CSCI 200

Master Theorem

The Master Theorem applies to recurrences of the following form:

$$T(n) = \begin{matrix} c, & \text{for } n < d \\ aT(n/b) + f(n), & \text{for } n \geq d \end{matrix}$$

where c and d are constants, $a \geq 1$ and $b > 1$ are constants, and $f(n)$ is an asymptotically positive function. Here, a represents the number of sub-problems, n/b is the size of each of those sub-problems, and $f(n)$ is the non-recursive overhead. There are three cases:

1. If $f(n) = O(n^{\log_b a - \epsilon})$ for some constant $\epsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$.
2. If $f(n) = \Theta(n^{\log_b a} \log^k n)$ with $k \geq 0$, then $T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$.
3. If $f(n) = \Omega(n^{\log_b a + \epsilon})$ for some constant $\epsilon > 0$, and $f(n)$ satisfies the regularity condition, then $T(n) = \Theta(f(n))$.
   Regularity condition: $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n.

Assuming the regularity condition holds, another way to think of this is evaluating what we call a **critical function** $n^{\log_b a}$ and comparing it to the non-recursive overhead $f(n)$. Then, the three cases are:

Case	**Condition**	**Result**
1.	$n^{\log_b a}$ is polynomially larger than $f(n)$	$T(n) = \Theta(n^{\log_b a})$
2.	$n^{\log_b a}$ has the same value as $f(n)$, up to some logarithmic power k	$T(n) = \Theta(n^{\log_b a} \log^{k+1} n)$
3.	$n^{\log_b a}$ is polynomially smaller than $f(n)$	$T(n) = \Theta(f(n))$

Practice Problems

1. $T(n) = 4T(n/2) + n$
2. $T(n) = 2T(n/2) + n \log n$
3. $T(n) = T(n/3) + n \log n$
4. $T(n) = 8T(n/2) + n^2$
5. $T(n) = 9T(n/3) + n^3$
6. $T(n) = T(n/2) + 1$

7. $T(n) = 2T(n/2) + \log n$

8. $T(n) = 2T(n/2) + 1$

9. $T(n) = 3T(n/2) + n^2$

10. $T(n) = 4T(n/2) + n^2$

11. $T(n) = 4T(n/2) + n^2 \log^2 n$

12. $T(n) = 4T(n/2) + n^2$

13. $T(n) = T(n/2) + 2^n$

14. $T(n) = 3T(n/3) + \sqrt{n}$

15. $T(n) = 4T(n/2) + cn$, where c is a constant

16. $T(n) = 3T(n/4) + n \log n$

17. $T(n) = 3T(n/3) + n/2$

18. $T(n) = 6T(n/3) + n^2 \log n$

19. $T(n) = 7T(n/3) + n^2$

20. $T(n) = 2T(n/4) + n^{0.51}$

21. $T(n) = 9(n/3) + n^2 \log^4 n$

Solutions

1. $T(n) = 4T(n/2) + n$ Case 1 - $T(n) = \Theta(n^2)$
2. $T(n) = 2T(n/2) + n \log n$ Case 2 with $k = 1$ - $T(n) = \Theta(n \log^2 n)$
3. $T(n) = T(n/3) + n \log n$ Case 3 - $T(n) = \Theta(n \log n)$
4. $T(n) = 8T(n/2) + n^2$ Case 1 - $T(n) = \Theta(n^3)$
5. $T(n) = 9T(n/3) + n^3$ Case 3 - $T(n) = \Theta(n^3)$
6. $T(n) = T(n/2) + 1$ (this is recurrence for binary search) Case 2 with $k = 0$ - $T(n) = \Theta(\log n)$
7. $T(n) = 2T(n/2) + \log n$ (this is recurrence for heap construction) Case 1 - $T(n) = \Theta(n)$
8. $T(n) = 2T(n/2) + 1$ Case 1 - $T(n) = \Theta(n)$
9. $T(n) = 3T(n/2) + n^2$ Case 3 - $T(n) = \Theta(n^2)$
10. $T(n) = 4T(n/2) + n^2$ Case 2 with $k = 0$ - $T(n) = \Theta(n^2 \log n)$
11. $T(n) = 4T(n/2) + n^2 \log^2 n$ Case 2 with $k = 2$ - $T(n) = \Theta(n^2 \log^3 n)$
12. $T(n) = 4T(n/2) + n^2$ Case 2 with $k = 0$ - $T(n) = \Theta(n^2 \log n)$
13. $T(n) = T(n/2) + 2^n$ Case 3 - $T(n) = \Theta(2^n)$
14. $T(n) = 3T(n/3) + \sqrt{n}$ Case 1 - $T(n) = \Theta(n)$
15. $T(n) = 4T(n/2) + cn$, where c is a constant Case 1 - $T(n) = \Theta(n^2)$
16. $T(n) = 3T(n/4) + n \log n$ Case 3 - $T(n) = \Theta(n \log n)$
17. $T(n) = 3T(n/3) + n/2$ Case 2 with $k = 0$ - $T(n) = \Theta(n \log n)$
18. $T(n) = 6T(n/3) + n^2 \log n$ Case 3 - $T(n) = \Theta(n^2 \log n)$
19. $T(n) = 7T(n/3) + n^2$ Case 3 - $T(n) = \Theta(n^2)$
20. $T(n) = 2T(n/4) + n^{0.51}$ Case 3 - $T(n) = \Theta(n^{0.51})$
21. $T(n) = 9(n/3) + n^2 \log^4 n$ Case 2 with $k = 4$, $T(n) = \Theta(n^2 \log^5 n)$