CS3230 Semester 1 2024/2025
Design and Analysis of Algorithms

Tutorial 02
Solving Recurrences and Master Theorem
For Week 03

Document is last modified on: July 24, 2024

1 Lecture Review: Recurrences

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

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

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

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

3. Case 3: $f(n) = \Omega(n^{d+\epsilon}) \Rightarrow T(n) = \Theta(f(n))$,
   assuming $a \cdot f(\frac{x}{b}) \leq c \cdot f(x), \forall x$, and some constant $c < 1$ (regularity condition).
   The root does most of the work.

However, there are also three other ways to solve recurrences, especially those that are not of the standard form 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**.

Q2-3-4). are hidden, they are of type:
Give a **tight** asymptotic bound for $T(n) = a \cdot T(\frac{n}{b}) + f(n)$.
But we guarantee that all three can be solved (easily) with master theorem.

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.