Analysis of Algorithms

Semester IV

Course Code
116U40C403
January – April 2024

SOMAIYA VIDYAVIHAR UNIVERSITY
K J Somaiya College of Engineering

Recurrence Relations

Recurrence Relation : Decreasing Functions

1. Algorithm A (n) .....T(n)

```
{   if (n>0)
    {
    print n; .....1
    }
    A(n-1) .......T(n-1)
    }
```

$$T(n) = \begin{cases} 1 & \ldots n=0 \\ T(n-1)+1 & \ldots n>0 \end{cases}$$

$$T(n-1) = T(n-2)+1$$

$$T(n-2) = T(n-3) +1$$

...

$$T(n) = T(n-2) + 1+1 \qquad = T(n-2) + 2$$

$$= T(n-3) + 1 + 2 \qquad = T(n-3)+3$$

$$T(n) = T(n-k) + k$$

when $n-k = 0$, $k = n$

$$T(n) = T(0) + n$$

$$= 1+n$$

$$= O(n)$$

Recurrence Relation

2. Algorithm A (n) .....T(n)

```
{   if (n>0)
    {
for ( i=0;i<n;i++)
    {
    print i; .....n
    }
    A(n-1) .......T(n-1)
    }
```

$T(n) = 1 \quad \ldots n=0$

$\quad = T(n-1)+n \quad \ldots n>0$

$T(n-1) = T(n-2)+(n-1)$

$T(n-2) = T(n-3) +(n-2)$

...

$T(n) = T(n-2) +(n-1)+n$

$\quad = T(n-3) + (n-2)+(n-1)+n$

$T(n) = T(n-k) + (n-k+1)+(n-k+2)+(n-k+3)\ldots+(n-1)+n$

when $n-k = 0$, $k = n$

$T(n) = T(0) + 1+2+3+4+\ldots.(n-1)+n$

$\quad = 1+n(n+1)/2$

$T(n) = O(n^2)$

Recurrence Relation

3. Algorithm A (n) .....T(n)

```
{   if (n>0)
    {
for ( i=0;i<n;i=i*2)
    {
    print i; .....log n
    }
    A(n-1) .......T(n-1)
    }
```

$T(n) = 1 \quad \ldots n=0$

$\quad = T(n-1)+\log n \quad \ldots n>0$

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

$T(n-2) = T(n-3) +\log (n-2)$

...

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

$\quad = T(n-3) + \log (n-2)+\log(n-1)+\log n$

$T(n) = T(n-k) + \log (n-k+1)+\log (n-k+2)+\ldots+\log (n-1$

when $n-k = 0$, $k = n$

$T(n) = T(0) + \log 1+\log 2+\log 3+\log 4+\ldots.\log (n-1)+$

$\quad = 1+ \log (1.2.3.4\ldots.(n-1).n)$

$\quad = 1+\log n!$

$T(n) = O(n \log n)$

Recurrence Relation

4. Algorithm A (n) .....T(n)

```
{   if (n>0)
    {
    print i; .....1
    A(n-1) .......T(n-1)
    A(n-1) .......T(n-1)
    }
}
```

$T(n) = 1 \quad \ldots n=0$

$= 2T(n-1)+1 \quad \ldots n>0$

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

$T(n-2) = 2\,T(n-3) + 1$

$T(n) = 2\,[2T(n-2)] + 1\,] + 1 = 2^2T(n-2) + 2 + 1$

$= 2^2[2T(n-3)+1]+2+1 = 2^3T(n-3) + 2^2 + 2 + 1$

$T(n) = 2^kT(n-k) + 2^{k-1} + 2^{k-2} + 2^{k-3} + \ldots + 2 + 1$

when $n-k = 0$, $k = n$

$T(n) = 2^n\,T(0) + 2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots + 2 + 1$

$= 2^n + 2^{n-1} + 2^{n-2} + 2^{n-3} + \ldots + 2 + 1$

$= 2^{n+1} - 1$

$T(n) = O(2^n)$

Master Theorem: Decreasing Functions

T(n) = a T (n-b) + f(n)

where,

n = size of input

a = number of subproblems in the recursion

n-b = size of each subproblem. All subproblems are assumed to have the same size.

f(n) = cost of the work done outside the recursive call, which includes the cost of dividing the problem and cost of merging the solutions

Here, a ≥ 0 and b > 0 are constants, and f(n) is an asymptotically positive function.

f(n) = Θ (n^k) where k>=0

Master Theorem: Decreasing Functions

$T(n) = a\,T(n-b) + f(n)$, $f(n) = \Theta(n^k)$ where $k>=0$

Case I if $a<1$, $b = 1$ $T(n) = \Theta(n^k)$ ...same as $f(n)$

Case II if $a = 1$, $b = 1$ $T(n) = O(n^{k+1})$...same as $O(n * f(n))$

Case III if $a > 1$, $b = 1$ $T(n) = O(a^n * n^k)$...same as $O(a^n * f(n))$

Case IV if $a > 1$, $b > 1$ $T(n) = O(a^{n/b} * n^k)$...same as $O(a^{n/b} * f(n))$

Master Theorem: Examples

$T(n) = a\,T(n-b) + f(n)$, $f(n) = \Theta(n^k)$ where $k>=0$

1. $T(n) = T(n-1) + 1$ $O(n)$ a=1 , b=1 ...Case II $O(n^* f(n))$
2. $T(n) = T(n-1) + n$ $O(n^2)$ a=1 , b=1 ...Case II $O(n^* f(n))$
3. $T(n) = T(n-1) + \log n$ $O(n\log n)$ a=1 , b=1 ...Case II $O(n^* f(n))$
4. $T(n) = 2\,T(n-1) + 1$ $O(2^n)$ a= 2, b=1...Case III $O(a^n * f(n))$
5. $T(n) = 2\,T(n-1) + n$ $O(n * 2^n)$ a= 2, b=1...Case III $O(a^n * f(n))$

Recurrence Relation : Dividing Functions

1. Algorithm A (n) …..T(n)

```
{   if (n>0)
    {
    print i; …..1
    A(n/2) …….T(n/2)
    }
}
```

$T(n) = 1 \quad \ldots n=0$

$\quad = T(n/2)+1 \quad \ldots n>0$

$T(n/2) = T(n/2^2) + 1$

$T(n/2^2) = T(n/2^3) + 1$

$T(n) = T(n/2^2)+1+1 = T(n/2^2)+2$

$\quad = T(n/2^3)+1+2 = T(n/2^3)+3$

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

when $n/k = 1$, $2^k = n \quad k = \log_2 n$

$T(n) = T(1) + \log_2 n$

$\quad = 1 + \log_2 n$

$T(n) = O(\log_2 n)$

Recurrence Relation : Dividing Functions

2. Algorithm A (n) .....T(n)

```
{   if (n>0)
    {
for (i=0;i<n;i++)
{
    print i; .....n
}
    A(n/2) .......T(n/2)
    }
}
```

$T(n) = 1 \quad \ldots n=0$

$= T(n/2)+n \quad \ldots n>0$

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

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

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

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

$T(n) = T(n/2^k)+n [(1/2^{k-1})+(1/2^{k-2})+\ldots\ldots(1/2)+1]$

$=T(n/2^k)+n [1+1]$

when $n/k = 1$, $2^k = n \quad k = \log_2 n$

$T(n) = T(1) + 2n$

$= 1 + 2 n$

$T(n) = O(n)$

Recurrence Relation : Dividing Functions

3. Algorithm A (n) .....T(n)

```
{   if (n>0)
    {
for (i=0;i<n;i++)
{
    print i; .....n
}
    A(n/2) .......T(n/2)
    A(n/2) .......T(n/2)
    }
}
```

$T(n) = 1 \quad \ldots n=0$

$\quad = 2T(n/2)+n \quad \ldots n>0$

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

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

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

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

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

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

$T(n) = 2^k\,T(n/2^k) + kn$

when $n/k = 1$, $2^k = n \quad k = \log_2 n$

$T(n) = n\,T(1) + \quad n \log_2 n$

$\quad = n + n\log_2 n$

$T(n) = O(n\log_2 n)$

Master Theorem: Dividing Functions

$T(n)$ = $a\,T(n/b) + f(n)$ where,
n = size of input
a = number of subproblems in the recursion
n/b = size of each subproblem. All subproblems are assumed to have the same size.
f(n) = cost of the work done outside the recursive call, which includes the cost of dividing the problem and cost of merging the solutions
Here, $a \geq 1$ and $b > 1$ are constants, and f(n) is an asymptotically positive function.

$f(n) = \Theta(n^k \log_p n)$

Master Theorem: Dividing Functions

$T(n) = a\,T(n/b) + f(n)$, $f(n) = \Theta(n^k \log^p n)$ where $k>=0$

Case I if $\log_b a > k$ $T(n) = \Theta(n^{\log_b a})$

Case II if $\log_b a = k$

a) if $p>-1$ $T(n) = \Theta(n^k * \log^{p+1} n)$
b) if $p = -1$ $T(n) = \Theta(n^k * \log \log n)$
c) if $p<-1$ $T(n) = \Theta(n^k)$

Case III if $\log_b a < k$

a) if $p>= 0$ $T(n) = T(n) = \Theta(n^k * \log^p n)$
b) if $p <0$ $T(n) = O(n^k)$

Master Theorem: Examples

Recurrence Function	Case	Formula	Answer
$T(n) = 2\,T(n/2)+1$	a= 2, b= 2, k=0, p=0, Case I $\log_b a > k$	$T(n) = \Theta(n^{\log_b a})$	$T(n) = \Theta(n)$
$T(n) = 4\,T(n/2)+n$	a= 4, b= 2, k=1, p=0, Case I $\log_b a > k$	$T(n) = \Theta(n^{\log_b a})$	$T(n) = \Theta(n^2)$
$T(n) = 8\,T(n/2)+n^2$	a= 8, b= 2, k=2, p=0, Case I $\log_b a > k$	$T(n) = \Theta(n^{\log_b a})$	$T(n) = \Theta(n^3)$
$T(n) = 9\,T(n/3)+n$	a= 9, b=3, k=1, p=0, Case I $\log_b a > k$	$T(n) = \Theta(n^{\log_b a})$	$T(n) = \Theta(n^2)$
$T(n) = 9\,T(n/3)+n^2$	a= 9, b=3, k=2, p=0, Case II $\log_b a = k$, p>-1	$T(n) = \Theta(n^k * \log^{p+1} n)$	$T(n) = \Theta(n^2 \log n)$
$T(n) = 8\,T(n/2)+n \log n$	a= 8, b= 2, k=1, p=1, Case I $\log_b a > k$	$T(n) = \Theta(n^{\log_b a})$	$T(n) = \Theta(n^3)$
$T(n) = 2\,T(n/2)+n$	a= 2, b= 2, k=1, p=0, Case II $\log_b a = k$, p>-1	$T(n) = \Theta(n^k * \log^{p+1} n)$	$T(n) = \Theta(n\log n)$
$T(n) = 2\,T(n/2)+n / \log n$	a= 2, b= 2, k=1, p=-1, Case II $\log_b a = k$, p>-1	$T(n) = \Theta(n^k * \log\log n)$	$T(n) = \Theta(n\log\log n)$

Master Theorem: Examples

Case	Recurrence Function	Answer
Case I : $\log_b a > k$ formula $T(n) = \Theta(n^{\log_b a})$	$T(n) = 2\,T(n/2)+1$	$T(n) = \Theta(n)$
	$T(n) =4\,T(n/2)+1$	$T(n) = \Theta(n^2)$
	$T(n) =4\,T(n/2)+n$	$T(n) = \Theta(n^2)$
	$T(n) = 8\,T(n/2)+n^2$	$T(n) = \Theta(n^3)$
	$T(n) = 16\,T(n/2)+n$	$T(n) = \Theta(n^4)$
Case III $\log_b a < k$ $p >= 0$, $T(n) = \Theta(n^k \log^p n)$	$T(n) = T(n/2)+n$	$T(n) = \Theta(n)$
	$T(n) = 2T(n/2)+n^2$	$T(n) = \Theta(n^2)$
	$T(n) = 2\,T(n/2)+n^2 \log n$	$T(n) = \Theta(n^2 \log n)$
$p < 0$, $T(n) = \Theta(n^k)$	$T(n) = 2\,T(n/2)+n^2 / \log n$	$T(n) = \Theta(n^2)$

Master Theorem: Examples

Case II : $\log_b a = k$	Recurrence Function	Answer
formula $p > -1$ $T(n) = \Theta(n^k * \log^{p+1} n)$	$T(n) = T(n/2)+1$	$T(n) = \Theta(\log n)$
	$T(n) = 2T(n/2)+n$	$T(n) = \Theta(n \log n)$
	$T(n) = 2T(n/2)+n \log n$	$T(n) = \Theta(n \log^2 n)$
	$T(n) = 4T(n/2)+n^2$	$T(n) = \Theta(n^2 \log n)$
	$T(n) = 4T(n/2)+(n \log n)^2$	$T(n) = \Theta(n^2 \log^3 n)$
$p = -1$ $T(n) = \Theta(n^k * \log \log n)$ $p < -1$, $T(n) = \Theta(n^k)$	$T(n) = 2T(n/2)+n/\log n$	$T(n) = \Theta(n \log \log n)$
	$T(n) = 2T(n/2)+n/\log^2 n$	$T(n) = \Theta(n)$

Thank you