

TIME COMPLEXITY OF A RECURRSIVE FUNCTION

MASTER THEOREM FOR

Master Theorem

Gives the Time Complexity for the recurrence relation:
 $T(n) = aT(n/b) + f(n)$



Master Theorem

For the Recurrence: $T(n) = aT(n/b) + \Theta(n^c)$, $a \geq 1$, $b > 1$

There are following three cases:

1. If $f(n) = \Theta(n^c)$ where $c < \log_b a$ then $T(n) = \Theta(n^{\log_b a})$
2. If $f(n) = \Theta(n^c)$ where $c = \log_b a$ then $T(n) = \Theta(n^c \log n)$
3. If $f(n) = \Theta(n^c)$ where $c > \log_b a$ then $T(n) = \Theta(f(n))$



Problems:

1. $T(n) = 2 T(n/2) + \Theta(n)$

$a = 2, b = 2, c = 1$
 $\rightarrow c = \log_b a$

Time Complexity: $\Theta(n \log_2 n)$



Time Complexity using Masters Theorem | C++ Placement Course | Lecture 16.5

Press **Esc** to exit full screen



Problems:

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

$a = 3, b = 2, c = 2$
 $\rightarrow c > \log_b a$

Time Complexity: $\Theta(n^2)$



4:18 / 6:54



Recurrence Tree Method:

$$1. \quad T(n) = T(n-1) + n$$

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

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

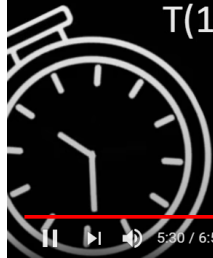
$$T(n-2) = T(n-3) + n-2$$

.

.

$$T(1) = 1$$

Adding all the terms, we get



5:30 / 6:54

Recurrence Tree Method:

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

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

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



Recurrence:

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

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

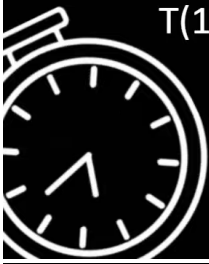
$$T(n/2) = 2T(n/4) + n/2$$

$$T(n/4) = 2T(n/8) + n/4$$

.

.

$$T(1) = 1$$



Quick Sort Complexity:

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

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

$$T(n/2) = 2T(n/4) + n/2$$

$$T(n/4) = 2T(n/8) + n/4$$

.

.

$$T(1) = 1$$

x1

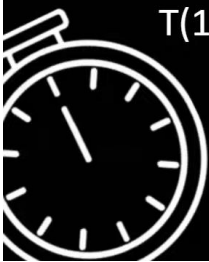
x2

x4

.

.

$x2^{\log n}$



Quick Sort Complexity:

$T(n) = n + n + n + \dots$ Log n terms
 $= \Theta(n \log n)$ in best case



13:58 / 15:23

