The master method for solving recurrences

CS-6th

Instructor: Dr. Ayesha Enayet

Definition

- A master recurrence describes the running time of a divide-and-conquer algorithm that divides a problem of size n into a subproblems, each of size $n/b<n$.
- For solving algorithmic recurrences of the form:
  - $T(n)=aT(n/b)+f(n)$
  - Where $a>0$ and $b>1$ are constants.
  - Merge Sort Example: $T(n)=2T(n/2)+\Theta(n)$, where $a=2$ and $b=2$ and $f(n)=\Theta(n)$

For cases where n is an odd number

- To ensure that the problem sizes are integers, we round one subproblem down to size $\lfloor n/2 \rfloor$and the other up to size $\lceil n/2 \rceil$, so the true recurrence for T(n)=2T(n/2)+ Θ(n), where n is an odd value, is T(n)=T($\lceil n/2 \rceil$)+T($\lfloor n/2 \rfloor$)+Θ(n).

Master Theorem

- Let a>0 and b>1 be constants, and let f(n) be a driving function that is defined and nonnegative on all sufficiently large reals. Define the recurrence T(n) on n∈N by
  - T(n)=aT(n/b) + f(n),
  - Where aT(n/b) actually means a'T($\lfloor n/b \rfloor$) +a'' T($\lceil n/b \rceil$) for some constants a'>=0 and a''>=0 satisfying a=a'+a''. Then the asymptotic behavior of T(n) can be characterized using three cases.

Master Theorem

1. If there exists a constant $\epsilon > 0$ such that $f(n) = O(n^{\log_b a - \epsilon})$, then $T(n) = \Theta(n^{\log_b a})$.
2. If there exists a constant $k \geq 0$ such that $f(n) = \Theta(n^{\log_b a} \lg^k n)$, then $T(n) = \Theta(n^{\log_b a} \lg^{k+1} n)$.
3. If there exists a constant $\epsilon > 0$ such that $f(n) = \Omega(n^{\log_b a + \epsilon})$, and if $f(n)$ additionally satisfies the ***regularity condition*** $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$. ∎

Simplified

- *We can express f(n) as f(n)= $\Theta(n^x lg^y n)$, where x is power of n and y is power of lg*
- *Case 1: If $lg_b a > x$, then:*
  - *T(n)= $\Theta(n^{lg_b a})$.*
- *Case 2: if $lg_b a = x$:*
  - *If y>-1 then T(n)=$\Theta(n^x lg^{y+1} n)$*
  - *If y=-1 then $\Theta(n^x lglgn)$*
  - *If y<-1 then $\Theta(n^x)$*
- *Case 3: if $lg_b a < x$:*
  - *Y>=0 then $\Theta(n^x lg^y n)$*
  - *Y<0 then $\Theta(n^x)$*

Simplified steps (Case1: $n^{lg_b a}$ >f(n))

- $n^{lg_b a}$ $is\ watershed\ function$
- In every case we compare the watershed function with f(n)
- Or we compare the power of n in f(n) with $lg_b a$
- Case1: the watershed function $n^{lg_b a}$ a must be asymptotically larger than the driving function f(n) by at least a factor of $\Theta(n^{\epsilon})$ for some constant $\epsilon>0$. The master theorem then says that the solution is $T(n)= \Theta(n^{lg_b a})$.

Using the master method (Case1)

- T(n)=9.T (n/3)+n
- Lets identify a and b.
- a=9, b=3
- $watershed\ funtion\ is\ n^{lg_b a} = n^{lg_3 9} = O(n^2)$
- n^2>n
- T(n)= $\Theta(n^2)$

Simplified steps (Case2: $n^{lg_b a} \leq$f(n))

- The driving function grows similar or faster than the watershed function by a factor of Θ(lg^k n), where k>=0. The master theorem says that we tack on an extra lgn factor to f(n), yielding the solution T(n)= Θ($n^{lg_b a} lg^{k+1} n$).
- Most commonly occurs for k=0
- For simplicity this is for the case where the $lg_b a$ is equal to the power of n in f(n).

Using the master method (Case2)

- T(n)=T(2n/3)+1
- a=1, b=3/2
- $watershed\ funtion\ is\ n^{lg_b a} = n^{lg_{3/2} 1} = n^0 = 1.$
- f(n)=1= $n^{lg_b a}$
- T(n)=Θ($n^{lg_b a} lg^{0+1} n$)= Θ(lgn)

Simplified steps (Case3: $n^{lg_b a}$ <f(n)))

- f(n) must be asymptotically larger than the watershed function $n^{lg_b a}$ by at least a factor of $\Theta(n^{\epsilon})$ for some constant $\epsilon>0$.
- The Master's Theorem says that T(n)= Θ(f(n))
- Moreover, the driving function must satisfy the regularity condition that a.f(n/b)<=c.f(n).

Using the master method (Case3)

- T(n)=3T(n/4)+nlgn
- a=3, b=4
- $watershed\ funtion\ is\ n^{lg_b a} = n^{lg_4 3} = O(n^{0.793})$
- f(n)=nlgn
- T(n)= Θ(nlgn)

Exercise (identify the cases)

- $T(n)=2T(n/2)+n\lg n$
- $T(n)=2T(n/2)+\Theta(n)$
- $T(n)=8T(n/2)+\Theta(1)$
- $T(n)=7T(n/2)+\Theta(n^2)$

Exercise (identify the cases)

- $T(n)=2T(n/2)+n\lg n \rightarrow$ Case 2/case 3
- $T(n)=2T(n/2)+\Theta(n) \rightarrow$ Case 2
- $T(n)=8T(n/2)+\Theta(1) \rightarrow$ Case 1
- $T(n)=7T(n/2)+\Theta(n^2) \rightarrow$ Case 1

Justification

- T(n)=2T(n/2)+nlgn→Case 2
- a=2,b=2, and $n^{lg_2 2} = n$
- f(n)>watershed function by a factor lgn

Justification

- $T(n)=2T(n/2)+\Theta(n) \rightarrow$ Case 2
- $f(n)=\Theta(n)$
- Watershed function evaluates to n
- Similar growth

Justification

- $T(n)=8T(n/2)+\Theta(1)$
- $a=8$, $b=2$, $f(n)=\Theta(1)$ and watershed function evaluates to= n^3
- $F(n)<$ watershed function

Justification

- $T(n)=7T(n/2)+\Theta(n^2) \rightarrow$ Case 1
- a=7, b=2, $f(n)=\Theta(n^2)$, $watershed\ funtion\ n^{lg_b a}$=$n^{2.807}$

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

- Case 3
- a=1, b=2, f(n)= 2^n watershed function evaluates to $n^0 = 1$
- Solution: $\Theta(2^n)$