Algorithm Homework

Tai Jiang

October 2023

Contents

1 Show that the solution of $T(n) = T(\lceil n/2 \rceil) + 1$ is $O(\lg n)$. **1**

2 Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 4T(n/2 + 2) + n$. Use the substitution method to verify your answer. **2**

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

Origin :

Exercise 4.3-2

Page 87

Answer :

That can build a recursion tree to visualize how this recurrence works:

1. Start with $T(n)$ at the top.
2. At each level of the tree, we have $T(\lceil n/2 \rceil) + 1$.
3. The tree branches into two subproblems, one with size $\lceil n/2 \rceil$ and another with size $\lceil n/2 \rceil$.

The tree might look like Figure 1.

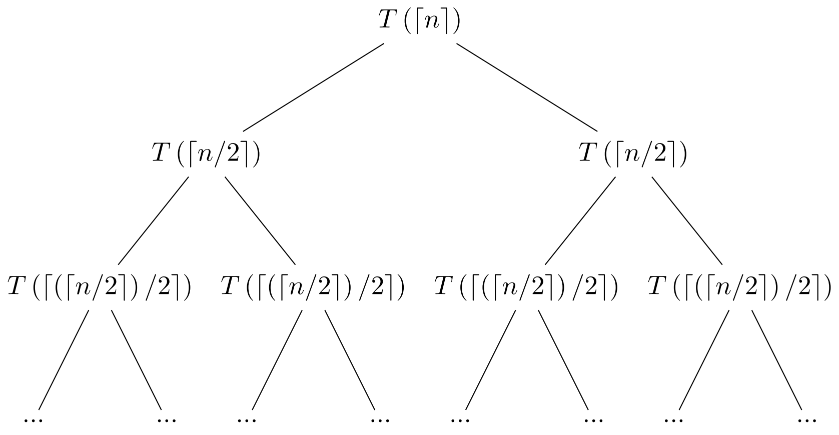
$T(\lceil n \rceil)$
$T(\lceil n/2 \rceil)$
$T(\lceil n/2 \rceil)$
$T(\lceil(\lceil n/2 \rceil)/2\rceil)$
$T(\lceil(\lceil n/2 \rceil)/2\rceil)$
$T(\lceil(\lceil n/2 \rceil)/2\rceil)$
$T(\lceil(\lceil n/2 \rceil)/2\rceil)$
...

Figure 1: Recursion Tree.

At each level, the size of the problem is divided by 2, and we keep going until the size becomes 1 (or less, in which case we stop). The depth of the tree will be the number of times we can halve n until it becomes 1. So it's the number of times we can take $\lceil n/2 \rceil$ until $\lceil n/2 \rceil \leq 1$.

At each step, we're taking the ceiling of half of the previous value, which is effectively dividing it by 2. We continue this process until the value is less than or equal to 1. So, let k be the number of steps it takes for $\lceil n/2 \rceil$ to become 1, look like equation (1).

$$\frac{\lceil n/2 \rceil}{2^k} \leq 1 \tag{1}$$

Then sovle for k, look like equation (2):

$$\begin{gathered} \frac{n}{2^k} \leq 1 \\ n \leq 2^k \\ \lg n \leq k \end{gathered} \tag{2}$$

The depth of the recursion tree is $O(\lg n)$, which means the algorithm has a time complexity of $O(\lg n)$. So the solution of $T(n) = T(\lceil n/2 \rceil) + 1$ is indeed $O(\lg n)$.

2 Use a recursion tree to determine a good asymptotic upper bound on the recurrence $T(n) = 4T(n/2 + 2) + n$. Use the substitution method to verify your answer.

Origin :

Exercise 4.4-3

Page 93

Answer :

That can build a recursion tree to visualize how this recurrence works:

1. Start with $T(n)$ at the top.
2. At each level of the tree, we have 4 subproblems of size $T(n/2 + 2) + n$.
3. The cost of each level is n.

The tree might look like Figure 2.

At each level, we have 4 subproblems of size T(n/2 + 2), and each subproblem incurs a cost of n. The number of levels in the tree will depend on how quickly the subproblem size decreases.

The subproblem size is equation (3):

$$T(n) = 4T(n/2 + 2) + n \tag{3}$$

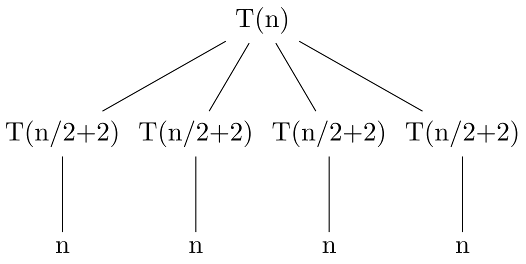

Figure 2: Recursion Tree.

The subproblem size is n/2 + 2, so we calculate the subproblem size for the next level by equation (4):

$$\begin{aligned} T(n/2+2) &= 4T((n/2+2)/2+2) + n/2+2 \\ &= 4T(n/4+1+2) + n/2+2 \\ &= 4T(n/4+3) + n/2+2 \end{aligned} \tag{4}$$

At each level, we are adding 2 to the subproblem size. Therefore, at level k, the subproblem size will be $n/(2^k) + 2k$.

Now, we want to find the level where the subproblem size becomes a constant, $n/(2^k) + 2k = C for some constant C$, calculate the k look like equation (5).

$$\begin{aligned} n/(2^k) + 2k &= C \\ n/(2^k) &= C - 2k \\ 2^k &= n/(C-2k) \\ k &= \lg(n/(C-2k)) \end{aligned} \tag{5}$$

C is a constant. The number of levels in the recursion tree is $O(\lg n)$.

The cost at each level in the recursion tree is equation (6):

$$\begin{aligned} &\text{At level 0:} && n \\ &\text{At level 1:} && 4*n/2 = 2n \\ &\text{At level 2:} && 4*(n/4) = n \\ &\quad ... \\ &\text{At level k:} && (4^k)*(n/(2^k)) = (n/2^k)*(4^k) = C*4^k \end{aligned} \tag{6}$$

Summing up the costs of all levels(Equation (7)):

$$T(n) = n + 2n + 4n + ... + C*4^k \tag{7}$$

This is a geometric series, and its sum can be bounded by(Equation (8)):

$$T(n) \leq n*(1-4^(k+1))/(1-4) \tag{8}$$

Since $k = O(\lg(n))$, $4^{(}k + 1)$ is polynomial in n. Therefore, T(n) is O(n).
verify this result using the substitution method:
Assume that $T(m) \leq km - p$ for some positive constants k and p, where m ¡ n(Equation (9)).

$$\begin{aligned} T(n) &= 4T(n/2 + 2) + n \\ &\leq 4(k(n/2 + 2) - p) + n \\ &= 2kn - 4k + 4n - 4p + n \\ &= (2k + 5)n - 4k - 4p \end{aligned} \tag{9}$$

Find k and p such that $(2k + 5)n - 4k - 4p \leq kn - p$.
This holds if $2k + 5 \leq kand - 4k - 4p \leq -p$.
Solving these inequalities(Equation (10)):

$$\begin{aligned} 2k + 5 &\leq k \\ k &\leq -5 \\ -4k - 4p &\leq -p \\ -4k &\leq 0 \\ k &\geq 0 \end{aligned} \tag{10}$$

Since we can't find k and p that satisfy these inequalities, our initial assumption that $T(m) \leq km - p$ for m ¡ n is incorrect.
Therefore, T(n) is not $O(n^k)$ for any positive constant k. Instead, as shown earlier, T(n) is O(n).

```
#!/usr/bin/perl
print S(@ARGV);sub S{$r=(@_[0]%4==0&&@_[0]%100!=0)||@_[0]%400=0;}

```