Algorithms: COMP3121/3821/9101/9801

Aleks Ignjatović

School of Computer Science and Engineering
University of New South Wales

LECTURE 2: RECURRENCES

Asymptotic notation

"Big Oh" notation: $f(n) = O(g(n))$ is an abbreviation for:

"There exists positive constants c and n_0 such that $0 \leq f(n) \leq c\, g(n)$ for all $n \geq n_0$".

In this case we say that $g(n)$ *is an asymptotic upper bound for* $f(n)$.

$f(n) = O(g(n))$ means that $f(n)$ does not grow substantially faster than $g(n)$ because a multiple of $g(n)$ eventually dominates $f(n)$.

Clearly, multiplying constants c of interest will be larger than 1, thus "enlarging" $g(n)$.

Asymptotic notation

"Omega" notation: $f(n) = \Omega(g(n))$ is an abbreviation for:

"There exists positive constants c and n_0 such that $0 \leq c\, g(n) \leq f(n)$ for all $n \geq n_0$."

In this case we say that $g(n)$ *is an asymptotic lower bound for* $f(n)$.

$f(n) = \Omega(g(n))$ essentially says that $f(n)$ grows at least as fast as $g(n)$, because $f(n)$ eventually dominates a multiple of g(n).

Clearly, multiplying constants c of interest will be smaller than 1, thus "shrinking" g(n) by a constant factor.

"Theta" notation: $f(n) = \Theta(g(n))$ iff and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$; thus, $f(n)$ and $g(n)$ have the same asymptotic growth rate.

Recurrences

Recurrences often arise in estimations of time complexity of divide-and-conquer algorithms

Merge-Sort(A,p,r) *sorting A[p..r]*

1. **if** $p < r$
2. **then** $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
3. Merge-Sort(A, p, q)
4. Merge-Sort$(A, q+1, r)$
5. Merge(A, p, q, r)

Since Merge(A, p, q, r) runs in linear time, the runtime $T(n)$ of Merge-Sort(A, p, r) satisfies

$$T(n) = 2T\left(\frac{n}{2}\right) + c\,n$$

Recurrences

If a divide-and-conquer algorithm:

- reduces a problem of size n to a many problems of smaller size n/b;
- if $f(n)$ is the overhead cost of splitting up/combining the solutions,

then the time complexity of such algorithm is given by

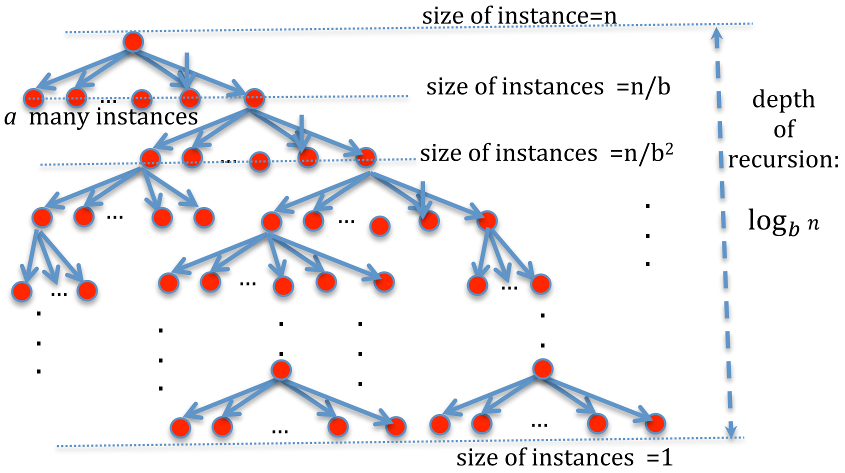
$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n)$$

Note: we should be writing

$$T(n) = a\,T\left(\left\lceil\frac{n}{b}\right\rceil\right) + f(n)$$

but it can be shown that assuming that n is a power of b is OK, and that the estimate produced is still valid of all n.

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n)$$



- Some recurrences can be solved explicitly, but this tends to be somewhat tricky.
- To estimate efficiency of an algorithm we **do not** need the exact solution of a recurrence
- We only need to find the **growth rate** of the solution i.e., its asymptotic behaviour plus the sizes of the constants involved (more about that later)
- This is what the **Master Theorem** provides (when it is applicable).

Master Theorem:

Let:

- $a \geq 1$ and $b > 1$ be integers;
- $f(n) > 0$ be a monotonically increasing function;
- $T(n)$ be the solution of the recurrence $T(n) = a\,T(n/b) + f(n)$;

Then:

1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$;
2. If $f(n) = \Theta(n^{\log_b a})$, then $T(n) = \Theta(n^{\log_b a} \log_2 n)$;
3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, **and** for some $c < 1$,

$$a\,f\,(n/b) \leq c\,f(n)$$

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

4. If none of these conditions hold, the Master Theorem is NOT applicable (in the form presented).

Master Theorem - Examples

- Let $T(n) = 4\,T(n/2) + n;$

  then $n^{\log_b a} = n^{\log_2 4} = n^2;$

  thus $f(n) = n = O(n^{2-\varepsilon})$ for any $\varepsilon < 1$.

  Condition of case 1 is satisfied; thus, $T(n) = \Theta(n^2)$.

- Let $T(n) = T(2n/3) + 2;$

  then $n^{\log_b a} = n^{\log_{3/2} 1} = n^0 = 1;$

  thus $f(n) = 2 = \Theta(1) = \Theta(n^{\log_{3/2} 1}) = \Theta(n^{\log_b a})$.

  Condition of case 2 is satisfied; thus,

$T(n) = \Theta(n^{\log_b a} \log_2 n) = \Theta(n^{\log_{3/2} 1} \log_2 n) = \Theta(n^0 \log_2 n) = \Theta(\log_2 n)$.

Master Theorem - Examples

- Let $T(n) = 3\,T(n/4) + n$;
  then $n^{\log_b a} = n^{\log_4 3} \approx n^{0.79}$;
  thus $f(n) = n = \Omega(n^{0.79+\varepsilon})$ for any $\varepsilon < 0.21$.
  Also, $af(n/b) = 3f(n/4) = 3/4\; n < c\,n$ for $c = .8 < 1$.

  Case 3 applies, and $T(n) = \Theta(f(n)) = \Theta(n)$.

- Let $T(n) = 2\,T(n/2) + n\log_2 n$;
  then $n^{\log_b a} = n^{\log_2 2} = n^1 = n$;
  thus $f(n) = n\log_2 n = \Omega(n)$, but
  $f(n) = n\log_2 n \neq \Omega(n^{1+\varepsilon})$, no matter how small $\varepsilon > 0$,
  because for every $\varepsilon > 0$, $\log_2 n < n^{\varepsilon}$ for all sufficiently large n.
  Homework: Prove this. *Hint:* Use *de L'Hôpital's Rule.*

  Thus, in this case the Master Theorem does **not** apply!

Master Theorem - Proof:

Since

$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n) \tag{1}$$

implies (by applying it to n/b in place of n)

$$T\left(\frac{n}{b}\right) = a\,T\left(\frac{n}{b^2}\right) + f\left(\frac{n}{b}\right) \tag{2}$$

and (by applying (1) to n/b^2 in place of n)

$$T\left(\frac{n}{b^2}\right) = a\,T\left(\frac{n}{b^3}\right) + f\left(\frac{n}{b^2}\right) \tag{3}$$

and so on ..., we get

$$\begin{aligned}
\overbrace{T(n) = a\,\underbrace{T\left(\frac{n}{b}\right)}_{(2)} + f(n)}^{(1)} &= a\Bigg(\underbrace{a\,T\left(\frac{n}{b^2}\right) + f\left(\frac{n}{b}\right)}_{(2)}\Bigg) + f(n)\\
= a^2\,\underbrace{T\left(\frac{n}{b^2}\right)}_{(3)} + a\,f\left(\tfrac{n}{b}\right) + f(n) &= a^2\Bigg(\underbrace{a\,T\left(\frac{n}{b^3}\right) + f\left(\frac{n}{b^2}\right)}_{(3)}\Bigg) + a\,f\left(\tfrac{n}{b}\right) + f(n)\\
= a^3\,\underbrace{T\left(\frac{n}{b^3}\right)} + a^2 f\left(\tfrac{n}{b^2}\right) + a\,f\left(\tfrac{n}{b}\right) + f(n) &= \ldots
\end{aligned}$$

Continuing in this way $\log_b n - 1$ many times we get ...

Master Theorem Proof:

$$T(n) = a^3 \underbrace{T\left(\frac{n}{b^3}\right)} + a^2 f\left(\frac{n}{b^2}\right) + a\,f\left(\frac{n}{b}\right) + f(n) =$$

$$= \dots$$

$$= a^{\lfloor \log_b n \rfloor} T\left(\frac{n}{b^{\lfloor \log_b n \rfloor}}\right) + a^{\lfloor \log_b n \rfloor - 1} f\left(\frac{n}{b^{\lfloor \log_b n \rfloor - 1}}\right) + \dots$$
$$+ a^3 f\left(\frac{n}{b^3}\right) + a^2 f\left(\frac{n}{b^2}\right) + a\,f\left(\frac{n}{b}\right) + f(n)$$

$$\approx a^{\log_b n} T\left(\frac{n}{b^{\log_b n}}\right) + \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right)$$

We now use $a^{\log_b n} = n^{\log_b a}$:

$$T(n) \approx n^{\log_b a} T(1) + \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right) \tag{4}$$

Note that so far we did not use any assumptions on $f(n)$ . . .

Master Theorem Proof:

Case 1: $f(m) = O(m^{\log_b a - \varepsilon})$

$$\sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right) = \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i O\left(\frac{n}{b^i}\right)^{\log_b a - \varepsilon}$$

$$= O\left(\sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a - \varepsilon}\right) = O\left(n^{\log_b a - \varepsilon} \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} \left(\frac{a^i}{(b^i)^{\log_b a - \varepsilon}}\right)\right)$$

$$= O\left(n^{\log_b a - \varepsilon} \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} \left(\frac{a}{b^{\log_b a - \varepsilon}}\right)^i\right) = O\left(n^{\log_b a - \varepsilon} \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} \left(\frac{a}{b^{\log_b a} b^{-\varepsilon}}\right)^i\right)$$

$$= O\left(n^{\log_b a - \varepsilon} \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} \left(\frac{a\, b^{\varepsilon}}{a}\right)^i\right) = O\left(n^{\log_b a - \varepsilon} \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} (b^{\varepsilon})^i\right)$$

$$= O\left(n^{\log_b a - \varepsilon} \frac{(b^{\varepsilon})^{\lfloor \log_b n \rfloor} - 1}{b^{\varepsilon} - 1}\right); \qquad \text{we are using } \sum_{i=0}^{m} q^m = \frac{q^{m+1} - 1}{q - 1}$$

Case 1 - continued:

$$\sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right) = O\left(n^{\log_b a - \varepsilon} \frac{(b^\varepsilon)^{\lfloor \log_b n \rfloor} - 1}{b^\varepsilon - 1}\right)$$

$$= O\left(n^{\log_b a - \varepsilon} \frac{\left(b^{\lfloor \log_b n \rfloor}\right)^\varepsilon - 1}{b^\varepsilon - 1}\right)$$

$$= O\left(n^{\log_b a - \varepsilon} \frac{n^\varepsilon - 1}{b^\varepsilon - 1}\right)$$

$$= O\left(\frac{n^{\log_b a} - n^{\log_b a - \varepsilon}}{b^\varepsilon - 1}\right)$$

$$= O\left(n^{\log_b a}\right)$$

Since we had: $T(n) \approx n^{\log_b a} T(1) + \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right)$ we get:

$$T(n) \approx n^{\log_b a} T(1) + O\left(n^{\log_b a}\right)$$

$$= \Theta\left(n^{\log_b a}\right)$$

Master Theorem Proof:

Case 2: $f(m) = \Theta(m^{\log_b a})$

$$\begin{aligned}
\sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right) &= \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i \Theta\left(\frac{n}{b^i}\right)^{\log_b a} \\
&= \Theta\left(\sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i \left(\frac{n}{b^i}\right)^{\log_b a}\right) \\
&= \Theta\left(n^{\log_b a} \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} \left(\frac{a^i}{(b^i)^{\log_b a}}\right)\right) \\
&= \Theta\left(n^{\log_b a} \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} \left(\frac{a}{b^{\log_b a}}\right)^i\right) \\
&= \Theta\left(n^{\log_b a} \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} 1\right) \\
&= \Theta\left(n^{\log_b a} \lfloor \log_b n \rfloor\right)
\end{aligned}$$

Case 2 (continued):

Thus,

$$\sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right) = \Theta\left(n^{\log_b a} \log_b n\right) = \Theta\left(n^{\log_b a} \log_2 n\right)$$

because $\log_b n = \log_2 n \cdot \log_b 2 = \Theta(\log_2 n)$. Since we had (1):

$$T(n) \approx n^{\log_b a} T(1) + \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right)$$

we get:

$$\begin{aligned} T(n) &\approx n^{\log_b a} T(1) + \Theta\left(n^{\log_b a} \log_2 n\right) \\ &= \Theta\left(n^{\log_b a} \log_2 n\right) \end{aligned}$$

Master Theorem Proof:

Case 3: $f(m) = \Omega(m^{\log_b a + \varepsilon})$ and $a\, f(n/b) \le c\, f(n)$ for some $0 < c < 1$.

We get by substitution:

$$f(n/b) \le \frac{c}{a}\, f(n)$$
$$f(n/b^2) \le \frac{c}{a}\, f(n/b)$$
$$f(n/b^3) \le \frac{c}{a}\, f(n/b^2)$$
$$\dots$$
$$f(n/b^i) \le \frac{c}{a}\, f(n/b^{i-1})$$

By chaining these inequalities we get

$$f(n/b^2) \le \frac{c}{a}\, \underbrace{f(n/b)} \le \frac{c}{a} \cdot \underbrace{\frac{c}{a}\, f(n)} = \frac{c^2}{a^2}\, f(n)$$

$$f(n/b^3) \le \frac{c}{a}\, \underbrace{f(n/b^2)} \le \frac{c}{a} \cdot \underbrace{\frac{c^2}{a^2}\, f(n)} = \frac{c^3}{a^3}\, f(n)$$

$$\dots$$

$$f(n/b^i) \le \frac{c}{a}\, \underbrace{f(n/b^{i-1})} \le \frac{c}{a} \cdot \underbrace{\frac{c^{i-1}}{a^{i-1}}\, f(n)} = \frac{c^i}{a^i}\, f(n)$$

Master Theorem Proof:

Case 3 (continued):

We got $$f(n/b^i) \leq \frac{c^i}{a^i} f(n)$$

Thus,

$$\sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right) \leq \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i \frac{c^i}{a^i} f(n) < f(n) \sum_{i=0}^{\infty} c^i = \frac{f(n)}{1-c}$$

Since we had (1):

$$T(n) \approx n^{\log_b a} T(1) + \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right)$$

and since $f(n) = \Omega(n^{\log_b a + \varepsilon})$ we get:

$$T(n) < n^{\log_b a} T(1) + O(f(n)) = O(f(n))$$

but we also have

$$T(n) = aT(n/b) + f(n) > f(n)$$

thus,

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

Master Theorem Proof: Homework

Exercise 1: Show that condition

$$f(n) = \Omega(n^{\log_b a+\varepsilon})$$

follows from the condition

$$a\, f(n/b) \leq c\, f(n) \;\; \text{for some } 0 < c < 1.$$

Exercise 2: Estimate $T(n)$ for

$$T(n) = 2\,T(n/2) + n \log n$$

Note: we have seen that the Master Theorem does NOT apply, but the technique used in its proof still works!