Master Theorem / Recurrence Equations / Asymptotic Growth

---

1. Find the time complexity of following algorithm:

TERNARY-SEARCH(x, A, i, j)

// Assumption: $A[i] \leq x < A[j]$
if $j - i \leq 1$:
    return i
$p = \frac{2}{3}i + \frac{1}{3}j$
$q = \frac{1}{3}i + \frac{2}{3}j$
if $x < A[p]$:
    return TERNARY-SEARCH(x, A, i, p)
elseif $A[p] \leq x < A[q]$:
    return TERNARY-SEARCH(x, A, p, q)
elseif $x \geq A[q]$:
    return TERNARY-SEARCH(x, A, q, j)

$$\boxed{\begin{aligned} T(n) &= T(n/3) + 1 \\ T(1) &= 1 \end{aligned}}$$

2. If f(n) = Θ(g(n)) and g(n) = Θ(h(n)), then h(n) = Θ(f(n)). True/False

   Solution: True. Θ is transitive and symmetric.

3. If f(n) = O(g(n)) and g(n) = O(f(n)) then f(n) = g(n). True/False

   Solution: False: f(n) = n and g(n) = n + 1.

4. Solving recurrences: Give solutions to the following.

$$T(n) = 8\,T(n/3) + n^2$$

   Solution: $T(n) = \Theta(n^2)$ by the Master Theorem

$$T(n) = 10\,T(n/3) + n^2$$

   Solution: $T(n) = \Theta(n^{\log_3 10})$ by the Master Theorem

$$T(n) = 2\,T(n/2) + n$$

   Solution: $T(n) = \Theta(n \log n)$ by the Master Theorem

5. What is the recurrence relation for the time to naively calculate (using T- notation) the *n'th* factorial number? Assume multiplication between two numbers takes constant time. Then solve this recurrence.

   Solution: $T(n) = T(n-1) + O(1)$, this is $T(n) = \Theta(n)$.

6. Find an asymptotic solution of the following functional recurrence. Express your answer using Θ-notation, and give a brief justification.

$$T(n) = 16\,T(n/4) + n^2 \log^3 n$$

   Solution: Using Master Theorem, we compare $n^2 \log^3 n$ with $n^{\log_4 16} = n^2$. This is case 2 of the generalized version of the theorem as treated in class, so we increment the $\log^k n$ for $\Theta(n^2 \log^4 n)$.

7. Find an asymptotic solution of the following functional recurrence. Express your answer using Θ-notation.

$$T(n) = 9\,T(n/3) + n^3$$

   Solution: Using the master theorem, $a = 9$, $b = 3$, and $\log_b a = \log_3 9 = 2$. Thus, we compare $n^{\log_b a} = n^2$ to $f(n) = n^3$. Since $n^3 = \Omega(n^{2+\varepsilon})$, $f(n)$ dominates the recurrence and we are in Case 3 of the master theorem. Thus, $T(n) = \Theta(n^3)$.

8. Running merge sort on an array of size n which is already correctly sorted takes O(n) time.

   Solution: FALSE. The merge sort algorithm presented in class always divides and merges the array $O(\log n)$ times, so the running time is always $O(n \log n)$.

9. Use the Master Theorem to find the runtime of a recursive algorithm whose execution time is given by the formula:

$$T(n) = 2\,T(n-1) + \log n.$$

   Hint: The Master Theorem cannot be used on the given formula as it stands. Consider what would happen if you substitute one of $n = 2^m$, $n = \log m$, or $n = m^2$ for n. Identify which substitution allows you to apply the Master Theorem, and use it to find the runtime of $T(n)$.

Solution

The correct substitution is $n = log\, m$. Substituting gives:

$$T(log\, m) = 2T(log\, m - 1) + log\, log\, m$$

Changing 1 to $log_2(2)$ gives:

$$T(log\, m) = 2T(log\, m - log_2(2)) + log\, log\, m$$

By the rule for division in logarithms, we can combine $log\, m - log_2(2)$ to give the result:

$$T(log\, m) = 2T\left(log\left(\frac{m}{2}\right)\right) + log\, log\, m$$

Then, we substitute the function $S(m) = T(log\, m)$ to get:

$$S(m) = 2S\left(\frac{m}{2}\right) + log\, log\, m$$

We can use Case 1 of Master Theorem on this recurrence.

$$S(m) = \Theta(m)$$

Substituting $m = 2^n$ gives the final runtime of $T(n)$:

$$T(n) = \Theta(2^n)$$

10. Solve the following recurrences, expressing your solution using asymptotic Θ notation:

$$T(n) = 9\ T(n/3) + \Theta\ (n \log n)$$

Solution: Because $n^{\log_3 9} = n^2$, and n^2 is asymptotically greater than n log n by more than polylogarithmic factor, this is case 1 of the Master Theorem, and $T(n) = \Theta(n^2)$.

$$T(n) = T(n/2) + \Theta\ (\log n)$$

Solution: Because $n^{\log_2 1} = 1$, which is the same as log n to within a polylogarithmic factor (that is, log n), this is case 2 of the Master Theorem, and T (n) = log n (log n), or $\log^2 n$.

11. Is it always true that f (n) + g(n) = Θ (*min*{f (n), g(n)})? If so, prove it. If not, find a counterexample and show that this statement is false.

Solution: This claim is false. In fact, f (n)+g(n) = Θ (*max* {f (n), g(n)}). A possible counter example is f(n) = 1 and g(n) = n. It is clear that g(n) asymptotically dominates f(n). Therefore, f(n) + g(n) should be Θ(g(n)) = Θ(n). However, the claim stipulates that f (n) + g(n) = Θ (*min* {f (n), g(n})) = Θ (1), which is incorrect.

12. Solve these recurrences:

$$T(n) = 4T(n/2) + \Theta(n^2)$$

Solution: This is case 2 of the master method and thus $\Theta\ (n^2 \log n)$.

$$T(n) = T(4n/5) + \Theta(n)$$

Solution: This is case 3 of the master method and thus Θ(n).

13. Solve these recurrences:

$$T(n) = 4T(n/2) + \Theta(n^2)$$

Solution: This is case 2 of the master method and thus Θ (n^2 log n).

$$T(n) = T(4n/5) + \Theta(n)$$

Solution: This is case 3 of the master method and thus Θ(n).

14. Find a solution to the recurrence $T(n) = T(n/3) + T(2n/3) + \Theta(n)$

Solution: Draw recursion tree. At each level, do Θ(n) work. Number of levels is l. $\log_{3/2} n = \Theta(\log n)$, so guess $T(n) = \Theta(n \log n)$ and use the substitution method to verify guess.

15. Order the function based on asymptotic growth

$$f_1(n) = 8\sqrt{n}, \quad f_2(n) = 25^{1000}, \quad f_3(n) = (\sqrt{3})^{\lg n}$$

Solution: f_2, f_1 , f_3

$$f_1(n) = \frac{1}{100}, \quad f_2(n) = \frac{1}{n}, \quad f_3(n) = \frac{\lg n}{n}$$

Solution: f_2, f_3, , f_1

$$f_1(n) = 2^{\lg^3 n}, \quad f_2(n) = n^{\lg n}, \quad f_3(n) = \lg n!$$

Solution: f_3, f_2, f_1

Reference: From “6.006: Introduction to Algorithms”, MIT.