Renju Radhakrishnan

4-3 **More recurrence examples**

Give asymptotic upper and lower bounds for T(n) in each of the following recurrences. Assume that T(n) is constant for sufficiently small n. Make your bounds as tight as possible, and justify your answers.

$T(n) = 4T(n/3) + n \lg n$ (a).

$n^{\log_3 4}$ $f(n) = n \lg n$

$n^{1.2618}$

or

$\boxed{\Theta(n^{\log_3 4})}$ master method

Renju Radhakrishnan

b. $T(n) = 3T(n/3) + n\lg n$

$T(n/3) = 3(3T(n/9) + (n/3)\lg(n/3)) + n\lg n$

$= 9T(n/9) + 3(n/3)\lg(n/3) + n\lg n$

$3^x T(n/3^x) + \sum_{j=1}^{i-1} n/\lg(n/3^{j-1})$

$T(n) = n\Theta(1) + \sum_{j=1}^{\log_3 n - 1} n/\lg(n/3^{j-1})$

$= \Theta(n) + n\log_3 2 \sum_{i=2}^{\log_3 n} (1/i)$

$\rightarrow \Theta(n\lg(\lg n))$

Renju Radhakrishnan

c. $T(n) = 4T(n/2) + n^2\sqrt{n}$

$+ n^{2.5}$

$\Rightarrow n^{\log_2 4}$

$\Rightarrow n^2$

$n^{2+0.5} \Rightarrow n^{2.5}$

$\Theta(f(n))$

$$\boxed{\Theta(n^2\sqrt{n})}$$

Renju Radhakrishnan

d. $T(n) = 3T(n/3 - 2) + n/2$

$\Rightarrow T(n) = 3T(n/3) + n/2$

$\Rightarrow T(n) = 3T(n/3) + \Theta(n)$

$\Rightarrow T(n) = \Theta(n \lg n)$

Renju Radhakrishnan

e. $T(n) = 2T(n/2) + n/\lg n$

Same steps as a.

$\Theta(n \lg(\lg(n)))$

Renju Radhakrishnan

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

guessing $\Theta(n)$

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

$\leq (5/8)cn + n \leq cn \quad (c \geq 8)$

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

$\geq (5/8)cn + n \geq cn \quad (c \leq 8)$

$\Theta(n)$

Renju Radhakrishnan

9. $T(n) = T(n-1) + 1/n$

$$T(n-1) = T(n-2) + \frac{1}{n-1} + 1/n$$

$$T(n-2) = T(n-4) + \frac{1}{n-3} + \frac{1}{n-2} + \frac{1}{n}$$

$$\sum_{i=1}^{n} \frac{1}{i} = \Theta(\lg n)$$

Renju Radhakrishnan

h. $T(n) = T(n-1) + \lg n$

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

$T(n-2) = T(n-4) + \lg(n-4) + \lg(n-2)$

$+ \lg(n)$

$T(n-4) = T(n-8) + \lg(n-8) + \lg(n-8)$

$+ \lg(n-4) \quad + \lg(n) \ldots$

$$\sum_{i=0}^{n-1} \lg(n-i) = \sum_{i=1}^{n} \lg(i)$$

$= \lg(n!) \leq \log n^n = n \lg n$

$\Theta(n \lg n)$

Renju Radhakrishnan

i. $T(n) = T(n-2) + 1/\lg n$

$$T(n-2) = T(n-4) + \frac{1}{\lg(n-2)} + \frac{1}{\lg n}$$

$$T(n-4) = T(n-8) + \frac{1}{\lg(n-6)} + \frac{1}{\lg(n-4)}$$

$$+ \ 1/\lg n$$

$$\sum_{i=1}^{n/2} \frac{1}{\lg(2i)} \Rightarrow \boxed{\Theta(\lg(\lg(n)))}$$

Renju Radhakrishnan

j. $T(n) = \sqrt{n}\,T(\sqrt{n}) + n$

$T(n) \leq cn\lg(\lg(n))$ ← guessing

$T(n) \leq \sqrt{n}\, c\, n\lg(\lg n) + n$

$cn\lg\frac{\lg n}{2} + n$

$cn\lg\lg n - cn\lg 2 + n$

$cn\lg\lg n + (1-c)n$

where $(c > 1)$

$\leq cn\lg\lg n$

$$\boxed{\Theta(n\lg(\lg(n)))}$$

Renju Radhakrishnan

You are given a set S of n integers, as well as one more integer v. Design an algorithm that determines whether or not there exist two distinct elements x, y ∈ S such that x + y = v. Your algorithm should run in time O(n log n), and it should return (x, y) if such elements exist and (NIL,NIL) otherwise. Prove the worst case running time bound and the correctness of the algorithm.

2.

```
S = MergeSort( S of n integers)
for n in S from 0 to S.length - 1
    X[n] = v - S[n]
end
for n in S from 0 to S.length - 1
    if [ X[X.length - n] == S[n])
        if listofValues > 2 : break, else
            AddTOLIST OFVALUES ( S[N])

end    end    end
return (listofValues)

    O(nlogn)
```

Renju Radhakrishnan

Since we know Mergesorts runtime is $O(n\log n)$ from the previous proof, we can move on to the next line. The next for loop iterates from 0 to N inside the set S where N is 0 to S.length−1, thus the runtime is $O(N)$ for the for loop. The next for loop does the same, thus $\Theta(N)$. The return is constant at $O(1)$. This means that

$$O(n\log n) + O(n) + O(n) + O(1)$$

$= O(n\log n)$ as the worst case runtime.

Loop invariant

At the start of each iteration of the for loop, each value from $S[p..k]$ contains integers.

Initialization

Prior to the first iteration, we have a full set of integers S.

Maintenance:

For each iteration, if values between ~~x~~ and S are the same, we add it to the list of return values. Only two values will exist since the loop terminates once we get those values.

Termination

At termination, there is only two possible values (x, y) or (NIL, NIL),

Prove tight **worst-case** asymptotic upper bounds for the following recurrence equations that satisfy T(n) = 1 for n<=2, and depend on a variable q E [0, n/4].

Renju Radhakrishnan

(a) $T(n) = T(n-2q-1) + T(3q/2) + T(q/2) + \Theta(1)$

(b) $T(n) = T(n-q-1) + T(n/2-q) + \Theta(n)$

(c) $T(n) = T(n-q-1) + T(3q) + \Theta(n)$

$T(n) = T(n-2q-1) + T(3q/2) + T(q/2) + \Theta(1)$ $T(n) > 2$

$T(n) = 1$ when $T(n) \leq 2$

Depends on $q \in [0, n/4]$

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

$= T(n-4q-2) + 2T(3q/2) + 2T(q/2) + \Theta(1)$

$T(n-4q-2) = T(n-8q-4) + 3T(3q/2) + 3T(q/3) + \Theta(1)$

$\Downarrow$

$$\sum_{i=1}^{2n} i \cdot x$$

$O(n)$ runtime

Renju Radhakrishnan

(b) $T(n) = T(n-q-1) + T(n/2 - q) + \Theta(n)$

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

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

$+ T(n/2 - q)$

$T(n-2q-2) = T(n-4q-4) + T(n/2 - q/2 - 1/2) + \Theta(n)$

$+ T(n/2 + q/2 - 1/2) + T(n/2 - q)$

$+ \Theta(n)$

...

$O(n \log n)$

Renju Radhakrishnan

(c.) $T(n) = T(n-q-1) + T(3q) + \Theta(n)$

$\Rightarrow \quad T(n-1) + T(0) + \Theta(n)$

$\Rightarrow \quad T(n-1) + \Theta(n)$

$\Rightarrow \quad O(n)$ worst case

Renju Radhakrishnan

4. (10 pts) Consider the following silly randomized variant of binary search. You are given a sorted array A of n integers and the integer v that you are searching for is chosen uniformly at random from A. Then, instead of comparing v with the value in the middle of the array, the randomized binary search variant chooses a random number r from 1 to n and it compares v with A[r]. Depending on whether v is larger or smaller, this process is repeated recursively on the left sub-array or the right sub-array, until the location of v is found. Prove a tight bound on the expected running time of this algorithm.

Given a sorted array A, the expected runtime of the silly randomized variant of quicksort is O(logn).

We can state this as we know that on the first occurrence of the algorithm, we have a 1/n of finding the v.

On the second 2/n

Third 4/n

and so on and so forth when dealing with randomization.

we can derive $\frac{2^{x-1}}{n}$, thus the average number of recursive calls is:

$$\frac{1}{n}\sum_{i=1}^{\log(n)} i2^{i-1} \Rightarrow i2^{i-1} < \log(n)\cdot 2^{i-1}$$

$$\Rightarrow \frac{1-2^{\log(n)}}{1-2} = 2^{\log n}-1 = n-1$$

$$\frac{\log(n)}{n}\cdot(n-1) \Rightarrow \log(n)$$

average case run time is log(n)

worst case is O(n) if it has v as the A[0] position and A[length-1] as the random choice for comparison on every iteration, but that's highly unlikely

Renju Radhakrishnan

5. (10 pts) Can the master method be applied to the recurrence T(n) = 4T(n/2)+n2 log n? Why or why not? Give an asymptotic upper bound for this recurrence

5. $T(n) = 4T(n/2) + n^2 \lg n$

$a=4 \quad b=2 \Rightarrow \quad n^{\log_2(4)} = n^2$

$f(n)/n^2 \lg n = \lg n$ which shows that $f(n)$ is asymptotically larger, not polynomially. Thus, the master theorem can't be applied. Using the recursion tree method:

$T(n) = 4T(n/2) + n^2 \lg n \ldots$

$T(n/2) = 4T(n/4) + (n/4)\lg(n/4) + n^2 \lg n \ldots$

$T(n/4) = 4T(n/16) + (n/16)\lg(n/16) + (n/4)\lg(n/4) + n^2 \lg n \ldots$

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

$O(n^2(\lg n)^2)$

Renju Radhakrishnan

6. (10 pts) Use a recursion tree to give an asymptotically tight solution to the recurrence:
(a) $T(n) = T(n - a) + T(a) + cn$, where $a \geq 1$ and $c > 0$ are constants.
(b) $T(n) = T(an) + T((1 - a)n) + cn$, where $a \in (0, 1)$ and $c > 0$ are constants..

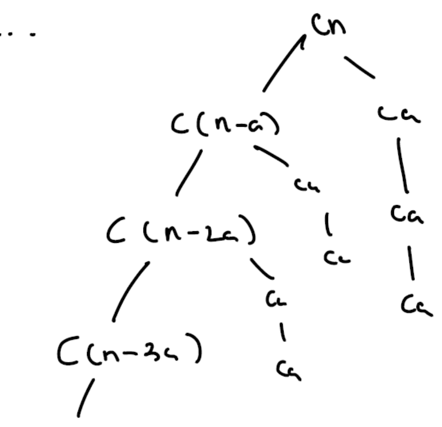
(a) $T(n) = T(n-a) + T(a) + cn$, where $a \geq 1$ and $c > 0$ are constants

$T(n-a) = T(n-2a) + 2T(a) + c(n-a) + cn$

$T(n-2a) = T(n-4a) + 3T(a) + c(n-3a) + c(n-a) + cn$

$T(n-4a) = T(n-8a) + 4T(c) + c(n-7a) + c(n-6a) \cdots \quad c(n)$

....



$T(1) \quad T(n) = (n/a)\,cn = O(n^2)$

Renju Radhakrishnan

b. $T(n) = T(an) + T((1-a)n) + cn$ where $a \in (0,1)$, and $c > 0$ are constants.

$$\begin{tikzcd}
an \arrow[d, dash] & (1-a)n \arrow[d, dash] & cn \arrow[d] \\
a(n+1) & (1-a)(n+1) & c(n+1)
\end{tikzcd}$$

$O(n)$ constant run time

Renju Radhakrishnan

7. (10 pts) For the following recurrences, find the bound implied by the master theorem. Then, try to prove the same bound using the substitution method. If your initial approach fails, show how it fails, and try subtracting off a lower-order term to make it work:

a. $T(n) = 4T(n/3) + n$

$a=4 \quad b=3$

$n^{\log_3(4)}$

$T(n) = \Theta(n^{\log_3 4})$ by case 1

$T(n) \leq 4(cn^{\log_3 4}/3) + n$

$= \frac{4}{3} cn^{\log_3 4} + n$

this holds as long as $c > 1$ ✓

Renju Radhakrishnan

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

$a = 4 \quad b = 2$

$n^{\log_2(4)} = n^2 \quad f(n) = O(n^{2-1})$ by

case 1 ✓

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

$T(n) = 4(c \cdot n^2/2) + n$

$= \frac{4}{2}(c \cdot n^2) + n$

$= 2cn^2 + n$

where $c > 1$ ✓

Renju Radhakrishnan

8. (5 pts) Solve the recurrence T(n) = 3T(√‾n) + log n by making a change of variable. Your solution should be asymptotically tight. Do not worry about whether values are integral.

$T(n) = 3T(\sqrt{n}) + \log n$

$x = \log(n)$

$T(n) = 3T(\sqrt{n}) + x$

$T(2^x) = 3T(2^{x/2}) + x$

$A(x) < cx^{\log 3} + dx$

$A(x) \le 3(c(x/2)^{\log 3} + d(x/2)) + x$ when $d \ge -2$

$\le cx^{\log 3} + (3/2\, d + 1)x$

$\le cx^{\log 3} + dx$

$A(x) \ge cx^{\log 3} + dx$ when $d \le -2$

. . . .

$\le cx^{\log 3} + dx$

$\Rightarrow A(x) = \Theta(x^{\log 3}) \Rightarrow T(n) = \Theta(\log^{\log 3} n)$

Renju Radhakrishnan

9. (5 pts) Using proof by induction, show that there are at most $\lceil n/2^{h+1} \rceil$ nodes of height h in any n-element heap.
Base case: Knowing that for any n > 0, the number of leaves of an almost complete binary tree is n/2

Knowing that for any $n > 0$, the number of leaves of an almost complete binary tree is $n/2$

Base case: We know that $h=0$ is true from the previous statement.

Proof by induction: Lets assume the statement is true for $h-1$. Let N be the # of nodes at height h. Looking at a tree T, it will have $n' = n - \lceil n/2 \rceil$ nodes thus

$$
\begin{aligned}
N &= n'/2^h \\
&= n/2 \,/ 2^h \\
&= (n/2)/2^h \\
&= \frac{n}{2^{h+1}}
\end{aligned}
$$