NUS | Computing
National University of Singapore

CS3230
eric_han@nus.edu.sg
https://eric-han.com
Computer Science

T02 – Week 3

Recurrences and Master Theorem

CS3230 – Design and Analysis of Algorithms

Recap from T01

Based on the post-tutorial poll, here are some advice: Here are some links (easy reads) to help you proof them for yourself. The **useful part of these properties are to be applied**.

Q1 Limits > Complexity

Read more and proof it both ways $\iff$!

- freecodecamp

Q2 Complexity Properties

Understand, proof the basic properties for yourself, and some additional properties.

- CLRS Pg. 61
- Stack Overflow, additional properties
- geeksforgeeks

Practical T01

Show me and take a snack!

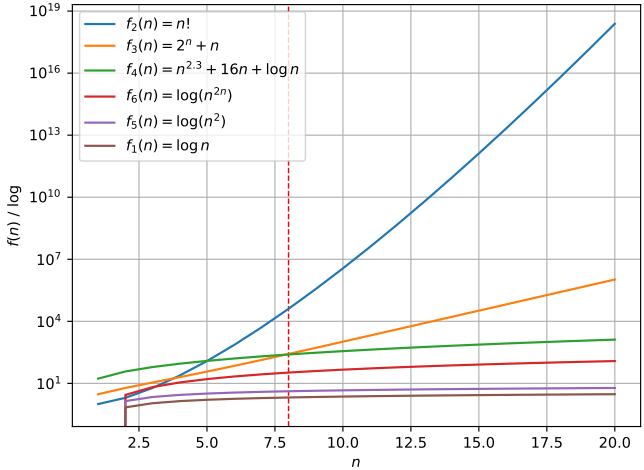

Figure 1: Functions compared graphically; Plotting/Practical also helps!

Abridged Admin

Repeated for the benefit of those joining us this week:

- **Policies**: Tutorials & Assignment Policy: Plagiarism (no AI tools).
- **Tutorial Discussion**: Learning is **social** and I hope that we are able to build friendships in this class. Identified by **[G]**.
- **Participation** (10 marks): *5 marks* for each of two presentations. Identified by presenter 1,2,3 (may be part of qn/etc): **[P1/2/3]**. Help yourself to the snacks.
- **Telegram Group**: Join for updates.
- **Consultations**: Wed 10-11 AM.
- **Assignments**: Good job on submission of A01, will be marking over the next few days.
- **Questions?**: Use ChatGPT, Telegram Group / PM `@Eric_Vader`.

Master Theorem: Asymptotic Bounds for $T(n)$

For recurrence $T(n) = a \cdot T(\frac{n}{b}) + f(n)$ with $f(n) = c \cdot n^m \log^k n$, the asymptotic behavior is determined by comparing $f(n)$ to n^d, where $d = \log_b a$:

1. **Case 1**: $f(n) \in O(n^{d-\epsilon}) \Rightarrow T(n) \in \Theta(n^d)$
   Dominated by work at the leaves.
2. **Case 2**: $f(n) \in \Theta(n^d \log^k n) \Rightarrow T(n) \in \Theta(n^d \log^{k+1} n)$
   Balanced contributions at all levels.
   (Extensions of this case will be covered.)
3. **Case 3**: $f(n) \in \Omega(n^{d+\epsilon}) \Rightarrow T(n) \in \Theta(f(n))$
   Dominated by work at the root.
   (Requires a regularity condition: $a \cdot f(\frac{x}{b}) \leq c \cdot f(x)$ for some $c < 1$.)

Alternative Methods for Solving Recurrences

When the recurrence does not fit the standard form, consider these approaches:

1. **Telescoping**:
   Simplify by collapsing terms across recursive levels (works when applicable).
2. **Substitution Method**:
   Guess and check by proposing a solution and verifying through induction.
   (Requires good initial guess(es)).
3. **Recursion Tree**:
   Visualize the recurrence as a tree, summing contributions across levels.
   (Interactive tool: Visualgo).

Survey Results

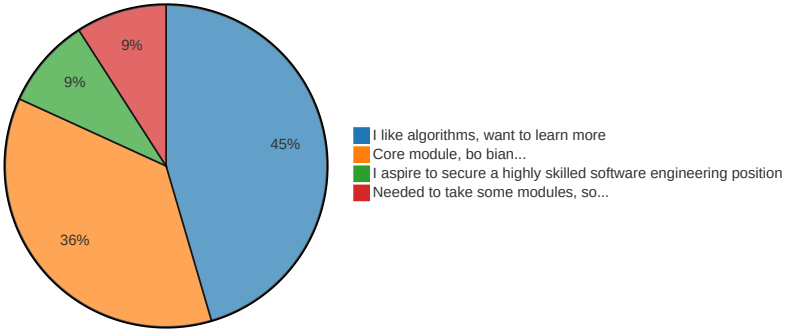

Figure 2: Why are you taking CS3230?

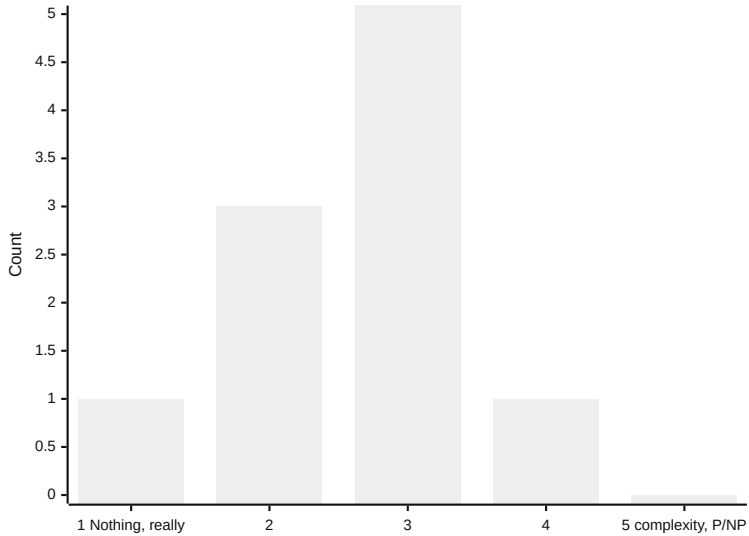

Figure 3: How much do you know about algorithms/analysis?

Want to achieve

1. 4x Understand the lecture and assignments better
2. 2x Improve problem solving + Being able to understand the main concept of algo and how to tailor solutions to solve leetcode challenges.
3. Learn some cool interesting stuff and also make sure I understand what is being taught in the course
4. delve deeper into a concept without it being redundant/inaccessible
5. Learn enough to get good enough score
6. have fun discussions

Suggestions

1. Go at a slow pace and fully utilise the class time
2. Post- class survey for questions not understood by majority so next class can go over it briefly?

Question 1 [G]/[P1]

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

Question 1 Variant [G]

Use Master Theorem instead (See Master Theorem Wiki).

Recap

- What is telescoping?
- **tight** asymptotic bound $= \cdots$?

Answer

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.

Rearrange for telescoping

$$T(n) = 4 \cdot T\left(\frac{n}{4}\right) + \frac{n}{\log n} \qquad \text{(the original recurrence)}$$

$$\frac{T(n)}{n} = \frac{4 \cdot T(\frac{n}{4})}{n} + \frac{1}{\log n} \qquad \text{(divide both LHS and RHS by } n\text{)}$$

$$\frac{T(n)}{n} = \frac{T(\frac{n}{4})}{\frac{n}{4}} + \frac{1}{\log n} \qquad \text{(rearrange to bring } 4 \text{ down to denominator)}$$

$$\frac{1}{\log n} = \frac{T(n)}{n} - \frac{T(\frac{n}{4})}{\frac{n}{4}}$$

Note: It is possible to **not** divide by n, one would need to multiply the correct multiple to each subsequent equation to cancel-out.

Telescope

$$\frac{1}{\log n} = \frac{T(n)}{n} - \frac{T\left(\frac{n}{4}\right)}{\frac{n}{4}}$$

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

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

$$\vdots$$

$$\frac{1}{\log 4} = \frac{T(4)}{4} - \frac{T(1)}{1} \qquad \text{(final term, } n \text{ with } 4\text{)}$$

Taking the sum on both LHS and RHS:

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

Manipulate

$$\sum_{i=1}^{\log_4 n} \frac{1}{i \log 4} = \frac{T(n)}{n} - T(1) \qquad \text{(simplify logarithms)}$$

$$\frac{T(n)}{n} - T(1) \in \Theta(\log \log n) \qquad \text{(harmonic series over } \log n\text{)}$$

$$\frac{T(n)}{n} \in \Theta(\log \log n) \qquad (T(1) \text{ is constant})$$

$$T(n) \in \Theta(n \log \log n). \qquad \text{(multiply through by } n\text{)}$$

See Master Theorem Case 2, extension 2b: When $f(n) = \Theta\left(n^d \log^k n\right)$, $k = -1$, then $T(n) = \Theta\left(n^d \log\log n\right)$.

Answer (Master Theorem)

$$T(n) = 4 \cdot T\left(\frac{n}{4}\right) + \frac{n}{\log n}$$

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 2b is applicable, and the result is:

$$T(n) \in \Theta(n^d \log\log n) = \Theta(n \log\log n).$$

Answer (Alt Telescope)

$$4^0 \cdot \frac{n}{\log n} = T(n) - 4 \cdot T\left(\frac{n}{4}\right) \qquad \text{(From the recurrence relation)}$$

$$4^1 \cdot \frac{n}{\log \frac{n}{4}} = 4 \cdot T\left(\frac{n}{4}\right) - 16 \cdot T\left(\frac{n}{16}\right) \qquad (n \to \frac{n}{4}\text{, multiply by } 4)$$

$$4^2 \cdot \frac{n}{\log \frac{n}{16}} = 16 \cdot T\left(\frac{n}{16}\right) - 64 \cdot T\left(\frac{n}{64}\right) \qquad (n \to \frac{n}{16}\text{, multiply by } 4^2)$$

$$\vdots$$

$$4^k \cdot \frac{n}{\log 4} = 4^k \cdot T(4) - 4^{k+1} \cdot T(1) \qquad (n \to 4^{k+1}\text{, multiply by } 4^k)$$

Sometimes, by choosing to do this way without dividing would cause one to be stuck (but sometimes it would work - try on the lecture example). In this question, going about the dividing by n method is the best way.

Question 2

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

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. $T(n) \in \Theta(n)$

Answer

$$T(n) = 5 \cdot T\left(\frac{n}{3}\right) + n$$

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:

$$T(n) \in \Theta(n^d) = \Theta(n^{\log_3 5}) = \Theta(n^{1.46...}).$$

Question 3 [P2]

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

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. $T(n) \in \Theta(n \log^2 n)$

Answer

$$T(n) = 9 \cdot T\left(\frac{n}{3}\right) + n^3$$

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 the form $f(n) = c \cdot n^d \log^k n$, the regularity condition may **not** hold, and it must be checked. In lecture 2, we saw that if the regularity condition holds, we are in case 3.

Question 4

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. $T(n) \in \Theta(n^4 \log n)$

Answer

$$T(n) = 16 \cdot T\left(\frac{n}{4}\right) + n^2 \log n$$

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:

$$T(n) \in \Theta(n^2 \log^{1+1} n) = \Theta(n^2 \log^2 n).$$

Question 5 [G]/[P3]

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

Question 5 Variant [G]

Use Master Theorem instead.

Recap

- What is substitution method?

Answer

Upper bound

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 $T(1) \leq c_2 - c_1$.

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

Lower Bound

For the lower bound, we *guess* $T(n) \geq c_3 \cdot n^2 - c_4 \cdot n$, where c_3 and c_4 are positive constants.

$$
\begin{aligned}
T(n) &= 4 \cdot T\left(\frac{n}{2}\right) + \sqrt{n} \\
&\geq 4 \cdot \left(\frac{c_3 \cdot n^2}{2^2} - \frac{c_4 \cdot n}{2}\right) + \sqrt{n} && \text{(substitute the guessed solution)} \\
&= c_3 \cdot n^2 - 2 \cdot c_4 \cdot n + \sqrt{n} && \text{(simplify)} \\
&\geq c_3 \cdot n^2 - c_4 \cdot n && \text{(since } \sqrt{n} \text{ adds extra positive cost).}
\end{aligned}
$$

Since $c_3 \cdot n^2 - c_4 \cdot n \leq T(n) \leq c_2 \cdot n^2 - c_1 \cdot n$, then:

$$T(n) \in \Theta(n^2).$$

Answer (Master Theorem)

$$T(n) = 4 \cdot T\left(\frac{n}{2}\right) + \sqrt{n}$$

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:

$$T(n) \in \Theta(n^d) = \Theta(n^2).$$

Answer (Direct)

$$
\begin{aligned}
T(n) &= 4 \cdot T\left(\frac{n}{2}\right) + \sqrt{n} && \text{(Given recurrence relation)} \\
&= 4 \cdot \left[4 \cdot T\left(\frac{n}{4}\right) + \sqrt{\frac{n}{2}}\right] + \sqrt{n} && \text{(Expand } T\left(\frac{n}{2}\right)\text{)} \\
&= 4^2 \cdot T\left(\frac{n}{4}\right) + 4 \cdot \sqrt{\frac{n}{2}} + \sqrt{n} && \text{(Simplify)} \\
&= 4^2 \cdot \left[4 \cdot T\left(\frac{n}{8}\right) + \sqrt{\frac{n}{4}}\right] + 4 \cdot \sqrt{\frac{n}{2}} + \sqrt{n} && \text{(Expand } T\left(\frac{n}{4}\right)\text{)} \\
&= 4^3 \cdot T\left(\frac{n}{8}\right) + 4^2 \cdot \sqrt{\frac{n}{4}} + 4 \cdot \sqrt{\frac{n}{2}} + \sqrt{n} && \text{(Simplify)} \\
&\vdots && \text{(Repeat expansion)} \\
&= 4^k \cdot T\left(\frac{n}{2^k}\right) + \sum_{i=0}^{k-1} 4^i \cdot \sqrt{\frac{n}{2^i}} && \text{(Generalize after } k \text{ steps)}
\end{aligned}
$$

$$
\begin{aligned}
T(n) &= 4^{\log_2 n} \cdot T(1) + \sum_{i=0}^{\log_2 n - 1} 4^i \cdot \sqrt{\frac{n}{2^i}} && \text{(Substitute } k = \log_2 n\text{)} \\
&= n^2 \cdot T(1) + \sum_{i=0}^{\log_2 n - 1} 4^i \cdot \sqrt{\frac{n}{2^i}} && (4^{\log_2 n} = n^2) \\
&= n^2 \cdot T(1) + \sqrt{n} \cdot \sum_{i=0}^{\log_2 n - 1} 2^{3i/2} && \text{(Factorize } \sqrt{n}\text{)} \\
&= n^2 \cdot T(1) + \sqrt{n} \cdot \frac{1 - (2^{3/2})^{\log_2 n}}{1 - 2^{3/2}} && \text{(Geometric series)} \\
&= n^2 \cdot T(1) + \sqrt{n} \cdot \frac{1 - n^{3/2}}{1 - 2^{3/2}} && (n^{3/2} = (2^{3/2})^{\log_2 n}) \\
&\in \Theta(n^2).
\end{aligned}
$$

Question 6 [G]

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 $T(k, n)$ denote 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.

Recap

- What is recursion tree?

Answer

The original problem with 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 directly fit into the Master Theorem. Instead, we use a **recursion tree** to analyze the cost.

Recursion Tree Analysis

Base Case

When $k = 1$, No merging is needed as there's only one array.

Tree Height

The height of the recursion tree is $\log_2 k$, as each recursive step divides the number of arrays into halves until only one array remains.

Cost at Each Level i (starting at level 0)

There are 2^i subproblems, each with a cost of $\frac{kn}{2^i}$:

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

The cost at each level is constant, $c \cdot k \cdot n$, for all levels of the recursion tree.

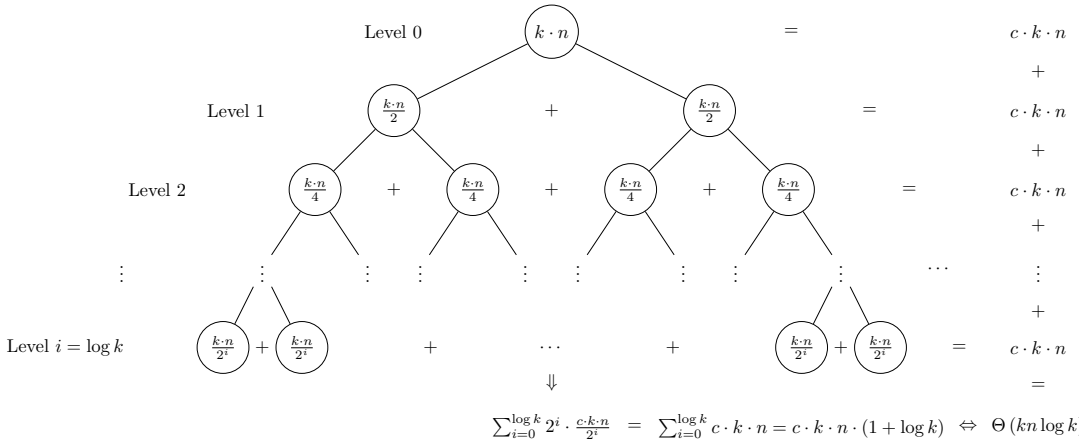

Figure 4: We can sum across all levels from 0 to $\log k$, resulting in $1 + \log k$ levels.

Practical [Optional]

Practical repo: To help you further your understanding, not compulsory; Work for Snack!

1. Implement Q6 in code, `merge_k_sorted_arrays` to return the merge the arrays, and the number of operations.
2. Check that you get this output:

```
Merged array: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
Number of operations (count): 36
Number of operations (theory): 36.0
```