Algorithms: COMP3121/9101

School of Computer Science and Engineering
University of New South Wales

3. RECURRENCES - part A

Asymptotic notation

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

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

- **"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 $g(n)$.

- Since $c\,g(n) \leq f(n)$ if and only if $g(n) \leq \frac{1}{c} f(n)$, we have $f(n) = \Omega(g(n))$ if and only if $g(n) = O(f(n))$.
- **"Theta" notation:** $f(n) = \Theta(g(n))$ if 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]*

```
if p < r
    then q ← ⌊(p+r)/2⌋
            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

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

- 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

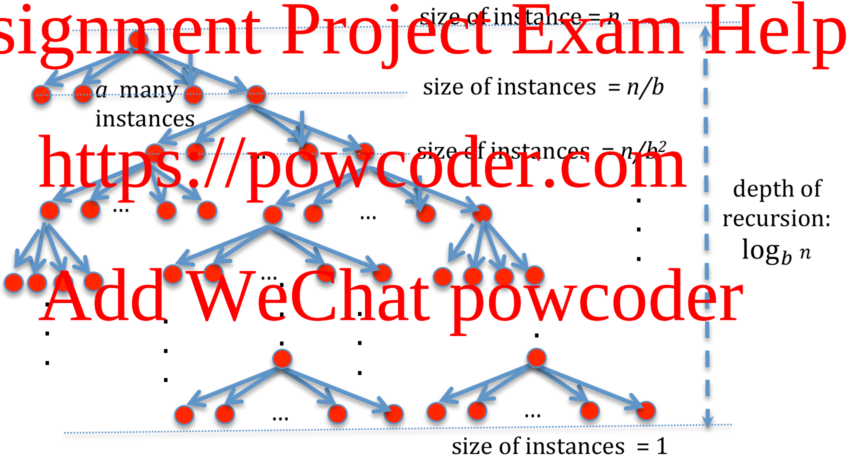
$$T(n) = a\,T\left(\frac{n}{b}\right) + f(n)$$

- **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 ignoring the integer parts and additive constants is OK when it comes to obtaining the asymptotics.

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



- 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 (approximate) **sizes of the constants** involved (more about that later)

- This is what the **Master Theorem** provides (when it is applicable).

Master Theorem:

Let:

- $a \geq 1$ be an integer and and $b > 1$ a real;
- $f(n) > 0$ be a non-decreasing 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 $T(n) = \Theta(n^{\log_b a} \log n)$;
3. If $f(n) = \Omega(n^{\log_b a + \varepsilon})$ for some $\varepsilon > 0$, **and** for some $c < 1$ and some n_0,

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

   holds for all $n > n_0$, then $T(n) = \Theta(f(n))$;
4. If none of these conditions 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 $T(n) = 2T(n/2) + n \log n$).

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 (but we do have to specify b in expressions such as $n^{\log_b a}$).

- 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 $T(n) = 2T(n/2) + 5\,n$;

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

  thus $f(n) = 5\,n = \Theta(n) = \Theta(n^{\log_2 2})$.

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

$$T(n) = \Theta(n^{\log_2 2} \log n) = \Theta(n \log n).$$

- Let $T(n) = 3\,T(n/4) + n$;
  - 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, $af(n/b) = 3f(n/4) = 3/4\,n < cn = cf(n)$ for $c = .9 < 1$.
  - Thus, 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, $f(n) = n\log_2 n = \Omega(n)$.
  - 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

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

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

$$\overbrace{T(n) = a\,\underbrace{T\left(\frac{n}{b}\right)}_{(2L)} + f(n)}^{(1)} = a\Bigg(\underbrace{a\,T\left(\frac{n}{b^2}\right) + f\left(\frac{n}{b}\right)}_{(2R)}\Bigg) + f(n)$$

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

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

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

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

$$= \ldots$$

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

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

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

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 $f(n)$ ...

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Master Theorem Proof:

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

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

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

$$\dots$$

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

Case 3 (continued):

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

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

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

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

follows from the condition

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

Example: Let us estimate the asymptotic growth rate of $T(n)$ which satisfies

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

Note: we have seen that the Master Theorem does **NOT** apply, but the technique used in its proof still works! So let us just unwind the recurrence and sum up the logarithmic overheads.

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

$$= 2\left(\overbrace{2T\left(\frac{n}{2^2}\right) + \frac{n}{2}\log\frac{n}{2}}\right) + n\log n$$

$$= 2^2\,\underbrace{T\left(\frac{n}{2^2}\right)} + n\log\tfrac{n}{2} + n\log n$$

$$= 2^2\left(\overbrace{2T\left(\frac{n}{2^3}\right) + \frac{n}{2^2}\log\frac{n}{2^2}}\right) + n\log\tfrac{n}{2} + n\log n$$

$$= 2^3\,\underbrace{T\left(\frac{n}{2^3}\right)} + n\log\tfrac{n}{2^2} + n\log\tfrac{n}{2} + n\log n$$

$$\ldots$$

$$= 2^{\log n}T\left(\tfrac{n}{2^{\log n}}\right) + n\log\tfrac{n}{2^{\log n-1}} + \ldots + n\log\tfrac{n}{2^2} + n\log\tfrac{n}{2} + n\log n$$

$$= nT(1) + n(\log n \times \log n - \log 2^{\log n-1} - \ldots - \log 2^2 - \log 2)$$

$$= nT(1) + n((\log n)^2 - (\log n - 1) - \ldots - 3 - 2 - 1)$$

$$= nT(1) + n((\log n)^2 - \log n(\log n - 1)/2$$

$$= nT(1) + n((\log n)^2/2 + \log n/2)$$

$$= \Theta\left(n(\log n)^2\right).$$

PUZZLE!

Five pirates have to split 100 bars of gold. They all line up and proceed as follows:

1. The first pirate in line gets to propose a way to split up the gold (for example: everyone gets 20 bars)
2. The pirates, including the one who proposed, vote on whether to accept the proposal. If the proposal is rejected, the pirate who made the proposal is killed.
3. The next pirate in line then makes his proposal, and the 4 pirates vote again. If the vote is tied (2 vs 2) then the proposing pirate is still killed. Only majority can accept a proposal. The process continues until a proposal is accepted or there is only one pirate left. Assume that every pirate :
    - above all wants to live;
    - given that he will be alive he wants to get as much gold as possible;
    - given maximal possible amount of gold, he wants to see any other pirate killed, just for fun;
    - each pirate knows his exact position in line;
    - all of the pirates are excellent puzzle solvers.

Question : What proposal should the first pirate make?

Hint: assume first that there are only two pirates, and see what happens. Then assume that there are three pirates and that they have figured out what happens if there were only two pirates and try to see what they would do. Further, assume that there are four pirates and that they have figured out what happens if there were only three pirates, try to see what they would do. Finally assume there are five pirates and that they have figured out what happens if there were only four pirates.