FALL 2019

CSE 5311-004

DESIGN & ANALYSIS OF ALGORITHMS

WRITTEN ASSIGNMENT - 2

GOUTAMI PADMANABHAN
1001669338

EXERCISE 4.3-1

Show that the solution of $T(n) = T(n-1) + n$ is $O(n^2)$

Answer :

We can use substitution method to prove this

The substitution method for solving recurrences.

comprises of two steps

1. Guess the form of the solution
2. Use mathematical induction to find the constants and show that the solution works.

* Lets use the substitution method to establish upper bounds on the recurrence $T(n)$ (i.e) $T(n) = O(n^2)$

* But we already know that if $f(n) = O(g(n))$ then, there exists positive constance c_1 and n_0 such that

$$f(n) \leq c_1 g(n) \quad \text{for all } n \geq n_0 \longrightarrow (1)$$

Here, $f(n) = T(n) = T(n-1) + n$

$g(n) = n^2$

For us to prove that $T(n) = O(n^2)$, let us assume

$T(n) \leq cn^2 \; \forall n \geq 1 \Rightarrow$ From (1)

If $T(n) \leq cn^2 \longrightarrow (2)$

then $T(n-1) \leq c(n-1)^2$

$$T(n-1) \leq c(n^2-2n+1)$$

$$T(n-1) \leq cn^2 - 2cn + c \longrightarrow (3)$$

But we are given that

$$T(n) = T(n-1) + n \longrightarrow (4)$$

Substituting (3) in (4)

$$T(n) = cn^2 - 2cn + c + n \longrightarrow (5)$$

By equating (2) and (5) we get

$$cn^2 - 2cn + c + n \leq \cancel{c}n^2$$

$$-2cn + c + n \leq 0 \quad \forall\, n \geq 1 \text{ and } c > 0$$

Let us consider $n = 1$ and compute

$$-2c + c + 1 \leq 0$$

$$1 - c \leq 0$$

$$\boxed{1 \leq c}$$

Here $n = 1$ and $c \geq 1$

* Let's use mathematical induction to find constant and show that the solution works

Base case:

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

LET $n = 1$, $T(1) = 1 \longrightarrow (A)$

Inductive Step:

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

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

$T(2) = 1 + 2 \Rightarrow$ From (A)

$T(2) = 3 \longrightarrow (B)$

LET $n = 3$, $T(3) = T(3-1) + 3$

$= T(2) + 3$

$T(3) = 1 + 2 + 3 \Rightarrow$ From (B) $\longrightarrow (C)$

LET $n = 4$, $T(4) = T(4-1) + 4$

$= T(3) + 4$

$= 1 + 2 + 3 + 4 \Rightarrow$ From (C) $\longrightarrow (D)$

and so on.

From (A), (B), (C) and (D) we see that we have a series

$T(n) = 1 + 2 + 3 + 4 + \cdots n$

$$T(n) = \frac{n(n+1)}{2} \longrightarrow (E)$$

But we assume that

$$T(n) \leq cn^2 \quad \text{to prove} \quad T(n) = O(n^2)$$

$\longrightarrow (F)$

Substituting (E) in (F)

$$\frac{n(n+1)}{2} \leq cn^2$$

Hence we prove that .

$T(n) = T(n-1) + n$ is $O(n^2)$

EXERCISE 4.3-2

Show that the solution of $T(n) = T(\lceil \frac{n}{2} \rceil) + 1$ is $O(\lg n)$

ANSWER:

Method 1:

By using Substitution method.

We guess the solution is $T(n) = O(\lg n)$

The substitution method requires us to prove

that $T(n) \leq c \lg n \quad \forall c > 0$.

We start by assuming that this bound holds

for all positive $m < n$, in particular for $m = \lceil \frac{n}{2} \rceil$

yeilding

$$T(\lceil \frac{n}{2} \rceil) \leq c \lg (\lceil \frac{n}{2} \rceil)$$

Substituting this into recurrence yields

$$T(n) \leq c \lg (\lceil \tfrac{n}{2} \rceil) + 1$$

$$T(n) \leq c \lg (\tfrac{n}{2}) + 1$$

$$= c \lg n - c \lg 2 + 1$$

$$= c \lg n - c + 1$$

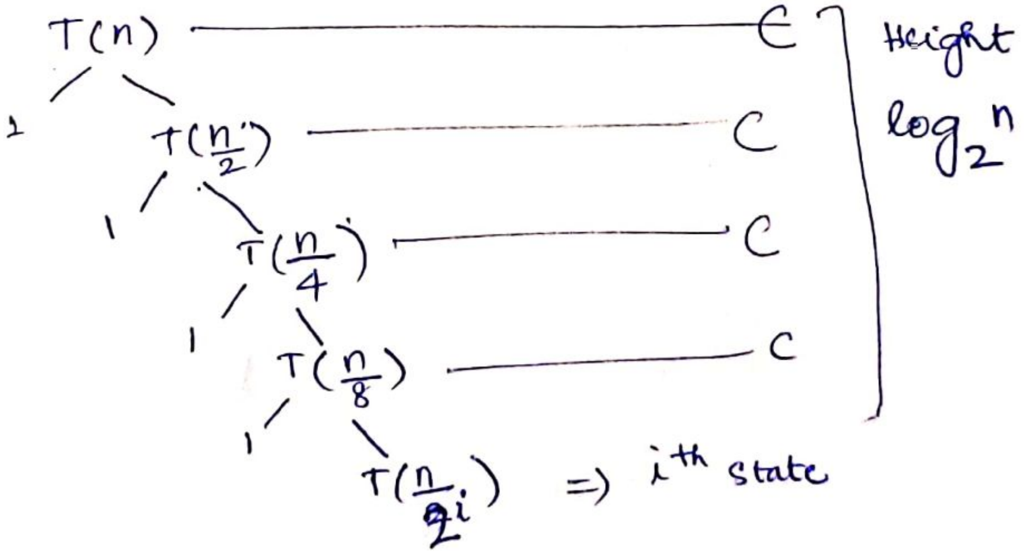
$$T(n) \leq c \lg n \quad \forall c > 0$$

Method 2: Recursion tree

1) Each node represents the cost of a sub-problem in the set of call to recursive functions
2) we sum costs per level and determine the total cost of all level of recursion
3) Useful when the recurrence describes execution time of a divide and conquer algorithm

Here we will create recursion tree from .

$T(n) = T(\lceil \frac{n}{2} \rceil) + 1$



$\frac{n}{2^i}$ where i is the level $i = \log_2 n \longrightarrow (1)$

Here we added the co-efficient $C > 0$.

$a = 1,\ b = 1,\ f(n) = 1 \qquad \log_2 1 = 0$

$n^{\log_2 1} = n^0 = 1 \Rightarrow$ Total $c \log_2 n$

No. of nodes at depth $= i$

Cost of all nodes at depth $i = (\frac{1}{2})^i n = \frac{n}{2^i} = T(1) = 1$

$n = 2^i$

Total cost $= C \log_2 n + 2(1) \simeq C \log_2 n \Rightarrow \boxed{O(\log n)}$

EXERCISE 4.5-1

Use the master method to give tight asymptotic bounds for the following recurrences

~~1) T(N) = 2T(n/4) + 1~~

ANSWER :

LET $a \geq 1$ and $b > 1$, let $f(n)$ be a function and let $T(n)$ be defined on the nonnegative integers by the recurrence: $T(n) = aT\left(\frac{n}{b}\right) + f(n)$,

where we interpret $\frac{n}{b}$ to mean either $\left\lfloor \frac{n}{b} \right\rfloor$ or $\left\lceil \frac{n}{b} \right\rceil$

Then $T(n)$ has the following asymptotic bounds

CASE 1:
a) If $f(n) = O\left(n^{\log_b a - \varepsilon}\right)$ for some constant $\varepsilon > 0$ then $T(n) = \Theta\left(n^{\log_b a}\right)$

CASE 2:
b) If $f(n) = \Theta\left(n^{\log_b a}\right)$, then $T(n) = \Theta\left(n^{\log_b a} \lg n\right)$

CASE 3:
c) If $f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right)$, for some constant $\varepsilon > 0$ and if $af\left(\frac{n}{b}\right) \leq cf(n)$ for some constant $c > 1$ and all sufficiently large n, then $T(n) = \Theta(f(n))$

1) $T(n) = 2T\left(\frac{n}{4}\right) + 1$

where $a = 2$, $b = 4$, $f(n) = 1$

$n^{\log_b a} = n^{\log_4 2}$
$= n^{1/2}$
$= \sqrt{n}$

Since $f(n) = 1$, 1 is constant. we can say

$\Theta(n^{1/2 - \varepsilon})$. Here $\varepsilon = \frac{1}{2}$

Hence, applying CASE 1

$T(n) = \Theta(n^{\log_a b}) = \Theta(n^{1/2}) = \Theta(\sqrt{n})$

2) $T(n) = 2T\left(\frac{n}{4}\right) + \sqrt{n}$

where $a = 2$, $b = 4$, $f(n) = \sqrt{n} = n^{1/2}$

$n^{\log_b a} = n^{\log_4 2} = n^{1/2}$

Since $f(n) = n^{1/2}$, we can apply CASE 2

$\sqrt{n} = \Theta(n^{1/2}) = \Theta(\sqrt{n})$

$T(n) = \Theta(n^{\log_b a} \lg n) = \Theta(n^{1/2} \lg n)$

$\boxed{T(n) = \Theta(\sqrt{n} \lg n)}$

3) $T(n) = 2T\left(\frac{n}{4}\right) + n$

Here $a = 2$, $b = 4$, $f(n) = n$

$n^{\log_b a} = n^{\log_4 2} = n^{1/2} = \sqrt{n}$

$n^{\log_b a} = n^{1/2}$

But $f(n) = n$

$f(n) = n^{\log_b a + \varepsilon}$

$n = n^{1/2 + \varepsilon}$ Here $\varepsilon = \frac{1}{2}$

Hence, applying CASE 3

$f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) = \Omega\left(n^{1/2 + \varepsilon}\right)$

$a f\left(\frac{n}{b}\right) \le c f(n)$

$2 \cdot \frac{n}{4} \le 3n$ we assume $c = 3 > 1$

$\frac{n}{2} \le 3n$ holds true

By applying case 3:

$T(n) = \Theta(f(n))$

$\boxed{T(n) = \Theta(n)}$

4) $T(n) = 2T\left(\frac{n}{4}\right) + n^2$

Here,

$a = 2, \quad b = 4, \quad f(n) = n^2$

$n^{\log_b a} = n^{\log_4 2} = n^{1/2} = \sqrt{n}$

$n^{\log_b a} = n^{1/2}$

But $f(n) = n^2$

$f(n) = n^{\log_b a + \varepsilon}$

$n^2 = n^{1/2 + 1.5}$ Here $\varepsilon = 1.5$

and $a f\left(\frac{n}{b}\right) \leq c f(n)$

$2\left(\frac{n}{4}\right)^2 \leq 2n^2$ Let's assume $c = 2 > 1$

$2 \times \frac{n^2}{16} \leq 2n^2$

$\frac{n^2}{8} \leq 2n^2 \Rightarrow \frac{1}{8} \leq 2$ Holds true

Hence, we apply case 3:

$T(n) = \Theta(f(n))$

$\boxed{T(n) = \Theta(n^2)}$