CS3230 Semester 2 2025/2026
Design and Analysis of Algorithms

Tutorial 02
Recurrences and Master Theorem
For Week 03

Document is last modified on: January 20, 2026

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 dominates.
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.
   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 $\sum_{i=0}^{n-1}(a_i - a_{i+1}) = a_0 - a_n$.

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**.

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

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

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

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

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.