Solutions to Problem 1 of Homework 2 (14+10 points)

Name: Advait Pravin Savant (N14108183) *Due: 8 am on Thursday, September 23*

Collaborators: NetID1, NetID2

Consider the recurrence $T(n) = 2T\left(\frac{n}{2}+1\right) + n - 2$. Further, let $T(3) = 0$. Assume that n is of the appropriate form such that it is always an integer, even as we go down the formula.

(a) (8 points) Prove that $T(n) = \Theta((n-2)\log(n-2))$ using the "guess-then-verify" method. (**Hint**: Recall that if $f(n) = \Theta(g(n))$, then there exists constants $c_1, c_2 > 0$ such that $c_1 g(n) \le f(n) \le c_2 g(n)$ for all $n \ge n_0 > 0$.)

Solution:
The guess then verify substitution method will have a proof by induction.

For the base case,

$$T(3) = 0 = 1log1$$

Hence the theta notation is valid in the base case.
Assume that for m=(n/2+1),the given theta notation is valid,

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

$$T(\frac{n}{2}+1) = \Theta\left((\frac{n}{2}+1) - 2)\log((\frac{n}{2}+1) - 2))\right)$$

There exist c_1, c_2 greater than zero such that for sufficiently large n such that,

$$c_2\left((\frac{n}{2}+1) - 2)\log((\frac{n}{2}+1) - 2))\right) \le T(\frac{n}{2}+1) \le c_1\left((\frac{n}{2}+1) - 2)\log((\frac{n}{2}+1) - 2))\right)$$

$$c_2\left((\frac{n-2}{2})\log((\frac{n-2}{2}))\right) \le T(\frac{n}{2}+1) \le c_1\left((\frac{n-2}{2})\log((\frac{n-2}{2})\right)$$

adding (n-2) to the above inequality as n is sufficiently large, we get the expression for T(n) as seen earlier.

$$c_2\left((\frac{n-2}{2})\log((\frac{n-2}{2}))\right) + (n-2) \le T(\frac{n}{2}+1) + (n-2) \le c_1\left((\frac{n-2}{2})\log((\frac{n-2}{2})\right) + (n-2)$$

$$c_2\left((\frac{n-2}{2})\log((\frac{n-2}{2}))\right) + (n-2) \le T(n) \le c_1\left((\frac{n-2}{2})\log((\frac{n-2}{2})\right) + (n-2)$$

let this be inequality set 1.

$$c_1\left((\frac{n-2}{2})\log((\frac{n-2}{2})\right) + (n-2) = (c_1/2) * (n-2)log(n-2) - (c_1/2)(n-2)log2 + (n-2)$$

As we look at the growth of the above function, we can see that the (n-2)log(n-2) term will dominate, this can be shown formally with limits, as n tends to infinity, we can find some constant c_3 such that for all sufficiently large n,

$$c_1\left((\frac{n-2}{2})\log((\frac{n-2}{2})\right)+(n-2)\leq c_3(n-2)log(n-2))$$

z Let this be inequality set 2.

The math works out as follows,

$$(c_1/2)*(n-2)log(n-2)+(1-c_1/2)(n-2)\leq c_3(n-2)log(n-2)$$

$$(1-c_1/2)(n-2)\leq(c_3-c_1/2)(n-2)log(n-2)$$

$$(1-c_1/2)\leq(c_3-c_1/2)log(n-2)$$

For some sufficiently large $n=n_0$ and given c_1 as per our assumption, we can see that this is solvable for c_3. Similarly, can also find that their exists some sufficiently small constant c_4 relative to c_2 such that for all sufficiently large $n>n_0$,

$$c_4(n-2)log(n-2)\leq c_2\left((\frac{n-2}{2})\log((\frac{n-2}{2}))\right)+(n-2)$$

let this be inequality set 3.

From inequality sets 1,2 and 3, we get,

$$c_4\,(n-2)\log(n-2))\leq T(n)\leq c_1\,((n-2)\log(n-2))$$

Hence, $T(n)=\Theta((n-2)\log(n-2))$. This completes our proof by induction for the substitution.

□

(b) (**Extra Credit**)(6 points) Solve the same recurrence relation by the domain-range substitution. Namely, make several changes of variables until you get a basic recurrence of the form $R(k)=R(k-1)+f(k)$ for some function f, and then compute the answer from there. This answer will have to be exact and not an asymptotic answer. For full credit, you will have to each of the intermediate substitutions and calculations. (**Hint**: As a first step, you will let $n=c^m+d$ for some constants c,d. What is this choice of c,d?)

Solution: Let $n=2^k+2$

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

$$T(2^k+2)=2T\left(\frac{2^k+2}{2}+1\right)+2^k+2-2$$

$$T(2^k+2)=2T\left(2^{k-1}+2\right)+2^k$$

let,

$$S(k) = T(2^k + 2)$$

Therefore,

$$S(k) = 2S\left(k-1\right) + 2^k$$

$$S(k)/2^k = S\left(k-1\right)/2^{k-1} + 1$$

$$S(k)/2^k = f(k)$$

$$f(k) = f\left(k-1\right) + 1$$

This is a telescoping series, using the appropriate base case for k=0 and adding the equations we get,

$$f(k) = \theta(k)$$

Therefore,

$$S(k)/2^k = \theta(k)$$

$$S(k) = 2^k\theta(k)$$
$$T(2^k + 2) = 2^k\theta(k)$$
$$T(n) = 2^k\theta(k)$$

We know,

$$k = \log(n-2)$$

Therefore,

$$T(n) = (n-2)\theta(log(n-2))$$

Assuming the second function as argument to the theta notation to be g, showing the inequality based bounds for g as per the theta notation, multiplying this by (n-2), as n is a large positive number, the inequality signs will remain, we will get inequality bounds for (n-2)*g. From here, using the definition of the theta notation we easily can show that,

$$T(n) = \theta((n-2)log(n-2))$$

□

(c) (6 points) Consider the recurrence relation

$$T(n) = 2T\left(n-1\right) + n, T(0) = 1.$$

Solve by recursion tree method. Towards this end, you will first complete the empty empty fields at this table. You can then conclude that $T(n)$ is the sum of the values in the last column. You will then simplify the sum to give a tight asymptotic bound for $T(n)$, i.e., $T(n) = \Theta(c^n \cdot n^d \cdot (\log n)^r)$ for some constants $c > 1, d, r > 0$.

(**Extra Credit**)(3 points) You get an additional three points if you solve for the value of $T(n)$ exactly and correctly.

(**Hint**: You will find the following formula very useful: Let

$$S(a,d) = \sum_{i=1}^{d} i \cdot a^i = a + 2a^2 + 3a^3 + \ldots + d \cdot a^d = \frac{d \cdot a^{d+2} - a^{d+1} \cdot (d+1) + a}{(a-1)^2} = \Theta(d \cdot a^d)$$

)

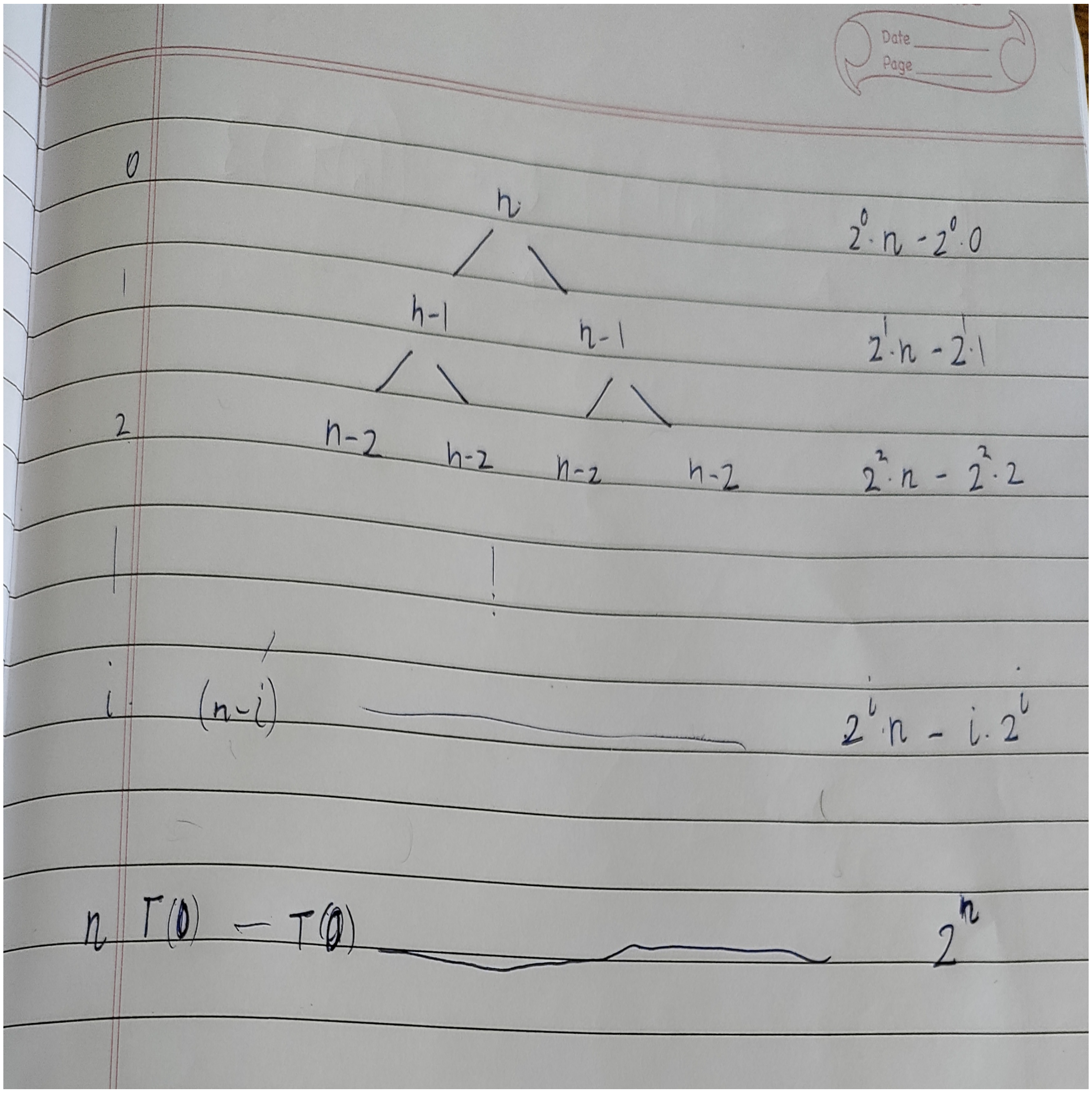
Level	Size of Problem	No. of Problems	Non-Recursive Cost of One Problem	Total Cost
0	T(n)	1	n	n
1	T(n-1)	2	n-1	2n-2
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
i	T(n-i)	2^i	n-i	2^i(n-i)
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$d-1$	T(1)	2^{n-1}	1	2^{n-1}
d	T(0)	2^n	1(T(0)=1)	2^n

Here, d is the depth of the recursion tree. (**Hint**: To produce a closed form solution)

Solution:
Given,

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

Consider the next picture, We can see from the recursion tree, for level i in the tree, the cost will be $2^i(n-i)$, until depth n-1, we consider constant T(0) sums for the leaves. Each node at level i will be of cost (n-i) and there will be 2^i such nodes.



Therefore,

$$T(n) = \sum_{i=0}^{n-1}(2^i n - 2^i i) + 2^n$$

$$T(n) = n\sum_{i=0}^{n-1}(2^i) + 0 - \sum_{i=1}^{n-1}(i2^i) + 2^n$$

The first series is a geometric progression and the second series is a arithmetic geometric progression whose expansions are known to us.

$$T(n) = n(2^n - 1) - ((n-1)2^{n+1} - n2^n + 2) + 2^n$$

$$T(n) = n(2^n - 1) - (2n2^n - 22^n - n2^n + 2) + 2^n$$

Therefore,

$$T(n) = 3.2^n - n - 2$$

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

$\square$

CSCI-GA.1170-001/002/003 Fundamental Algorithms	September 22, 2021

Solutions to Problem 2 of Homework 2 (15+1 Points)

Name: Advait Pravin Savant (N14108183) *Due: 8 am on Thursday, September 23*

Collaborators: NetID1, NetID2

(a) (5 points) Consider the following recurrence:

$$T(n,1) = 3n$$
$$T(1,m) = 3m$$
$$T(n,m) = 3n + T(n/3, m/3)$$

Solve for $T(n, n^2)$ to get a tight asymptotic bound. Assume that n is an exponent of 3 for simplicity.

Extra Credit (1 point): Keep track of the leading coefficient rather than just state $T(n) = \Theta(f(n))$. This means that express $T(n) = c \cdot f(n) + o(f(n))$ for an appropriate choice $c, f(n)$.

Solution: Substituting for $T(n, n^2)$ in the given recurrence relations,

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

$$T(n/3, n^2/3) = 3\frac{n}{3} + T(n/9, n^2/9)$$

$$T(n/9, n^2/9) = 3\frac{n}{9} + T(n/27, n^2/27)$$

Continuing this way, we look for where we get the base case in the telescoping series. Consider next,

$$T\left(\frac{n}{3^{\log_3 n-1}}, \frac{n^2}{3^{\log_3 n-1}}\right) = 3\frac{n}{3^{\log_3 n-1}} + T\left(\frac{n}{3^{\log_3 n}}, \frac{n^2}{3^{\log_3 n}}\right)$$

The next term becomes,

$$T\left(\frac{n}{3^{\log_3 n}}, \frac{n^2}{3^{\log_3 n}}\right) = T(1,n) = 3n$$

We add the terms in this series to get,

$$T(n, n^2) = 3n\left(1 + \frac{1}{3} + \frac{1}{9} + ... + \frac{1}{3^{\log_3 n-1}}\right) + 3n$$

$$T(n, n^2) = 3n.\frac{1-(1/3)^{log_3 n}}{1-(1/3)} + 3n$$

$$T(n, n^2) = \frac{15}{2}n - \frac{9}{2}$$

$$T(n, n^2) = \theta(n)$$
$$T(n, n^2) = \frac{15}{2}n + o(n)$$

□

(b) (5 points) Consider the following recurrence

$$T(n) = T(0.01n) + T(0.99n) + cn$$

where $c > 0$ is a constant. Solve for $T(n)$ by the recursion tree method to derive an asymptotically tight solution. (**Hint**: Note that this tree is not going to be a balanced tree. What is the depth of the shortest branch? What is the depth of the largest branch? Use this to derive a Θ bound for $T(n)$.)

Solution: The first level of the recursion tree will have node cn which will branch into 2 nodes, one for T(0.01n) and one for T(0.99n).

$$T(n) = T(0.01n) + T(0.99n) + cn$$

The next level will have two nodes, 0.01n and 0.99n which will branch accordingly as per the substitution in the recurrence.The next level will have 4 corresponding nodes.

The key insight is that the ith level will have as nodes the coefficients corresponding to the of binomial expansion of $(0.01+0.99)^i$ times cn as long as the cn term remains. The summation at that level will be the binomial expansion times cn which is essentially cn. We will deal with singular cases soon. This can be proved by induction.

for n=0, we have node cn.

for n=1, we have nodes 0.01cn +0.09cn which follow our hypothesis.

for n=i,

$$T(i) = (0.01 + 0.99)^i cn = \sum_{r=0}^{i} \binom{i}{r} * 0.01^r 0.09^{i-r} cn$$

Now, in order to get to the next level, each term will be multiplied by 0.01 and 0.09 each. That new term in the next level will be added in the summation at the next level.

$$T(i) = \sum_{r=0}^{i} \binom{i}{r} * 0.01^r 0.09^{i-r+1} cn + \sum_{r=0}^{i} \binom{i}{r} * 0.01^{r+1} 0.09^{i-r} cn$$

$$T(i+1) = 0.09 \sum_{r=0}^{i} \binom{i}{r} * 0.01^r 0.09^{i-r} cn + 0.01 \sum_{r=0}^{i} \binom{i}{r} * 0.01^r 0.09^{i-r} cn$$

$$T(i+1) = \left(\sum_{r=0}^{i} \binom{i}{r} * 0.01^r 0.09^{i-r} cn \right) (0.01 + 0.09)$$

$$T(i+1) = (0.01 + 0.99)^{i+1} cn$$

Which proves our induction hypothesis. Now, consider the branching. The shortest branch will correspond to the recursive 0.01 multiplications one after the other and the longest branch will correspond to the recursive 0.99 multiplications. let k,m be the lengths of shortest and longest branches respectively.

$$(0.01)^k n = 1$$

$$k \log_{100}(0.01) + \log_{100}(n) = 0$$

$$k = \log_{100}(n)$$

Similarly,

$$m \log_{100/99}(99/100) + \log_{100/99}(n) = 0$$

$$m = \log_{100/99}(n)$$

Using properties of the logarithm,

$$m = \frac{\log_{100} n}{log_{100} 100/99}$$

There will be at least k complete levels each of which will give cn as their sum. For succeeding levels, there will be base case atomic sub problems which need not be divided further. Hence the levels after that will be partially incomplete with certain branches terminating. The longest branch, which is the last to terminate, will be corresponding to m. Based on our arguments, we can see that the total cost will be lower bounded by k*cn and upper bounded by m*cn.

$$k * cn \leq T(n) \leq m * cn$$

$$\log_{100}(n) * cn \leq T(n) \leq \frac{\log_{100} n}{log_{100} 100/99} * cn$$

This is essentially the definition for our theta notation. The log bases can be adjusted using properties of the logarithm.

$$\log_a b = \frac{\log_c b}{\log_c a}$$

We get,

$$T(n) = \theta(nlogn)$$

□

(c) (5 Points) Consider the following recurrence

$$T(n) = 9T(n/3) + \frac{n^2}{\log_3 n}.$$

Solve for $T(n)$ by domain-range substitution. Namely, make several changes of variables until you get a basic recurrence of the form $R(k) = R(k-1) + f(k)$ for some f, and then compute the answer from there. You may assume that n is a power of 3. (**Hint**: Begin by expressing n as a function of another variable m. Then change the variable T. You will change T to a function S and then to R. Note: The two changes can be combined in one step too.)

Solution:

$$T(n) = 9T(n/3) + \frac{n^2}{\log_3 n}$$

let $n = 3^k$

$$T(3^k) = 9T(3^{k-1}) + \frac{3^{2k}}{k}$$

let $T(3^k) = S(k)$

$$S(k) = 9S(k-1) + \frac{3^{2k}}{k}$$

$$S(k)/9^k = S(k-1)/9^{k-1} + \frac{3^{2k}}{k * 9^k}$$

let $S(k)/9^k = f(k)$

$$f(k) = f(k-1) + \frac{1}{k}$$

$$f(k-1) = f(k) + \frac{1}{k-1}$$

...

$$f(2) = f(1) + \frac{1}{2}$$

f(1) is our base case which corresponds to n=3.

$$f(1) = 1$$

Therefore,

$$f(k) = \sum_{i=1}^{k} \frac{1}{i}$$

As n grows large, the harmonic series summation is proportional to the logarithm of n. This can be ascertained using the nature of integration of 1/x and the second fundamental theorem of calculus.

$$f(k) = logk + O(1)$$

$$S(k)/9^k = logk + O(1)$$

$$S(k) = 9^k logk + 9^k O(1)$$

$$S(k) = 9^k logk + 9^k O(1)$$

$$T(3^k) = \theta(9^k logk)$$

Therefore,

$$T(n) = \theta(n^2 log(logn))$$

$\square$

CSCI-GA.1170-001/002/003 Fundamental Algorithms	September 22, 2021

Solutions to Problem 3 of Homework 2 (7+5 points)

Name: Advait Pravin Savant (N14108183) *Due: 8 am on Thursday, September 23*

Collaborators: NetID1, NetID2

Consider the pseudocode for the following randomized algorithm:

```
BLA(n)
    if n ≤ 5 then return 1
    else
        Assign x value of 0 with probability 1/4 and 1 with probability 3/4
        if x = 1 then return BLA(n)
        else return BLA(n/3)
```

(a) (5 points) Let $T(n)$ denote the expected running time of BLA. Derive a recurrence equation for $T(n)$. Solve your recurrence relation using Master Theorem to obtain a asymptotically tight bound.

Solution:

Let R(n) be the random function denoting the running time of the algorithm for input of size n. R(n) will take on values and the act of R(n) taking on values can be assigned a probability distribution. As we run BLA(n), with probability 3/4, we call the same function again and with probability 1/4 we call BLA(n/3). The operations for this particular function call constituting comparisons and assigning probabilities to variable x take constant time.

$$P(R(n) = k + 1) = \frac{3}{4}P(R(n) = k) + \frac{1}{4}P(R(n/3) = k)$$
$$R(n) = \frac{3}{4}R(n) + \frac{1}{4}R(n/3) + 1$$

$$\frac{1}{4}R(n) = \frac{1}{4}R(n/3) + 1$$

Using the linearity of expectation,

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

$$T(3^k) = T(3^{k-1}) + 4$$

We have dealt with similar telescoping series earlier, we can also verify with the master theorem for T(n).

$$T(3^k) = \theta(k)$$

$$T(n) = \theta(logn)$$

$\square$

(b) (2 points) What is the functionality of this algorithm, i.e., what is the expected value returned by this algorithm? Justify your answer briefly.

Solution:

Since the same value of 1 is returned after the completion of recursion for every function call, the expected value of such a constant function will be 1 itself. In summation for the expectation, the constant, here 1, is taken outside, the probabilities will add to one.

$\square$

Consider the pseudo code for the following randomized algorithm:

FOO(n)
  if $n \leq 1$ **then return** 5
  else
    for $i = 1$ **to** n **do**
      continue
    Assign x value of 0 with probability 1/4 and 1 with probability 3/4
    if $x = 1$ **then return** FOO(n)
    else return BLA(FOO($n/3$))

(c) (**Extra Credit**)(5 points) Let $S(n)$ denote the expected running time of FOO. Derive a recurrence equation for $S(n)$. Solve your recurrence relation by the Recursion Tree Method, by completing the table as shown below:

Solution: Let S(n) denote the random function for the running time of Foo(n). R denotes the running time for BLA as seen previously and for each S(n) call, the operations specific to that call take linear time relatively to n. We have,

$$S(n) = \frac{1}{4}S(n) + \frac{3}{4}R(S(n/3)) + n$$

$$S(n) = \frac{1}{4}S(n) + \frac{3}{4}R(S(n/3)) + n$$

$$S(n) = \frac{1}{4}S(n) + \frac{3}{4}R(S(n/3)) + n$$

$$S(n) = R(S(n/3)) + \frac{4}{3}n$$

Taking the expectation,

$$S(n) = R(S(n/3)) + \frac{4}{3}n$$

$$E[S(n)] = E[R(S(n/3))] + E[(4/3)n]$$

The expectation operation of a function of consisting of more than one random variables is taken relative the variables and the function under consideration. For the second expectation, we can rearrange the terms in the summation for the expectation operation, for some given S(n/3), the summation will be over R, and we will get the expectation for R which we know as the log, but this must done for all S(n/3) in order to complete the expectation operation.After we sum over R, we must also sum over S values. What we get is E[log(S(n/3))] which is T(n/3) and is the expectation relative to S values.

$$E[S(n)] = E[log(S(n/3)] + E[(4/3)n]$$
$$T(n) = E[log(S(n/3)] + (4/3)n$$

Here T(n) is the expected running time for FOO. For BLA, we have seen the logarithmic expected running time previously.

We can see that,

$$(4/3)n \leq T(n)$$

Now consider Jensen's inequality for the logarithm which is a concave function.

$$E[log(X)] \leq log[E(X)]$$
$$E[log[S(n/3)] \leq log(E[S(n/3)])$$

Therefore,

$$E[log[S(n/3)] \leq log(T(n/3))$$

Using this result in our equation for T(n),

$$E[log[S(n/3)] + (4/3)n \leq log(T(n/3)) + (4/3)n$$

$$T(n) \leq log(T(n/3)) + (4/3)n$$

Consider the recurrence equation for the upper bound on T(n).

$$T(n) = log(T(n/3)) + (4/3)n$$

$$T((n/3)) = log(T(n/9)) + (4/3)(n/3)$$
$$T((n/9)) = log(T(n/27)) + (4/3)(n/9)$$

...

We have done similar series earlier, we can verify easily by master theorem that the bound comes out to be linear in n. Note that this is the upper bound for our T(n). We see that both upper and lower bounds are linear in n. This is equivalent to the definition of our theta notation. Hence,

$$T(n) = \theta(n)$$

$\square$