Algorithms
COMP3121/3821/9101/9801
3. RECURRENCES

School of Computer Science and Engineering
University of New South Wales Sydney

Asymptotic notations

"Big O" notation: $f(n) = O(g(n))$

- This is an abbreviation for: *"There exist positive constants c and n_0 such that $0 \leq f(n) \leq cg(n)$ for all $n \geq n_0$".*
- In this case we say $g(n)$ *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 $f(n)$.

"Big Omega" notation: $f(n) = \Omega(g(n))$

- This is an abbreviation for: *"There exists positive constants c and n_0 such that $0 \leq cg(n) \leq f(n)$ for all $n \geq n_0$."*
- In this case we say $g(n)$ *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 $g(n)$.

Asymptotic notations

"Big Theta" notation: $f(n) = \Theta(g(n))$

- $f(n) = \Theta(g(n))$ if and only if $f(n) = O(g(n))$ and $f(n) = \Omega(g(n))$;
- We say $f(n)$ *and* $g(n)$ *have the same asymptotic growth rate.*

Other notational remarks

- $f(n) = h(n) + O(g(n))$ means that $f(n) - h(n) = O(g(n))$.
- The notation is not symmetric. $O(n) = O(n^2)$ but $O(n^2) \neq O(n)$.

Basic properties

$f_1 = O(g_1) \wedge f_2 = O(g_2)$ implies $f_1 + f_2 = O(\max(g_1, g_2))$
$f_1 = O(g_1) \wedge f_2 = O(g_2)$ implies $f_1 f_2 = O(g_1 g_2)$
$f(n) = O(g(n))$ iff $f(n) = O(cg(n))$
$f(n) = O(\log(n^c))$ iff $f(n) = O(\log(n))$
$f(n) = O(n^2)$ implies $f(n) = O(n^3)$ but not the reverse
$f(n) = O(2^n)$ implies $f(n) = O(3^n)$ but not the reverse
$f = \Omega(g)$ iff $g = O(f)$.

Recurrences

- Recurrences are important to us because they arise in estimations of time complexity of divide-and-conquer algorithms.

Merge Sort

merge-sort(A, p, r)   /* sorting A[p..r] */
    if $p < r$ **then**
        $q \leftarrow \lfloor \frac{p+r}{2} \rfloor$
        merge-sort(A, p, q)
        merge-sort(A, $q+1$, r)
        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

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

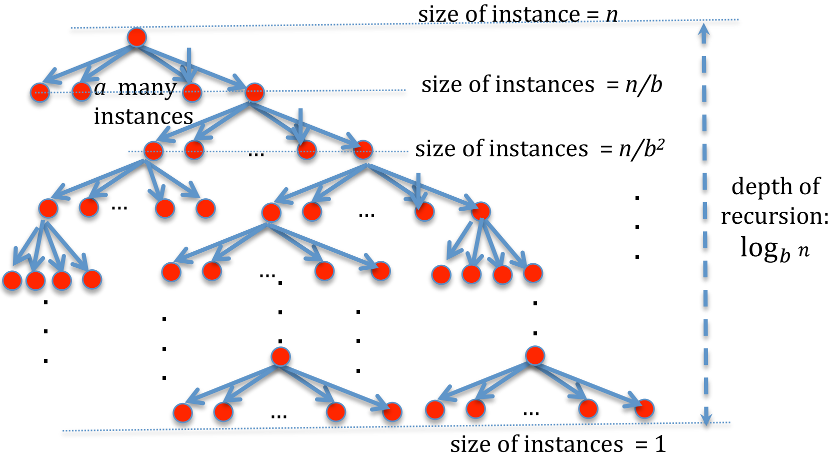
Recurrences

- Let $a \geq 1$ be an integer and $b > 1$ a real number;
- Assume that a divide-and-conquer algorithm:
  - reduces a problem of size n to a many problems of smaller size n/b;
  - the overhead cost of splitting up/combining the solutions for size n/b into a solution for size n is if $f(n)$,
- then the time complexity of such algorithm satisfies

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

- **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 for all n.

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



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

Master Theorem

Theorem

Let $a \geq 1$ and $b > 1$ be integers, $f(n) > 0$ be a non-decreasing function, $T(n)$ be the solution of the recurrence $T(n) = aT(n/b) + f(n)$, 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. $\left.\begin{array}{l}\textit{If } f(n) = \Omega(n^{\log_b a + \varepsilon}) \textit{ for some } \varepsilon > 0, \textbf{\textit{ and}} \\ \textit{for some } c < 1 \textit{ and } n_0, \textit{ for all } n \geq n_0,\ af(n/b) \leq cf(n)\end{array}\right\}$ *then* $T(n) = \Theta(f(n))$.

If conditions (1–3) do not hold, the Master Theorem is NOT applicable. (But often the proof of the Master Theorem can be tweaked to obtain the asymptotic of the solution $T(n)$ in such a case when the Master Theorem does not apply; an example is $T(n) = 2T(n/2) + n \log n$).

Remark

- For any $b > 1$, $\quad \log_b n = \log_b 2 \, \log_2 n$.
- Since $b > 1$ is constant (does not depend on n), we have for $c = \log_b 2 > 0$

$$\log_b n = c \; \log_2 n;$$
$$\log_2 n = \frac{1}{c} \log_b n;$$

- Thus, $\quad \log_b n = \Theta(\log_2 n)$ and also $\quad \log_2 n = \Theta(\log_b n)$.
- So whenever we have $f = \Theta(g(n) \log n)$ we do not have to specify what base the log is—all bases produce equivalent asymptotic estimates.

Examples

Theorem (Master Theorem)

$T(n) = a\,T(n/b) + f(n)$, *then:*

1. $f(n) = O(n^{\log_b a - \varepsilon}) \Rightarrow T(n) = \Theta(n^{\log_b a})$;
2. $f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \log_2 n)$;
3. $f(n) = \Omega(n^{\log_b a + \varepsilon}) \wedge af(n/b) \le cf(n) \Rightarrow T(n) = \Theta(f(n))$.

$T(n) = 4\,T(n/2) + n$ (naive large integer multiplication)

$a = 4$, $b = 2$, and $f(n) = n$.
Since $n^{\log_b a - \varepsilon} = n^{\log_2 4 - \varepsilon} = n^{2-\varepsilon}$ and $n = O(n^{2-\varepsilon})$ for $\varepsilon = 0.5 > 0$,
then condition of case 1 is satisfied. $T(n) = \Theta(n^2)$.

$T(n) = 2T(n/2) + c\,n$ (`merge-sort`)

$a = 2$, $b = 2$, and $f(n) = cn$.
Since $n^{\log_b a} = n^{\log_2 2} = n^1 = n$ and $cn = \Theta(n)$,
then condition of case 2 is satisfied. $T(n) = \Theta(n \log n)$.

Theorem (Master Theorem)

$T(n) = a\,T(n/b) + f(n)$, *then:*

1. $f(n) = O(n^{\log_b a - \varepsilon}) \quad \Rightarrow \quad T(n) = \Theta(n^{\log_b a})$;
2. $f(n) = \Theta(n^{\log_b a}) \quad \Rightarrow \quad T(n) = \Theta(n^{\log_b a} \log_2 n)$;
3. $f(n) = \Omega(n^{\log_b a + \varepsilon}) \wedge af\,(n/b) \le cf(n) \quad \Rightarrow \quad T(n) = \Theta(f(n))$.

$T(n) = 3\,T(n/4) + n$

$a = 3$, $b = 4$, and $f(n) = n$.
Since $n^{\log_b a + \varepsilon} = n^{\log_4 3 + \varepsilon} \le n^{0.8+\varepsilon}$ and $n = \Omega(n^{0.8+\varepsilon})$ for $\varepsilon = 0.1 > 0$,
and $af(n/b) = 3f(n/4) = 3/4 n < cn = cf(n)$ for $c = .8 < 1$
then condition of case 3 is satisfied. $T(n) = \Theta(n)$.

$T(n) = T(n/2) + c$ (`binary-search`)

$a = 1$, $b = 2$, and $f(n) = c$.
Since $n^{\log_b a} = n^{\log_2 1} = n^0 = 1$ and $c = \Theta(1)$,
then condition of case 2 is satisfied. $T(n) = \Theta(\log n)$.

Theorem (Master Theorem)

$T(n) = a\,T(n/b) + f(n)$, *then:*

1. $f(n) = O(n^{\log_b a - \varepsilon}) \Rightarrow T(n) = \Theta(n^{\log_b a})$;
2. $f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \log_2 n)$;
3. $f(n) = \Omega(n^{\log_b a + \varepsilon}) \wedge af\,(n/b) \leq cf(n) \Rightarrow T(n) = \Theta(f(n))$.

$T(n) = 2\,T(n/2) + n \log_2 n$

$a = 2$, $b = 2$, and $f(n) = n \log_2 n$.
$n^{\log_b a} = n^{\log_2 2} = n$ so the case 1 and 2 do not apply.
Furthermore, $f(n) = n \log_2 n \neq \Omega(n^{1+\varepsilon})$, no matter how small $\varepsilon > 0$.
So case 3 does not apply either.
In this example the Master Theorem does **not** apply!

Homework:

Formally prove that $f(n) = n \log_2 n \neq O(n^1)$;
and that $f(n) = n \log_2 n \neq \Omega(n^{1+\varepsilon})$, no matter how small $\varepsilon > 0$.
Hint: Use *de L'Hôpital's Rule* to show that $\log n / n^{\varepsilon} \to 0$.

Theorem (Master Theorem)

$T(n) = a\,T(n/b) + f(n)$, *then:*

1. $f(n) = O(n^{\log_b a - \varepsilon}) \Rightarrow T(n) = \Theta(n^{\log_b a})$;
2. $f(n) = \Theta(n^{\log_b a}) \Rightarrow T(n) = \Theta(n^{\log_b a} \log_2 n)$;
3. $f(n) = \Omega(n^{\log_b a + \varepsilon}) \wedge af\,(n/b) \leq cf(n) \Rightarrow T(n) = \Theta(f(n))$.

$T(n) = 3\,T(n/2) + n$ (Karatsuba integer multiplication)

$a = 3$, $b = 2$, and $f(n) = n$.
Since $n^{\log_b a - \varepsilon} = n^{\log_2 3 - \varepsilon} = \Omega(n)$ for $\varepsilon = 0.5 > 0$ (indeed $\log_2 3 \simeq 1.58$), then condition of case 1 is satisfied. $T(n) = \Theta(n^{\log_2 3})$.

$T(n) = 7\,T(n/2) + n^2$ (Strassen matrix multiplication)

$a = 7$, $b = 2$, and $f(n) = n^2$.
Since $n^{\log_b a - \varepsilon} = n^{\log_2 7 - \varepsilon} = \Omega(n^2)$ for $\varepsilon = 0.5 > 0$ (indeed $\log_2 7 \simeq 2.8$), then condition of case 1 is satisfied. $T(n) = \Theta(n^{\log_2 7})$.

Master Theorem—Proof.

Assume that $$\forall m, T(m) = a\,T\left(\frac{m}{b}\right) + f(m) \tag{1}$$

$$T\left(\frac{n}{b}\right) = a\,T\left(\frac{n}{b^2}\right) + f\left(\frac{n}{b}\right) \qquad \text{(apply (1) to } m = n/b) \tag{2}$$

$$T\left(\frac{n}{b^2}\right) = a\,T\left(\frac{n}{b^3}\right) + f\left(\frac{n}{b^2}\right) \qquad \text{(apply (1) to } m = {}^{n}/_{b^2}) \tag{3}$$

$$\dots$$

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

we get

$$\begin{aligned}
T(n) &= a\,\overbrace{T\left(\frac{n}{b}\right)}^{(2)} + f(n) = a\left(\overbrace{aT\left(\frac{n}{b^2}\right) + f\left(\frac{n}{b}\right)}^{(2)}\right) + f(n) \\
&= a^2\,\overbrace{T\left(\frac{n}{b^2}\right)}^{(3)} + af\left(\frac{n}{b}\right) + f(n) = a^2\left(\overbrace{aT\left(\frac{n}{b^3}\right) + f\left(\frac{n}{b^2}\right)}^{(3)}\right) + af\left(\frac{n}{b}\right) + f(n) \\
&= a^3 T\left(\frac{n}{b^3}\right) + a^2 f\left(\frac{n}{b^2}\right) + af\left(\frac{n}{b}\right) + f(n) = \dots \\
&\dots \\
&= a^{\lfloor \log_b n \rfloor} T\left(\frac{n}{b^{\lfloor \log_b n \rfloor}}\right) + \sum_{i=0}^{\lfloor \log_b n \rfloor - 1} a^i f\left(\frac{n}{b^i}\right)
\end{aligned}$$

Master Theorem—Proof.

Recall that $a^{\log_b n} = n^{\log_b a}$.

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

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

Master Theorem—Proof. Case 1: $f(m) = O(m^{\log_b a - \varepsilon})$

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

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

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

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

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

Master Theorem—Proof. Case 1: $f(m) = O(m^{\log_b a - \varepsilon})$

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

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

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

Master Theorem—Proof. Case 2: $f(m) = \Theta(m^{\log_b a})$

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

Master Theorem—Proof. Case 2: $f(m) = \Theta(m^{\log_b a})$

Recall that $\log_b n = \log_2 n \cdot \log_b 2 = \Theta(\log_2 n)$.

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

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

we get:

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

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

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

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

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

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

then we get $T(n) \leq n^{\log_b a} T(1) + O(f(n)) = O(f(n))$

but recall we also had $T(n) = aT(n/b) + f(n) > f(n)$

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

Master Theorem—Homework.

Exercise: 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! Just unwind the recurrence and sum up the logarithmic overheads.

That's All, Folks!!