Algorithms:
COMP3121/3821/9101/9801

Aleks Ignjatović

School of Computer Science and Engineering
University of New South Wales

TOPIC 1: RECURRENCES
“Big Oh” notation: \( f(n) = O(g(n)) \) is an abbreviation for:

“There exist positive constants \( c \) and \( n_0 \) such that 
\[ 0 \leq f(n) \leq c \, g(n) \, \text{for all} \, n \geq n_0 \].

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
  \( 0 \leq c \cdot g(n) \leq f(n) \) 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 are important to us because they 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 \(\text{Merge}(A, p, q, r)\) runs in linear time, the runtime \(T(n)\) of \(\text{Merge-Sort}(A, p, r)\) satisfies

\[
T(n) = 2T \left( \frac{n}{2} \right) + cn
\]
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 $f(n)$,

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

- size of instance = \( n \)
- size of instances = \( n/b \)
- size of instances = \( n/b^2 \)
- ... 
- size of instances = 1

depth of recursion: \( \log_b n \)
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:

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 \( T(n) = aT(n/b) + f(n) \);

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. 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 - a remark

- Note that for any $b > 1$,

$$\log_b n = \log_b 2 \log_2 n;$$

- 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,

$$\log_b n = \Theta(\log_2 n)$$

and also

$$\log_2 n = \Theta(\log_b n).$$

- 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.
Master Theorem - Examples

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

  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 $T(n) = 2T(n/2) + cn$;

  then $n^{\log_b a} = n^{\log_2 2} = n^1 = n$;

  thus $f(n) = cn = \Theta(n) = \Theta(n^{\log_2 2})$.

  Thus, condition of case 2 is satisfied; and so,

  $T(n) = \Theta(n^{\log_2 2 \log n}) = \Theta(n \log n)$. 
Let $T(n) = 3T(n/4) + n$;

- then $n^{\log_b a} = n^{\log_4 3} < n^{0.8}$;
- thus $f(n) = n = \Omega(n^{0.8+\varepsilon})$ for any $\varepsilon < 0.2$.
- Also, $af(n/b) = 3f(n/4) = 3/4 n < cn$ for $c = .8 < 1$.

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

Let $T(n) = 2T(n/2) + n \log_2 n$;

- then $n^{\log_b a} = n^{\log_2 2} = n^1 = n$.
- Thus, $f(n) = n \log_2 n = \Omega(n)$.
- However, $f(n) = n \log_2 n \neq \Omega(n^{1+\varepsilon})$, no matter how small $\varepsilon > 0$.
- This is because for every $\varepsilon > 0$, and every $c > 0$, no matter how small, $\log_2 n < c \cdot n^\varepsilon$ for all sufficiently large $n$.
- **Homework:** Prove this.
  *Hint:* Use de L’Hôpital’s Rule to show that $\log n/n^\varepsilon \to 0$.

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

Since

\[ T(n) = a \ T\left(\frac{n}{b}\right) + f(n) \]  \hspace{1cm} (1)

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

\[ T\left(\frac{n}{b}\right) = a \ T\left(\frac{n}{b^2}\right) + f\left(\frac{n}{b}\right) \]  \hspace{1cm} (2)

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

\[ T\left(\frac{n}{b^2}\right) = a \ T\left(\frac{n}{b^3}\right) + f\left(\frac{n}{b^2}\right) \]  \hspace{1cm} (3)

and so on ..., we get

\[
T(n) = a \ T\left(\frac{n}{b}\right) + f(n) = a \left(a \ T\left(\frac{n}{b^2}\right) + f\left(\frac{n}{b}\right)\right) + f(n)
\]

(1)

\[
= a^2 \ T\left(\frac{n}{b^2}\right) + a f\left(\frac{n}{b}\right) + f(n) = a^2 \left(a \ T\left(\frac{n}{b^3}\right) + f\left(\frac{n}{b^2}\right)\right) + a f\left(\frac{n}{b}\right) + f(n)
\]

(2)

\[
= a^3 \ T\left(\frac{n}{b^3}\right) + a^2 f\left(\frac{n}{b^2}\right) + a f\left(\frac{n}{b}\right) + f(n) = \ldots
\]

(3)
Master Theorem Proof:

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

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

$$= \ldots$$

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

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

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

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

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

Note that so far we did not use any assumptions on $f(n)$, \ldots
Master Theorem Proof:

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

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

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

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

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

\[
= O\left(n^{\log_b a - \varepsilon} \frac{(b^\varepsilon)^{\lfloor \log_b n \rfloor - 1}}{b^\varepsilon - 1}\right); \quad \text{we are using } \sum_{i=0}^{m} q^m = \frac{q^{m+1} - 1}{q - 1}
\]
Master Theorem Proof:

Case 1 - continued:

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

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

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

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

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

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

\[
\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) \\
= \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} \right) \\
= \Theta \left( n^{\log_b a} \lfloor \log_b n \rfloor \right)
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:

\[ T(n) \approx n^{\log_b a} T(1) + \Theta \left( n^{\log_b a \log_2 n} \right) = \Theta \left( n^{\log_b a \log_2 n} \right) \]
Master Theorem Proof:

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

We get by substitution:

\[
\begin{align*}
  f(n/b) &\leq \frac{c}{a} f(n) \\
  f(n/b^2) &\leq \frac{c}{a} f(n/b) \\
  f(n/b^3) &\leq \frac{c}{a} f(n/b^2) \\
  &\vdots \\
  f(n/b^i) &\leq \frac{c}{a} f(n/b^{i-1})
\end{align*}
\]

By chaining these inequalities we get

\[
\begin{align*}
  f(n/b^2) &\leq \frac{c}{a} f(n/b) \leq \frac{c}{a} \cdot \frac{c}{a} f(n) = \frac{c^2}{a^2} f(n) \\
  f(n/b^3) &\leq \frac{c}{a} f(n/b^2) \leq \frac{c}{a} \cdot \frac{c^2}{a^2} f(n) = \frac{c^3}{a^3} f(n) \\
  &\vdots \\
  f(n/b^i) &\leq \frac{c}{a} f(n/b^{i-1}) \leq \frac{c}{a} \cdot \frac{c^{i-1}}{a^{i-1}} f(n) = \frac{c^i}{a^i} f(n)
\end{align*}
\]
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,

\[
T(n) = \Theta(f(n))
\]
Exercise 1: Show that condition

\[ f(n) = \Omega(n^{\log_b a + \epsilon}) \]

follows from the condition

\[ a f(n/b) \leq c f(n) \quad \text{for some } 0 < c < 1. \]

Exercise 2: Estimate \( T(n) \) for

\[ T(n) = 2T(n/2) + n \log n \]

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.