A1-S20190010034

C Bhavesh Kumar

August 2020

1 Fun with Big-O Notation:

(a) $n = O(n * log(n))$: This Statement is True.

To prove this we use the definition of $O(.)$.By the Definition of $O(.)$, $\exists c, n_0 > 0$ such that the inequality $c * nlog(n) \geq n$ is satisfied
$=> c * log(n) \geq 1$
$=> c \geq 1/log(n)$
This is possible for all $n > 1$ and the corresponding values of c.
For example c=1 and n_0=1, which satisfies the inequality and the condition $n \geq n_0$.

(b) $n^{1/log(n)} = \Theta(1)$: This Statement is True.
The following problem can be simplified as
$=> n^{log_2(2)/log_2(n)}$
$=> n^(log_n(2))$
$=> T(n) = 2$
To prove this we use the definition of $\Theta(.)$
By the definition of $\Theta(.)$ the expression must follow the definitions of O(.) and $\Omega(.)$
By the Definition of $O(.)$, $\exists c, n_0 > 0$ such that the inequality $2 \leq c * 1$ holds.
$=> c \geq 2$, As the expression doesn't involve n, The inequality is satisfied when $c \geq 2$ $\forall n > 0$.
Therefore the values of c and n_0 are $c \geq 2$ and $n_0 = 0$.
Similarily By the Definition of $\Omega(.)$, $\exists c, n_0 > 0$ such that the inequality $2 \geq c * 1$ holds.
$=> c \leq 2$, As the inequality doesn't depend on n it is true $\forall n > 0$.
Therefore the values of c and n_0 are $0 < c \leq 2$ and $n_0 = 0$.
Since it follows O(.) and $\Omega(.)$ it follows $\Theta(.)$.

(c) $f(n) = 5^n$ if $n < 2^{1000}$ and
$f(n) = 2^{1000}n^2$ if $n \geq 2^{1000}$
This statement is True.
To prove this we use the definition of O(.) and guess the values of c and

n_0 and check whether they follow the definition.
By the Definition of $O(.)$, $\exists c, n_0 > 0$ such that the inequalities given below hold.
$5^n \leq c * n^2/2^{1000}$ for $n_0 \leq n < 2^{1000}$ and
$2^{1000}n^2 \leq c.n^2/2^{1000}$ for $n \geq n_0$ and $n \geq 2^{1000}$
The both inequalities hold for $n_0 = 2^{1000}$ and $c \geq 2^{2000}$. Since $n_0 = 2^{1000}$, the first part of f(n) is excluded [as $n < 2^{1000}$ and as per definition $n \geq n_0$] and only second part is considered, which satisfies the condition.
Since the values of c and n_0 exists, the given statement is true.

(d) if $f(n) = O(g(n))$ then $2^{f(n)} = O(2^{g(n)})$
The above statement is False
To prove this we provide a counter example.
By the definition of O(.), $\exists c, n_0 > 0$ such that $f(n) \leq c * g(n)$.[Given]
let $f(n) = k * log(n)$ and $g(n) = log(n) where\ k \geq 0$
The above example follows the definition of O(.) and $f(n) * g(n) \geq 0$
Now according to question $2^{f(n)} = O(2^{g(n)})$, if this is true,
=> $2^{k*log(n)} \leq 2^{log(n)}$
=> $n^k \leq c * n$
$n^k = O(n)$
Which is false for all $k \geq 1$
=> $\forall k \geq 1$ the given examples serves as a counterexample for the given statement
Therefore the given statement is false

(e) $5^{log(log(n))} = O(log^2(n))$
The given statement is True
To prove this we use definition of O(.) and assume c=1 and $n_0 = 10$
By the definition of O(.), $\exists c, n_0 > 0$ such that
$5^{log(log(n))} \leq c * log^2(n)$
for $n_0 = 10$ and c=1 the above condition is satisfied.
For c=1 and $n \geq 10$ $5^{log(log(n))} \leq c * log^2(n)$.
Therefore the given statement is true.

(f) n = $\Theta(100^{log(n)})$ The given Statement is False.
This is because when T(n) is in terms of Θ it must follow both O(.) and Ω(.).In this case the function follows O(.) but not Ω(.).Proof is given below.
$\Theta(100^{log(n)})$ can be simplified as
=> $\Theta(100^{log_{10}(n)*log_2(10)})$
=> $\Theta(100^{log_{10}n})$
=> $\Theta(n^2)$
By the Definition of $O(.)$, $\exists c, n_0 > 0$ such that $n \leq c * n^2$
=> $n * c \geq 1$ => $n \geq 1/c$.
Therefore $\forall n \geq 1/k$ and c=k the above definition and inequality is satisfied.
For example, for c=1 and $n \geq 1$ the inequality holds.

Since such a pair exists, $n = O(100^{log(n)})$—(1)

By the Definition of $\Omega(.)$, $\exists c, n_0 > 0$ such that $n \geq c * n^2 => n \leq 1/c$
for $n = 1 + 1/k$ and c=k the above condition is not satisfied.
This works as a counterexample and hence can prove that $n \neq \Omega(100^{log(n)})$—(2)
From (1) and (2) we can tell that $n \neq \Theta(100^{log(n)})$

2 Fun with recurrences

(a) $T(n) = 2T(n/2) + 3n$
We apply master Theorem with a=b=6 and with d=1,we have $a = b^d$ and so the runtime is O(nlogn).

(b) $T(n) = 3T(n/4) + \sqrt{n}$
We apply master Theorem with a=3,b=4,and with $d = 1/2$,we have $a > b^d$ $[5 > 2]$ and so the runtime is $O(n^{log_4 3})$

(c) $T(n) = 7T(n/2) + \Theta(n^3)$
We apply Master Theorem with a=7,b=2 and with d=3 we have $a < b^d$ $[7 < 8]$ and so the runtime is $O(n^3)$

(d) $T(n) = 4T(n/2) + n^2 logn$
We solve this using substitution method
$=> T(n/2) = 4T(n/4) + n^2/4log(n/2)$
$=> T(n) = 16T(n/4) + n^2(log(n/2) + log(n))$
.
.
$=> T(n) = 4^k T(n/2^k) + n^2 * (log(n) + log(n/2) + ... + log(n/2^{k-1}))$
This recursion ends when $n/2^k = 1$ that is when $k = log(n)$
$=> T(n) = n^2 + n^2 * (log(n) + log(n/2) + ... + log(n/2^{k-1}))$
Lets denote $(log(n) + log(n/2) + ... + log(n/2^{k-1}))$ as A
A=$(log(n) + log(n/2) + ... + log(n/2^{k-1}))$ and $T(n) = n^2 + A$
$=> A = log(n * n/2 * n/4 * ... * n/2^{k-1})$
$=> A = log(n^k/2^{k*(k-1)/2})$
$=> A = klog(n) - log(2) * k * (k-1)/2$
$=> A = log^2(n) - log^2(n)/2 + log(n)/2$
$=> A = log^2(n)/2 + log(n)/2$
$=> T(n) = n^2 + n^2 * (log^2(n)/2 + log(n)/2)$
Therefore $T(n) = O(n^2 log^2(n))$

(e) $T(n) = 2T(n/3) + n^c$
We apply Master Theorem with a=2,b=3 and with d=c where $c \geq 1$ we have $a < b^c$ [as $2 < 3^c forc \geq 1$] and so the runtime is $O(n^c)$

(f) $T(n) = 2T(\sqrt{n}) + 1$ T(2)=1
We use substitution method to solve this problem

$=> T(n) = 4 * T(n^{1/4}) + 2 + 1$
.
.
$=> T(n) = 2^k T(n^{1/2^k}) + (1 + 2 + 4 + ... + 2^{k-1})$
Lets denote $(1 + 2 + 4 + ... + 2^{k-1})$ as A
$=> T(n) = 2^k T(n^{1/2^k}) + A$
$=> A = 1 + 2 + 4 + ... + 2^{k-1}$ This is nothing but geometric series.
$=> A = 1 * (2^k - 1)/(2 - 1)$
$=> A = 2^k - 1$
$=> T(n) = 2^k T(n^{1/2^k}) + 2^k - 1$
This recursion ends when $n^{1/2^k} = 2$
$=> 1/2^k * log(n) = 1$
$=> 2^k = log(n)$
$=> T(n) = log(n) + log(n) - 1$
Therefore $T(n) = O(log(n))$

3 3.Different Sized Sub Problem

$T(n) = T(n/2) + T(n/4) + T(n/8) + n$
To Prove this statement we use recursion tree/substitution to solve.

The given equation can be drawn as a tree with starting node as T(n) which divides into T(n/2),T(n/4),T(n/8).
These further divide into T(n/4),T(n/8),T(n/16),T(n/32),T(n/64).

Therefore by the structure of tree :

The contribution of the first layer is n.
The contribution of the second layer is $n/2 + n/4 + n/8 = 7n/8 = (7/8)^1 n$.
The contribution of third layer is $n/4 + n/8 + n/16 + n/8 + n/16 + n/32 + n/16 + n/32 + n/64 = 49n/64 = (7/8)^2 n$.
..
..
The contribution of k-th layer is $(7/8)^{k-1} n$.

The recursion ends when $n/2^k = 1$.
$=> k = log(n)$.Here k is the height of tree which is log(n).

T(n) = Total contribution [Sum of contributions of all layers].
$=> T(n) = n + (7/8)n + (7/8)^2 n + ... + (7/8)^{k-1} n$
$=> T(n) = n * (1 + (7/8) + (7/8)^2 + ... + (7/8)^{k-1})$
By using sum of n terms in geometric progression with $a = 1$ and $r = 7/8$, We have
T(n) = $n * (1 - (7/8)^k)/(1 - 7/8)$
$=> T(n) = 8 * n * (1 - (7/8)^k)$

$=> T(n) = 8 * n * (1 - (7/8)^{log(n)})$ [substituting the value of k]
$=> T(n) = O(n)$
Therefore, the time complexity of the given equation is O(n)

4 What's wrong with this proof?

(a) Claim 1 and Claim 3 are correct.

(b) Claim 2 is incorrect.
This is because b is violating the condition in master's theorem, b is supposed to be independent of n but in this case it is dependent on n.
Also when n=5, b=infinity which doesn't make sense according to the logic of master's theorem.
Therefore the proof contains many ambiguities and hence is not correct.