50.004 Introduction to Algorithms, Fall 2015
Singapore University of Technology and Design — 29 Sep, 2015
Simon Lui — Quiz - 1

Problem 1. Asymptotic Growth

Consider the functions

$$f_1(n) = (\log_2(n))^2,\ f_2(n) = \log_e(2^{\log_2(n)}),\ f_3(n) = \log_2(n!),\ f_4(n) = 5^{(n + \log_2(n))}.$$

Q1 (8 points): Compute the asymptotic complexity of f_1, f_2, f_3, f_4.

Solution:

$$f_1(n) = \Theta(\log_2(n))^2\ ,\ f_2(n) = \Theta(\log_e(n))\ ,\ f_3(n) = \Theta(n\log_2(n))\ ,\ f_4(n) = \Theta(5^n).$$

1. Cannot simply f_1 more. Since every function is Θ of itself, hence, $f_1(n) = \Theta(\log_2(n))^2$.
2. Apply logarithm property, $x^{(\log_b(n))} = n^{(\log_b(x))}$, where b, x and n are positive real numbers and $b \neq 1$. Then $2^{(\log_2(n))} = n^{(\log_2(2))} = n$, then $f_2 = \log_e(n)$, and then, same as the reasoning of 1) every function is Θ of itself, hence $f_2(n) = \log_e(n)$.
3. Using Stirling's approximation $n! \approx \sqrt{2\pi n}(\frac{n}{e})^n$ and the properties of logarithm

$$\begin{aligned}
\log_2(n!) &\approx \log_2(\sqrt{2\pi n}(\frac{n}{e})^n) \\
&\approx \log_2(\sqrt{2\pi n}) + \log_2(\frac{n}{e})^n \\
&\approx \frac{1}{2}log_2(2\pi n) + n\log_2(\frac{n}{e}) \\
&\approx \frac{1}{2}log_2(2\pi n) + n\log_2(n) - n\log_2(e) \\
&\approx \Theta(\log_2 n) + \Theta(n\log_2 n) - \Theta(n) \\
&\approx \Theta(n\log_2 n). \text{ (Dropping the low-order terms)}
\end{aligned}$$

4. As polynomial function n grows faster than polylogarithmic function $\log_2 n$, hence, by dropping the low-order term, $f_4 = \Theta(5^n)$.

Q2 (4 points): Rank the above functions by increasing order of growth.

Solution:

$$f_2 < f_1 < f_3 < f_4$$

Problem 2. Sorting

Given an array of size n, $A[1, \cdots, n]$:

Q1 (6 points): We first split the array into two equal-size pieces, sort the two pieces using insertion sort method, and then merge them into one array. Write down the complexity (or running time)$T(n)$ of this method.
(Hint: Recall the complexity of insertion sort.)

Solution:

- Time for diving the array into two pieces: $\Theta(1)$;
- Time for sorting the two pieces using insertion sort method: $2\Theta((n/2)^2) = \Theta(n^2)$;
- Time for merging:$\Theta(n)$.

Therefore, the complexity of this method: $T(n) = \Theta(n^2) + \Theta(n) + \Theta(1) = \Theta(n^2)$

Q2 (6 points): Use Merge-Sort, we split the array into four equal-size pieces, then recursively sort each piece, and then merge them into one sorted array. Write and solve the recurrence related to the complexity of this **Four-way Merge-Sort**.

Solution:

The recurrence:

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

Solve the recurrence using master theorem case 2, we have: $T(n) = \Theta(nlogn)$.

Q3 (2 points): Compare the sorting method in **Q1** and **Q2**, which one is better, and why?

Solution:

The sorting method in **Q2** is better.

Problem 3.Msater Theorem

Q1 (6 points): $T\left(n\right) = 5T\left(\frac{n}{2}\right) + \Theta\left(n^3\right)$

Solution:

Case 3. For some constant ε, we have $\log_2 5 + \varepsilon = 3$, $T(n) = \Theta(n^3)$.

Q7 (6 points): $T(n) = 3T\left(\frac{n}{5}\right) + \log^2 n$

Solution:

Case 1. $T(n) = \Theta\left(n^{\log_5 3}\right)$

Q3 (7 points): $T(n) = T(\sqrt{n}) + 1$. (Hint: Variable change: $m = \log n$.)

Solution:

Let $m = \log n$, and$S(m) = T(2^m)$, $T(2^m) = T(2^{m/2}) + 1$, so we have $S(m) = S(m/2) + 1$. Using the master theorem, $n^{\log_b a} = 1$ and $f(n) = 1$, Case 2 applies, and $S(m) = \Theta(\log m)$. Therefore, $T(n) = \Theta(\log \log n)$

Bonus Problem (15 points)

Consider the recurrence:

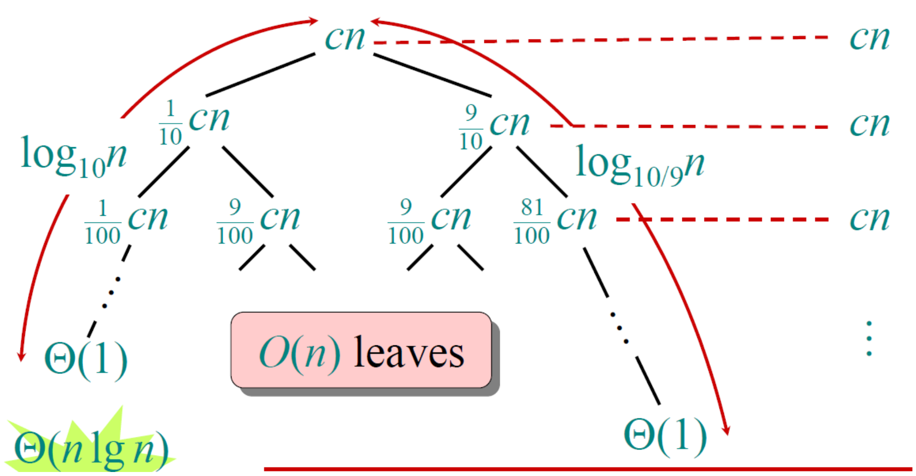
$$T(n) = T(\frac{n}{10}) + T(\frac{9n}{10}) + \Theta(n)$$

Can we apply Master theorem directly in this case? If **yes**, state your answer. If **no**, try to use a recursion tree to solve the recurrence.

Solution:

In this case, we cannot apply master theorem directly.

Draw the recursion tree as below:



$$cn\log_{10}n + O(n) \leq T(n) \leq cn\log_{10/9}n + O(n)$$

we have: $T(n) = O(n \log n)$ and also $T(n) = \Omega(n \log n)$, therefore $T(n) = \Theta(n \log n)$