CS3230 Semester 2 2024/2025

Design and Analysis of Algorithms

Tutorial 02
Recurrences and Master Theorem
For Week 03

Document is last modified on: January 21, 2025

MODEL ANSWER IS FOR OUR CLASS ONLY; NOT TO BE DISTRIBUTED IN PUBLIC

1 Lecture Review: Recurrences

Given a recurrence in the form of $T(n) = a \cdot T(\frac{n}{b}) + f(n)$, where $f(n) = c \cdot n^m \log^k n$, we want to give a **tight** asymptotic bound for $T(n)$.

There are a few ways to solve recurrences, with the easiest being the Master theorem (master method). Let $d = \log_b a$ (this d plays an important role; also notice $b^d = a$).

1. Case 1: $f(n) \in O(n^{d-\epsilon}) \Rightarrow T(n) \in \Theta(n^d)$.
   The work done at the leaves dominate.

2. Case 2: $f(n) \in \Theta(n^d \log^k n) \Rightarrow T(n) \in \Theta(n^d \log^{k+1} n)$.
   There are some extensions of case 2, to be elaborated in this tutorial.

3. Case 3: $f(n) \in \Omega(n^{d+\epsilon}) \Rightarrow T(n) \in \Theta(f(n))$,
   assuming, for some constant $c < 1$, that for all x, $a \cdot f(\frac{x}{b}) \leq c \cdot f(x)$ (regularity condition).
   The work done at the root dominates.

However, there are at least three other ways to solve recurrences, especially useful when the recurrences are not of the form stated above: Telescoping (if applicable), substitution method (guess and check; need good guess(es)), or draw the recursion tree (try exploring `https://visualgo.net/en/recursion`).

1.1 Recap About Telescoping

Consider any sequence $a_0, a_1, \ldots, a_n$ and suppose we need to find $\sum_{i=0}^{n-1}(a_i - a_{i+1})$.
Expanding $\sum_{i=0}^{n-1}(a_i - a_{i+1})$, we have $(a_0 - a_1) + (a_1 - a_2) + (a_2 - a_3) + \ldots + (a_{n-1} - a_n)$.
Which can be rewritten as $a_0 + (-a_1 + a_1) + (-a_2 + a_2) + \ldots + (-a_{n-1} + a_{n-1}) - a_n$.
Thus, except for a_0 at the beginning and $-a_n$ at the end, all other a_i appear exactly once as a negative and then as a positive in the sum, and thus cancel each other, making the $\sum_{i=0}^{n-1}(a_i - a_{i+1}) = a_0 - a_n$.

2 Tutorial 02 Questions

Q1). Give a **tight** asymptotic bound for $T(n) = 4 \cdot T(\frac{n}{4}) + \frac{n}{\log n}$ **using telescoping**.

Since $a = 4$, $b = 4$, $d = \log_4 4 = 1$, and $f(n) = \frac{n}{\log n} = n^1 \log^{-1} n \in \Theta(n^d \log^{-1} n)$.
Master theorem case 2 (extension 2b when $k = -1$, see `https://en.wikipedia.org/wiki/Master_theorem_(analysis_of_algorithms)#Generic_form`) is actually applicable, and the result is:
$T(n) \in \Theta(n^d \log\log n) = \Theta(n \log\log n)$.

But you are forced to use the 'harder' telescoping method for this question...

For ease of notation, assume n is a power of 4 (to avoid floors/ceilings). If n is not a power of 4, it would only make a constant factor difference.

$$T(n) = 4 \cdot T\left(\frac{n}{4}\right) + \frac{n}{\log n} \qquad \text{(the original recurrence)}$$
$$\frac{T(n)}{n} = \frac{4 \cdot T\left(\frac{n}{4}\right)}{n} + \frac{1}{\log n} \qquad \text{(divide both LHS and RHS by } n\text{)}$$
$$\frac{T(n)}{n} = \frac{T\left(\frac{n}{4}\right)}{\frac{n}{4}} + \frac{1}{\log n} \qquad \text{(rearrange to bring 4 down to denominator)}$$
$$\frac{1}{\log n} = \frac{T(n)}{n} - \frac{T\left(\frac{n}{4}\right)}{\frac{n}{4}}$$

Now, we telescope

$$\frac{1}{\log n} = \frac{T(n)}{n} - \frac{T\left(\frac{n}{4}\right)}{\frac{n}{4}}$$
$$\frac{1}{\log \frac{n}{4}} = \frac{T\left(\frac{n}{4}\right)}{\frac{n}{4}} - \frac{T\left(\frac{n}{16}\right)}{\frac{n}{16}} \qquad \text{(substitute } n \text{ with } \frac{n}{4}\text{)}$$
$$\frac{1}{\log \frac{n}{16}} = \frac{T\left(\frac{n}{16}\right)}{\frac{n}{16}} - \frac{T\left(\frac{n}{64}\right)}{\frac{n}{64}} \qquad \text{(substitute } n \text{ with } \frac{n}{16}\text{)}$$
$$\vdots$$
$$\frac{1}{\log 4} = \frac{T(4)}{4} - \frac{T(1)}{1} \qquad \text{(final term, } n \text{ with 4)}$$

Taking the sum on both LHS and RHS:

$$\frac{1}{\log 4} + \frac{1}{\log 16} + \cdots \frac{1}{\log n} = \sum_{i=1}^{\log_4 n} \frac{1}{\log 4^i} = \frac{T(n)}{n} - \frac{T(1)}{1}$$

Finally,

$$\begin{aligned}
\sum_{i=1}^{\log_4 n} \frac{1}{i \log 4} &= \frac{T(n)}{n} - T(1) && \text{(simplify logarithms)} \\
\frac{T(n)}{n} - T(1) &\in \Theta(\log \log n) && \text{(harmonic series over } \log n\text{)} \\
\frac{T(n)}{n} &\in \Theta(\log \log n) && (T(1) \text{ is constant}) \\
T(n) &\in \Theta(n \log \log n). && \text{(multiply through by } n\text{)}
\end{aligned}$$

Q2). Give a **tight** asymptotic bound for $T(n) = 5 \cdot T(\frac{n}{3}) + n$.

1. $T(n) \in \Theta(n^2)$
2. $T(n) \in \Theta(n^{\log_5 3})$
3. $T(n) \in \Theta(n^{\log_3 5})$
4. $T(n) \in \Theta(n \log n)$
5. $T(n) \in \Theta(n)$

$T(n) = 5 \cdot T(\frac{n}{3}) + n$.
Since $a = 5, b = 3, d = \log_3 5 \approx 1.46...$, and $f(n) = n = n^1 \in O(n^{d-\epsilon})$,
Master theorem case 1 is applicable, and the result is:
$T(n) \in \Theta(n^d) = \Theta(n^{\log_3 5}) = \Theta(n^{1.46\cdots})$.

Q3). Give a **tight** asymptotic bound for $T(n) = 9 \cdot T(\frac{n}{3}) + n^3$.

1. $T(n) \in \Theta(n^9)$
2. $T(n) \in \Theta(n^3 \log n)$
3. $T(n) \in \Theta(n^2)$
4. $T(n) \in \Theta(n^3)$
5. $T(n) \in \Theta(n \log^2 n)$

$T(n) = 9 \cdot T(\frac{n}{3}) + n^3$.
Since $a = 9, b = 3, d = \log_3 9 = 2$, and $f(n) = n^3 \in \Omega(n^{d+\epsilon})$,
Master theorem case 3 is applicable, and the result is:
$T(n) \in \Theta(n^3)$.

Regularity condition holds for $c = \frac{1}{3} < 1$:
$\left\{a \cdot f\left(\frac{x}{b}\right) = 9 \cdot f\left(\frac{x}{3}\right) = 9 \cdot \frac{x^3}{3^3} = \frac{x^3}{3}\right\} \leq \left\{\frac{1}{3} \cdot f(x) = c \cdot f(x)\right\}$,

Extra remarks: If the given $f(n)$ is not of $f(n) = c \cdot n^d \log^k n$ format, there can be a case that regularity condition does **not** hold, and thus must be checked. In lec02, we had also seen that if we check regularity condition first and it is true, we must be in case 3.

Q4). Give a **tight** asymptotic bound for $T(n) = 16 \cdot T(\frac{n}{4}) + n^2 \log n$.

1. $T(n) \in \Theta(n^2 \log n)$
2. $T(n) \in \Theta(n^2 \log^2 n)$
3. $T(n) \in \Theta(n^2)$
4. $T(n) \in \Theta(n^3)$
5. $T(n) \in \Theta(n^4 \log n)$

$T(n) = 16 \cdot T(\frac{n}{4}) + n^2 \log n$.
Since $a = 16, b = 4, d = \log_4 16 = 2$, and $f(n) = n^2 \log n \in \Theta(n^d \log^1 n)$,
Master theorem case 2 is applicable, and the result is:
$T(n) \in \Theta(n^2 \log^{1+1} n) = \Theta(n^2 \log^2 n)$.

Q5). Give a **tight** asymptotic bound for $T(n) = 4 \cdot T(\frac{n}{2}) + \sqrt{n}$ **using the substitution method**.

Since $a = 4, b = 2, d = \log_2 4 = 2$, and $f(n) = \sqrt{n} = n^{0.5} \in O(n^{d-\epsilon})$,
Master theorem case 1 is applicable, and the result is:
$T(n) \in \Theta(n^d) = \Theta(n^2)$.

But you are forced to use the 'harder' substitution method for this question...

We guess $T(n) \leq c_2 \cdot n^2 - c_1 \cdot n$ (a wrong guess will make the math "not work").
For $c_1 = 1$, we set c_2 large enough so that $T(1) \leq c_2 - c_1$.

$$\begin{aligned}
T(n) &= 4 \cdot T\left(\frac{n}{2}\right) + \sqrt{n} \\
&\leq 4 \cdot \left(\frac{c_2 \cdot n^2}{2^2} - \frac{c_1 \cdot n}{2}\right) + \sqrt{n} && \text{(substitute the guessed solution)} \\
&= c_2 \cdot n^2 - 2 \cdot c_1 \cdot n + \sqrt{n} && \text{(simplify)} \\
&\leq c_2 \cdot n^2 - c_1 \cdot n + \left(\sqrt{n} - c_1 \cdot n\right) && \text{(rearrange terms)} \\
&\leq c_2 \cdot n^2 - c_1 \cdot n && \text{(since } \sqrt{n} - c_1 \cdot n < 0 \text{ for } n > 1/c_1^2\text{)}.
\end{aligned}$$

Upper bound is shown.
Lower Bound: Either show similarly, or note that $\sqrt{n}$ only gives extra cost.

Q6). Suppose that you are given k sorted arrays: $\{A_1, A_2, \ldots, A_k\}$, with n elements each.
Your task is to merge them into one combined sorted array of size $k \cdot n$.

Let $T(k, n)$ denotes the complexity of merging k arrays of size n.
Suppose that you decide that the best way to do the above is via recursion (when $k > 1$):

1. Merge the first $\lceil \frac{k}{2} \rceil$ arrays of size n,
2. Merge the remaining $\lfloor \frac{k}{2} \rfloor$ arrays of size n,
3. Merge the two sorted subarrays obtained from the first two steps above.

Give a formula for $T(k, n)$ based on the recursive algorithm above and solve the recurrence. You can assume that merging two arrays takes time proportional to the sum of the sizes of the two arrays.

The k sorted arrays is divided into two sub-problems of size $\frac{k}{2}$ sorted arrays each (rounded up or down if necessary). Finally, these two arrays, each of size $\frac{k \cdot n}{2}$, are merged with a cost of:

$$\frac{k \cdot n}{2} + \frac{k \cdot n}{2} = c \cdot k \cdot n, \quad \text{(for some constant } c\text{)}.$$

Thus, the recurrence is:

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

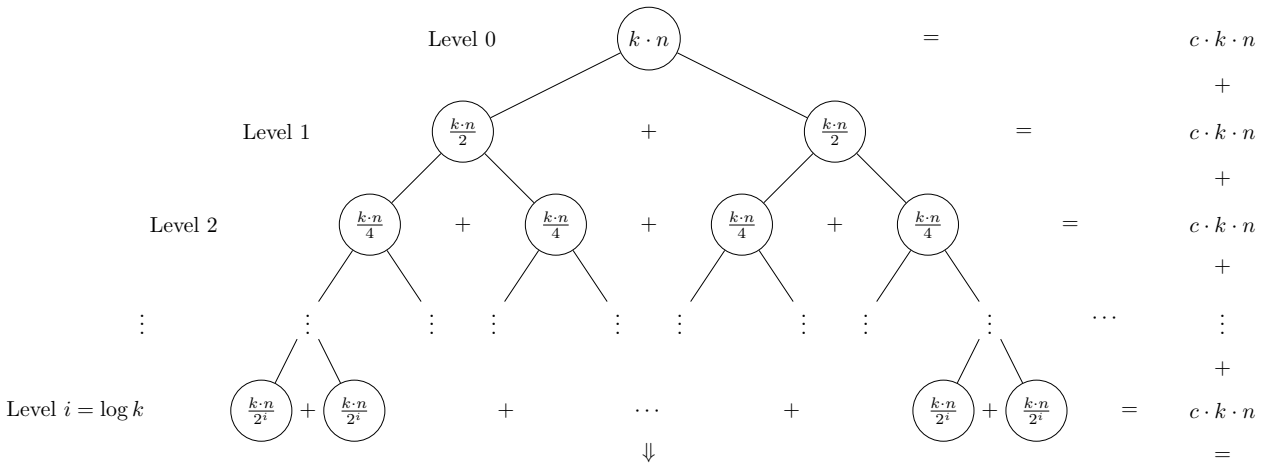
This recurrence does not fit the Master Theorem directly. Instead, a recursion tree can be used (base case $k = 1$, no merging). The tree height is $\log k$ (or explictly $\log_2 k$). At level i (root is level 0), there are 2^i subproblems, each costing $\frac{kn}{2^i}$:

$$\text{Cost at level } i = 2^i \cdot \frac{k \cdot n}{2^i} = c \cdot k \cdot n.$$

Since there are $1 + \log k$ levels, the total cost is:

$$T(k, n) = \sum_{i=0}^{\log k} c \cdot k \cdot n = c \cdot k \cdot n \cdot (1 + \log k) \in \Theta(kn \log k)$$

The recursion tree diagram is shown below:



$$\sum_{i=0}^{\log k} 2^i \cdot \frac{c \cdot k \cdot n}{2^i} \quad = \quad \sum_{i=0}^{\log k} c \cdot k \cdot n = c \cdot k \cdot n \cdot (1 + \log k) \quad \Leftrightarrow \quad \Theta\left(kn \log k\right)$$