ where $d ::= \log_b n_0$. Therefore

$$\begin{array}{rcl} T(n) & = & T(b^k) \quad = \quad \sum_{i=0}^{k} a^i f(b^{k-i}) \\ & = & \sum_{i=0}^{k-d} a^i f(b^{k-i}) + \sum_{i=k-d+1}^{k} a^i f(b^{k-i}) \\ & \leq & \sum_{i=0}^{k-d} a^i (c/a)^i f(n) + \sum_{i=0}^{d-1} a^{k-i} f(b^i) \\ & = & f(n) \sum_{i=0}^{k-d} c^i + a^k \sum_{i=0}^{d-1} a^{-i} f(b^i) \\ & \leq & f(n) \sum_{i=0}^{\infty} c^i + n^\beta \sum_{i=0}^{d-1} a^{-i} f(b^i). \end{array}$$

But $c < 1$, so the geometric series above is bounded by a constant, and the final sum above does not depend on n. This shows that

$$T(n) = O(f(n)) + O(n^\beta).$$

But we observed above that $f = \Omega(n^{\beta+\epsilon})$, so

$$T(n) = O(f(n)).$$

Since we also observed already that $T(n) = \Omega(f(n))$, we have finally that

$$T(n) = \Theta(f(n))$$

as was claimed in this case.

■

4 Change of variable

Another useful method to make a difficult looking recurrence easier is by a change of variable. This is similar to the change of variable technique in calculus. For example, consider the recurrence

$$T(n) = 2T(\lfloor \sqrt{n} \rfloor) + \log n$$

with base condition $T(1) = 1$. Here $T(n)$ is defined for all n. Let us first note that $T(n)$ is a nondecreasing function of n.

Claim 4.1 *For all $n > 1$, $T(n) \geq T(n-1)$.*

Proof. By strong induction on n.

I.H.: $P(n) : T(n) \geq T(n-1)$.

Base: $n = 2$: $T(2) = 2T(\lfloor\sqrt{2}\rfloor) + 1 = 2T(1) + 1 = 3 > 1 = T(1)$.

Ind. Step: Assume $P(k)$ is true for all $k \leq n$. We will prove that $P(n+1)$ is true.

$$\begin{aligned} T(n+1) &= 2T(\lfloor\sqrt{n+1}\rfloor) + \log(n+1) \\ &\geq 2T(\lfloor\sqrt{n}\rfloor) + \log n \end{aligned}$$

The latter is true because either $\lfloor\sqrt{n+1}\rfloor = \lfloor\sqrt{n}\rfloor$ or we can apply $P(\lfloor\sqrt{n+1}\rfloor)$ which says that $T(\lfloor\sqrt{n+1}\rfloor) \geq T(\lfloor\sqrt{n}\rfloor)$.

But then the right hand side is $T(n)$. So $T(n+1) \geq T(n)$. ■

We will now solve the recurrence assuming n is a power of 2. First, make the substitution $n = 2^m$. The recurrence becomes

$$T(2^m) = 2T(\lfloor 2^{m/2}\rfloor) + m$$

Now define $S(m) = T(2^m)$ for all integers $m \geq 0$. Note that since T is nondecreasing, so is S. Therefore, $S(m) = T(2^m) = 2T(\lfloor 2^{m/2}\rfloor) + m \leq 2T(2^{\lceil m/2\rceil}) + m = 2S(\lceil m/2\rceil) + m$.

That is,

$$S(m) \leq 2S(\lceil m/2\rceil) + m.$$

Now to bound S, we define $S'(0) = S(0), S'(1) = S(1)$, and $S'(m) = 2S'(\lceil m/2\rceil) + m$ for $m > 1$. Now we can apply Case 2 of the Master Corollary to get $S'(m) = \Theta(m \log m)$. But $S(m) \leq S'(m)$ for all m, so the $\Theta()$ bound on S' becomes an $O()$ bound on S:

$$S(m) = O(m \log m).$$

Therefore, there is a $d > 0$ such that $T(2^m) \leq d(m \log m)$ for all large m. Equivalently, $T(n) \leq d(\log n \log\log n)$ whenever n is a large power of two. But $\log 2n = \Theta(\log n)$ and T is nondecreasing, so we can conclude that the bound of a constant times $\log n \log\log n$ holds for even for n *between* powers of two. In other words,

$$T(n) = O(\log n \log\log n).$$

What about a lower bound? This can be derived in a similar fashion.

$$S(m) = T(2^m) = 2T(\lfloor 2^{m/2} \rfloor) + m \geq 2T(2^{\lfloor m/2 \rfloor}) + m = 2S(\lfloor m/2 \rfloor) + m.$$

That is, $S(m) \geq 2S(\lfloor m/2 \rfloor) + m$.

This time define $S'(0) = S(0)$ and $S'(m) = 2S'(\lfloor m/2 \rfloor) + m$ for $m > 0$, so $S(m) \geq S'(m)$ for all m. Now we can apply the Master Corollary to solve the recurrence for S' to get $S'(m) = \theta(m \log m)$, so $S(m) = \Omega(m \log m)$. By reasoning similar to that above for the $O()$ bound, we conclude that

$$T(n) = \Omega(\log n \log \log n).$$

Combining the upper and lower bounds, we have shown

$$T(n) = \Theta(\log n \log \log n).$$

The hard part in changing variables to solve recurrences is guessing what substitution to make, but this is a skill that comes out of practice, just as in calculus.

5 Team Problems

Problem 1. **(a)** Show that the Master Theorem does not apply to

$$S(m) = S(m/2) + (\log m)^2.$$

(b) Find a $\Theta()$ bound for S anyway.

(c) Find a $\Theta()$ bound for

$$T(n) = T(\sqrt{n}) + (\lg \lg n)^2.$$

The base condition is $T(2) = 1$.

Solution:

(a) Use $a = 1, b = 2$, so $\beta = 0$, and $n^\beta = 1$. The first two cases of master theorem obviously don't apply since $f(n)$ grows much faster than $\Theta(1)$. Now let's consider the thrid case. (assume $\log m$ is base 2)

$$\begin{array}{lcl} af(\frac{m}{b}) & \leq & cf(m) \\ f(\frac{m}{2}) & \leq & cf(m) \\ (\log \frac{m}{2})^2 & \leq & c(\log m)^2 \\ (\log m - 1)^2 & \leq & c(\log m)^2 \\ (\frac{\log m - 1}{\log m})^2 & \leq & c \end{array}$$

For large n, $\frac{\log m-1}{\log m}$ becomes 1, so c must be greater than or equal to 1. Therefore, we can not use the third case of master theorem since $c < 1$ does not exist.

(b) Assume $\log m$ is base 2.

$$
\begin{aligned}
S(m) &= S(\frac{m}{2}) + (\log m)^2 \\
&= (\log m)^2 + (\log \frac{m}{2})^2 + (\log \frac{m}{4})^2 \cdots (\log 2)^2 + S(1) \\
&= \sum_{i=0}^{(\log m)-1} (\log \frac{m}{2^i})^2 + S(1) \\
&= \sum_{i=0}^{(\log m)-1} (\log m - i)^2 + S(1) \\
&= \sum_{i=1}^{\log m} i^2 + S(1) \\
&= \frac{1}{6} \log m(\log m + 1)(2\log m + 1) + 1
\end{aligned}
$$

So, $S(m)$ is $\Theta((\log m)^3)$.

(c) Substitute $n = 2^m$ and note that $m = \log n$.

$$T(2^m) = T(2^{\frac{m}{2}}) + (\log m)^2.$$

So let $S(m) = T(2^m)$. Now,

$$
\begin{aligned}
T(n) = T(2^m) = S(m) &= S(\frac{m}{2}) + (\log m)^2 \\
&= \Theta((\log m)^3) \\
&= \Theta((\log \log n)^3)
\end{aligned}
$$

So, $T(n)$ is $\Theta((\log \log n)^3)$.

Problem 2. Life and Death. In ancient Mars, cannibalism was an essential way of life. The Marons and the Batons were two species that lived for several thousand years in close contact. Every year, first all the Marons sustained themselves by first consuming Batons: two Marons ate one Baton between them. Then they reproduced: one-year-old Marons doubled in number while Marons that were two or older tripled in number. Then the surviving Batons consumed one Maron each and then all the Batons reproduced to quadruple in number. This cycle repeated every year. After 1 year there were two 1-year old Marons and two 1-year old Batons. After 2 years there were two 1-year olds and two 2-year olds in each of the two species. How many of each will there be of each species at the end of n years?

Solution: Let $g(n)$ be the number of Marons after n years, and $f(n)$ the number of Batons.

After $n-1$ years, we have $g(n-1)$ Marons and $f(n-1)$ Batons. Then in the n^{th} year, from the rules of sustenance, the population of Batons drops to $f(n-1)-(1/2)g(n-1)$. The number of 1-year old Marons is $g(n-1)-g(n-2)$. Their population goes up to $2(g(n-1)-g(n-2))$. Meanwhile the number of older Marons goes up from $g(n-2)$ to $3g(n-2)$. Then it is the Batons turn. The survivors consume a Maron each. So the population of Marons after n years is

$$g(n) = 2(g(n-1)-g(n-2)) + 3g(n-2) - (f(n-1) - \frac{1}{2}g(n-1))$$

That is,

$$g(n) = \frac{5}{2}g(n-1) + g(n-2) - f(n-1)$$

The Batons then quadruple. So the number of Batons after n years is:

$$f(n) = 4(f(n-1) - \frac{1}{2}g(n-1))$$

We know that $f(1) = g(1) = 2$ and $f(2) = g(2) = 4$. Substituting these values in the recurrences, we get $f(3) = g(3) = 8$ and $f(4) = g(4) = 16$.

We guess that $g(n) = f(n) = 2^n$. We can verify this by strong *double* induction. Note that there is only 1 variable, n, the number of years, but we will argue inductively about both $f(n)$ and $g(n)$.

I.H. : $P(n) : f(n) = g(n) = 2^n$.

Base: $n = 1$: $f(1) = g(1) = 2$ and

$n = 2$: $f(2) = g(2) = 4$.

Ind. Step.: Assume $P(k)$ is true for $1 \leq n$. To prove $P(n+1)$ consider $f(n+1), g(n+1)$.

$$f(n+1) = 4(f(n) - \frac{1}{2}g(n)) = 4(2^n - 2^{n-1})$$

$$= 4(2^{n-1}) = 2^{n+1}$$

And,

$$g(n+1) = \frac{5}{2}g(n) + g(n-1) - f(n) = \frac{5}{2}(2^n) + 2^{n-1} - 2^n = 2^{n+1}$$