CMPS 101
Algorithms and Abstract Data Types
Recurrence Relations

Iteration Method
Recall the following example from the induction handout.

Example 1

$$T(n) = \begin{cases} 0 & n = 1 \\ T(\lfloor n/2 \rfloor) + 1 & n \geq 2 \end{cases}$$

We begin by illustrating a solution technique called *iteration*, which consists of repeatedly substituting the recurrence into itself until a pattern emerges.

$$\begin{aligned} T(n) &= 1 + T(\lfloor n/2 \rfloor) \\ &= 1 + 1 + T(\lfloor \lfloor n/2 \rfloor /2 \rfloor) = 2 + T(\lfloor n/2^2 \rfloor) \\ &= 2 + 1 + T(\lfloor \lfloor n/2^2 \rfloor /2 \rfloor) = 3 + T(\lfloor n/2^3 \rfloor) \\ &\vdots \\ &= k + T(\lfloor n/2^k \rfloor) \end{aligned}$$

This process must terminate when the recursion depth k is at its maximum, i.e. when $\lfloor n/2^k \rfloor = 1$. To solve this equation for k in terms of n, we use the inequality definition of the floor function.

$$\begin{aligned} & 1 \leq n/2^k < 2 \\ \therefore \quad & 2^k \leq n < 2^{k+1} \\ \therefore \quad & k \leq \lg(n) < k + 1 \\ \therefore \quad & k = \lfloor \lg(n) \rfloor \end{aligned}$$

Thus for the recursion depth $k = \lfloor \lg(n) \rfloor$ we have $T(\lfloor n/2^k \rfloor) = T(1) = 0$, and hence the solution to the above recurrence is $T(n) = \lfloor \lg(n) \rfloor$. It follows that $T(n) = \Theta(\log(n))$.

Exercise 1
Check directly that $T(n) = \lfloor \lg(n) \rfloor$ is the solution to the above recurrence relation, i.e. check that $T(1) = 0$, and for any $n \geq 2$, that $T(n) = 1 + T(\lfloor n/2 \rfloor)$.

Exercise 2
Use this same technique to show that the recurrence

$$S(n) = \begin{cases} 0 & n = 1 \\ S(\lceil n/2 \rceil) + 1 & n \geq 2 \end{cases}$$

has solution $S(n) = \lceil \lg(n) \rceil$, and hence also $S(n) = \Theta(\log(n))$.

Comparing the solutions to the preceding examples, we see that replacing floor $\lfloor\ \rfloor$ by ceiling $\lceil\ \rceil$ has no affect on the *asymptotic solution* $\Theta(\log(n))$, while the *explicit solutions* are different: $\lfloor \lg(n) \rfloor$ vs. $\lceil \lg(n) \rceil$. We can change other details of a recurrence without changing the asymptotic solution. The following recurrence satisfies $T(n) = \Theta(\log(n))$ for any values of the constants *c, d*, and n_0.

Example 2

$$T(n) = \begin{cases} c & 1 \le n < n_0 \\ T(\lfloor n/2 \rfloor) + d & n \ge n_0 \end{cases}$$

Using the iteration method yields:

$$\begin{aligned} T(n) &= d + T(\lfloor n/2 \rfloor) \\ &= d + d + T(\lfloor n/2^2 \rfloor) \\ &\vdots \\ &= kd + T(\lfloor n/2^k \rfloor) \end{aligned}$$

We seek the first (i.e. the smallest) integer k such that $\lfloor n/2^k \rfloor < n_0$. This is equivalent to $n/2^k < n_0$, whence

$$\begin{aligned} \frac{n}{n_0} &< 2^k \\ k - 1 \le \lg(n/n_0) &< k \qquad \text{(since } k \text{ is the least integer satisfying the previous inequality)} \\ k - 1 &= \lfloor \lg(n/n_0) \rfloor \\ k &= \lfloor \lg(n/n_0) \rfloor + 1 \end{aligned}$$

Thus $T(n) = (\lfloor \lg(n/n_0) \rfloor + 1)d + c$ for $n \ge n_0$, and hence $T(n) = \Theta(\log(n))$ as claimed.

It is often difficult or impossible to determine an explicit solution via the iteration method, while it *is* possible to obtain an asymptotic solution. Consider

Example 3

$$T(n) = \begin{cases} 1 & n = 1 \\ T(\lfloor n/2 \rfloor) + n^2 & n \ge 2 \end{cases}$$

Upon iterating this recurrence we find $T(n) = \sum_{i=0}^{k-1} \left\lfloor \frac{n}{2^i} \right\rfloor^2 + 1$, where $k = \lfloor \lg n \rfloor$. We can use this expression to show that $T(n) = \Theta(n^2)$ as follows.

$$\begin{aligned} T(n) &= \sum_{i=0}^{k-1} \left\lfloor \frac{n}{2^i} \right\rfloor^2 + 1 \\ &\le n^2 \cdot \sum_{i=0}^{k-1} (1/4)^i + 1 \qquad \text{(since } \lfloor x \rfloor \le x) \\ &\le n^2 \cdot \sum_{i=0}^{\infty} (1/4)^i + 1 \qquad \text{(letting } k \to \infty) \\ &= n^2 \left(\frac{1}{1 - (1/4)} \right) + 1 \qquad \text{(from a well known series formula)} \end{aligned}$$

$$= \frac{4}{3}n^2 + 1 = O(n^2)$$

Therefore $T(n) = O(n^2)$. Showing that $T(n) = \Omega(n^2)$ is easier.

$$T(n) = \sum_{i=0}^{k-1} \left\lfloor \frac{n}{2^i} \right\rfloor^2 + 1$$

$$\geq n^2 + 1 \qquad \text{(dropping all but the } 0^{\text{th}} \text{ term)}$$

$$= \Omega(n^2)$$

It follows that $T(n) = \Omega(n^2)$ and therefore $T(n) = \Theta(n^2)$, as claimed. Note $T(n) = \Omega(n^2)$ can also be seen directly from the recurrence. Indeed, $T(n)$ has only non-negative values, so $T(n) = T(\lfloor n/2 \rfloor) + n^2 = \Omega(n^2)$.

The Master Method

The Master Theorem determines (asymptotic) solutions to recurrences of the form

$$T(n) = a\mathrm{T}(n/b) + f(n)$$

It is required that $a \geq 1$, $b > 1$, and that the function $f(n)$ be asymptotically positive. It is understood that $T(n/b)$ denotes either $T(\lfloor n/b \rfloor)$ or $T(\lceil n/b \rceil)$, and that $T(n) = \Theta(1)$ for some finite set of initial terms. It is part of the content of the theorem that these details, while they may change the recurrence solution, do not change the asymptotic solution. Such recurrence relations describe the run time of 'divide and conquer' algorithms that divide a problem of size *n* into *a* subproblems, each of size n/b. In this context $f(n)$ represents the cost of doing the dividing and re-combining.

Master Theorem

Let $a \geq 1$, $b > 1$, $f(n)$ be asymptotically positive, and let $T(n)$ be defined by $T(n) = a\mathrm{T}(n/b) + f(n)$. Then we have three cases:

1. If $f(n) = O\left(n^{\log_b(a)-\varepsilon}\right)$ for some $\varepsilon > 0$, then $T(n) = \Theta\left(n^{\log_b(a)}\right)$.
2. If $f(n) = \Theta\left(n^{\log_b(a)}\right)$, then $T(n) = \Theta\left(n^{\log_b(a)} \cdot \log(n)\right)$.
3. If $f(n) = \Omega\left(n^{\log_b(a)+\varepsilon}\right)$ for some $\varepsilon > 0$, and if $af(n/b) \leq cf(n)$ for some *c* in the range $0 < c < 1$ and for all sufficiently large *n*, then $T(n) = \Theta\left(f(n)\right)$.

Remarks In each case we compare $f(n)$ to the polynomial $n^{\log_b(a)}$, then determine which of these two functions is of a higher asymptotic order. In case (1) $n^{\log_b(a)}$ is of higher order than $f(n)$ (by a polynomial factor n^{ε}) and the solution $T(n)$ is in the class $\Theta\left(n^{\log_b(a)}\right)$. In case (3) $f(n)$ is the higher order (again by a polynomial factor n^{ε}), and an additional *regularity condition* is satisfied. The Master Theorem tells us in this case that $T(n) = \Theta\left(f(n)\right)$. In case (2) the two functions are asymptotically equivalent, and $T(n)$ is in the class $\Theta\left(n^{\log_b(a)} \cdot \log(n)\right) = \Theta\left(f(n) \cdot \log(n)\right)$.

We sometimes say, as in case (1), that $n^{\log_b(a)}$ is *polynomially larger* than $f(n)$, meaning that $f(n)$ is bounded above by a function that grows slower than $n^{\log_b(a)}$ by a polynomial factor, namely n^{ε} for some $\varepsilon > 0$.

Example 4 $T(n) = 8T(n/2) + n^3$
Observe that $a = 8$, $b = 2$, and $\log_b(a) = 3$. Hence $f(n) = n^3 = \Theta\left(n^{\log_b(a)}\right)$, so we are in case (2). Therefore $T(n) = \Theta\left(n^3 \log(n)\right)$.

Example 5 $T(n) = 5T(n/4) + n$.
In this case $a = 5$, $b = 4$, and $\log_b(a) = 1.1609... > 1$. Letting $\varepsilon = \log_4(5) - 1$ we have $\varepsilon > 0$, and therefore $f(n) = n = O\left(n^{\log_4(5)-\varepsilon}\right)$. Thus case(1) applies, and $T(n) = \Theta\left(n^{\log_4(5)}\right)$.

Example 6 $T(n) = 5T(n/4) + n^2$
Again $a = 5$, $b = 4$, so that $\log_b(a) = 1.1609... < 2$. Upon setting $\varepsilon = 2 - \log_4(5)$, we have that $\varepsilon > 0$, and $f(n) = n^2 = \Omega\left(n^{\log_4(5)+\varepsilon}\right)$. If the Master Theorem applies at all then, it must be that case (3) applies. However, the regularity condition must also be checked: $5f(n/4) \leq \text{cf}(n)$ for some $0 < c < 1$ and all sufficiently large *n*. This inequality says $5(n/4)^2 \leq \text{cn}^2$, i.e. $(5/16)n^2 \leq \text{cn}^2$, which is true as long as *c* is chosen to satisfy $5/16 \leq c < 1$. By case (3) $T(n) = \Theta(n^2)$.

The conclusion reached by the Master Theorem does not change when we replace $f(n)$ by a function that is asymptotically equivalent to it. This is implied by the theorem since the hypothesis in each case refers only to the asymptotic growth rate of $f(n)$, not it's actual numerical values. Thus we can, if convenient, replace $f(n)$ by a any simpler asymptotically equivalent function. For instance the recurrence $T(n) = 8T(n/2) + 10n^3 + 15n^2 - n^{1.5} + n\log(n) + 1$ can be reduced to $T(n) = 8T(n/2) + n^3$ which was our first example above, and we arrive at the very same asymptotic solution. (Of course the exact solution to the recurrence would be quite different.) For this reason recurrences are sometimes specified in the form $T(n) = \text{aT}(n/b) + \Theta\left(f(n)\right)$, if all that is required is an asymptotic solution. Notice that there is no mention of initial terms in the Master Theorem. It is part of the content of the theorem that the initial values of the recurrence do not effect it's asymptotic solution.

Observe that in the three preceding examples, $f(n)$ was a polynomial. This is a particularly easy setting in which to apply the Master Theorem. To establish which case applies, we merely compare $\deg(f)$ to the number $\log_b(a)$. If they are the same, case (2) applies. When $\log_b(a)$ is larger, case (1) applies by defining $\epsilon = \log_b(a) - \deg(f)$. When $\deg(f)$ is larger, case (3) applies upon setting $\epsilon = \deg(f) - \log_b(a)$, and it turns out that the regularity condition automatically holds.

Exercise 3 Prove that if $f(n)$ is a polynomial, and if $\deg(f) > \log_b(a)$, then case (3) of the Master Theorem applies, and that the regularity condition necessarily holds.

Checking the hypotheses can be a little more complicated if $f(n)$ is not a polynomial.

Example 7 $T(n) = T(\lfloor n/2 \rfloor) + 2T(\lceil n/2 \rceil) + \log(n!)$
First write this as $T(n) = 3T(n/2) + n\log(n)$. Let $\varepsilon = \frac{1}{2}(\log_2(3) - 1)$, then $\varepsilon > 0$, and $1 + \varepsilon = \log_2(3) - \varepsilon$. Observe $n\log(n) = o(n^{1+\varepsilon})$, whence $n\log(n) = O(n^{1+\varepsilon}) = O(n^{\log_2(3)-\varepsilon})$. Case (1) gives $T(n) = \Theta(n^{\log_2(3)})$.

In spite of the name "Master Theorem" the three cases do not cover all possibilities. There is a gap between cases (1) and (2) when $n^{\log_b(a)}$ is of higher asymptotic order than $f(n)$, but not *polynomially* higher. The following example illustrates this situation.

Example 8 $T(n) = 2T(n/2) + n/\log(n)$.
Observe that $n/\log(n) = \omega(n^{1-\varepsilon})$ for any $\varepsilon > 0$, whence $n/\log(n) \neq O(n^{1-\varepsilon})$, and therefore we are not in case (1). But also $n/\log(n) = o(n)$, so that $n/\log(n) \neq \Theta(n)$, and neither are we in case (2). Thus the Master Theorem cannot be applied to this recurrence.

A similar gap exists between cases (2) and (3). It is also possible that the regularity condition in case (3) fails, even though $f(n) = \Omega(n^{\log_b(a)+\varepsilon})$ for some $\varepsilon > 0$. We leave it as an exercise to find such an example.

Example 9 Recall the recurrence representing Binary Search: $T(n) = T(n/2) + 1$. We compare $1 = n^0$ to $n^{\log_2 1} = n^0$. Case 2 gives $T(n) = \Theta(\log(n))$.

Example 10 Recall the MergeSort recurrence: $T(n) = 2T(n/2) + n$. In this case we compare n to $n^{\log_2 2} = n^1 = n$. This time case 2 yeilds $T(n) = \Theta(n \log n)$.

Exercise 4 Re-write the algorithms MergeSort and Merge so as to divide the array into 3 subarrays, all of approximately equal lengths, sort these subarrays recursively, then merge the resulting sorted subarrays into a sorted version of the original array. Call the resulting algorithm 3-way MergeSort. By an analysis similar to that for MergeSort, show that 3-way MergeSort is modeled by the recurrence $T(n) = 3T(n/3) + n$, and again by case 2 of the Master theorem, we have $T(n) = \Theta(n \log n)$. Interestingly then, as far as asymptotic run time is concerned, MergeSort is not improved by dividing the array into more than 2 subarrays.

Example 11 Recall problem 5d on page 39 of CLRS. An inversion in an array $A[1 \cdots n]$ is a pair of indices (i, j) for which $1 \leq i < j \leq n$ and $A[i] > A[j]$. In other words, an inversion is a pair of positions for which the corresponding array elements are out of order.

The following problem was posed:

> Write an algorithm that takes an integer array as input, and returns the number of inversions in that array. Your algorithm should run in time $\Theta(n \log(n))$.

Here is one possible solution to this problem. It is modeled on `MergeSort()` and uses a subroutine called `Compare()` which is analogous to `Merge()`.

```
Inversions(A, p, r)
1. if p < r
2.     q = ⌊(p+r)/2⌋
3.     a = Inversions(A, p, q)          // count inversions in left half
4.     b = Inversions(A, q + 1, r)      // count inversions in right half
5.     c = Compare(A, p, q, r)          // count inversions "between" the two halves
6.     return a + b + c                 // return the sum
7. else
8.     return 0
```

Compare(A, p, q, r)
1. count $= 0$
2. for $i = p$ to q
3. for $j = q + 1$ to r
4. if $A[i] > A[j]$
5. count $++$
6. return count

Let $T(n)$ denote the number of array comparisons (line 4 of `Compare()`) performed by this algorithm on arrays of length n. Its not difficult to see that $T(n)$ must satisfy the following recurrence relation.

$$T(n) = \begin{cases} 0 & \text{if } n = 1 \\ T(\lfloor n/2 \rfloor) + T(\lceil n/2 \rceil) + \lfloor n/2 \rfloor \cdot \lceil n/2 \rceil & \text{if } n \geq 2 \end{cases}$$

Simplifying this recurrence as appropriate for use of the Master Theorem, we get $T(n) = 2T(n/2) + n^2$. Case 3 now yields $T(n) = \Theta(n^2)$. This is not the run time the problem statement called for. The solution (posted on the webpage) presents a different modification of `MergeSort()` in which the array is sorted as a side effect of the inversion count. That algorithm runs in time $T(n) = \Theta(n \log(n))$ by an analysis identical to that of Example 10 above. Surprisingly then, we can count inversions more effeciently if we sort as we go. The Master Theorem makes it easy to see such efficiencies, almost at a glance.

Proof of the Master Theorem

We sketch here the proof of part (1) of the Master Theorem. Basically the proof is the iteration method applied with full generality. We simplify matters by ignoring all floors and ceilings in the argument. A more rigorous treatment would of course include these details. Upon iteration we obtain

$$\begin{aligned} T(n) &= f(n) + \text{aT}(n/b) \\ &= f(n) + \text{af}(n/b) + a^2 T(n/b^2) \\ &= f(n) + \text{af}(n/b) + a^2 f(n/b^2) + a^3 T(n/b^3) \\ &\vdots \\ &= \textstyle\sum_{i=0}^{k-1} a^i f(n/b^i) + a^k T(n/b^k) \end{aligned}$$

The recurrence terminates when $n/b^k = 1$, so the recursion depth is $k = \log_b(n)$. For this value of k we have that

$$T(n) = \sum_{i=0}^{k-1} a^i f(n/b^i) + a^{\log_b(n)} T(1) \geq \text{const} \cdot n^{\log_b(a)} = \Omega(n^{\log_b(a)})$$

It remains to show only that $T(n) \leq O(n^{\log_b(a)})$, for then $T(n) = \Theta(n^{\log_b(a)})$ as required. Since we are in case (1) we have $f(n) = O(n^{\log_b(a)-\varepsilon})$, whence there exist positive c and ε such that $f(n) \leq cn^{\log_b(a)-\varepsilon}$ for all sufficiently large n. Therefore for such n, we have

$$\sum_{i=0}^{k-1} a^i f\left(n/b^i\right) \leq \sum_{i=0}^{k-1} a^i \cdot c \left(\frac{n}{b^i}\right)^{\log_b(a)-\varepsilon}$$

$$= cn^{\log_b(a)-\varepsilon} \cdot \sum_{i=0}^{k-1} a^i \left(\frac{1}{b^i}\right)^{\log_b(a)-\varepsilon}$$

$$= cn^{\log_b(a)-\varepsilon} \cdot \sum_{i=0}^{k-1} \frac{a^i}{(b^{\log_b a})^i} (b^\varepsilon)^i$$

$$= \frac{cn^{\log_b(a)}}{n^\varepsilon} \cdot \sum_{i=0}^{k-1} (b^\varepsilon)^i$$

$$= \frac{cn^{\log_b(a)}}{n^\varepsilon} \cdot \frac{(b^\varepsilon)^k - 1}{b^\varepsilon - 1}$$

$$\leq \frac{c}{b^\varepsilon - 1} \cdot \frac{n^{\log_b(a)}}{n^\varepsilon} (b^{\log_b n})^\varepsilon$$

$$= \frac{c}{b^\varepsilon - 1} \cdot n^{\log_b(a)},$$

whence $T(n) \leq \left(\frac{c}{b^\varepsilon - 1} + T(1)\right) n^{\log_b(a)} = \text{const} \cdot n^{\log_b(a)} = O\left(n^{\log_b(a)}\right)$. ■

We emphasize that the above argument was only a "sketch" and not a complete proof, owing to the fact that we ignored floors and ceilings. The reader should fill in those details as an exercise. See the proofs of cases (2) and (3) in section 4.6 of the text.