CS21003: Algorithms-I (Theory)

Tutorial – 1 (Algorithmic Time Complexity and Recurrences)

Date: 16-January-2020

1. Compare the following functions based on asymptotic notations:
   (a) $f(n) = \sqrt{n}$ and $g(n) = (\log n)^2$
   (b) $f(n) = \log(\log n)$ and $g(n) = \sqrt{n}$
   (c) $f(n) = n^{1.5}$ and $g(n) = n \log n$
2. Prove the following statements for the non-negative functions, $f(n)$, $g(n)$, $h(n)$, $g_1(n)$ and $g_2(n)$:
   (a) If $f(n) = O(g(n))$ and $g(n) = O(h(n))$, then prove that $f(n) = O(h(n))$.
   (b) If $f(n) = O(g_1(n))$ and $f(n) = O(g_2(n))$, then prove that, $f(n) = O\big(\texttt{MIN}(g_1(n), g_2(n))\big)$.
   (c) If $f(n) = \Omega(g_1(n))$ and $f(n) = \Omega(g_2(n))$, then prove that, $f(n) = \Omega\big(\texttt{MAX}(g_1(n), g_2(n))\big)$.
3. Argue whether $2^n = O(2^{n-1})$?
   If the above is YES, then determine the fallacy in the following derivation:
   $2^n = O(2^{n-1})$ and $2^{n-1} = O(2^{n-2})$ implies $2^n = O(2^{n-2})$. (Using Problem-2(a) Statement)
   Now, $2^n = O(2^{n-2})$ and $2^{n-2} = O(2^{n-3})$ implies $2^n = O(2^{n-3})$, and so on ...
   Continuing in this way, we get $2^n = O(2^{n-1}) = O(2^{n-2}) = \ldots = O(2^1) = O(2^0) = O(1) = constant$.
4. Find two functions $f(n)$ and $g(n)$ such that, neither $f(n) = O(g(n))$, nor $g(n) = O(f(n))$.
5. What is the time complexity of the following algorithms/programs? Explain.
   (a)
```
void fun ( int n ) {
    int j = 1, i = 0;
    while ( i < n ) {
        // Some constant-time tasks
        i = i + j;
        j++;
    }
}
```
   (b)
```
long int exponentiation ( int x , unsigned int n ) {
    if ( n == 0 )  return 1;
    if ( n == 1 )  return x;
    if ( n is EVEN )  return ( exponentiation(x*x,n/2) );
    else  return ( exponentiation(x*x,n/2) * x );
}
```
   What happens when the last line be: `else return ( exponentiation(x,n-1) * x );`
6. Let the running time of a recursive algorithm satisfy the recurrence: $T(n) = aT(\sqrt{n}) + h(n)$. Deduce the running time $T(n)$ in asymptotic Θ notation for the cases:
   (i) $h(n) = n^d$ for some $d \in \{1, 2, 3, \ldots\}$, and
   (ii) $h(n) = \log^d n$ for some $d \in \{0, 1, 2, \ldots\}$.
7. Solve the following recurrence relations:
   (a) $nT(n) = (n+1)T(n-1) + 2n$ for $n \geq 1$, with the initial condition $T(0) = 0$.
   (b) $T(n) = nT(n-1) + n(n-1)T(n-2) + n!$ for $n \geq 2$, with $T(0) = 0, T(1) = 1$.
8. Derive asymptotic time complexities from the following recurrences relation *using Master Theorem*:
   (a) $T(n) = 4T(n/2) + n$.
   (b) $T(n) = 4T(n/2) + n^2$.
   (c) $T(n) = 7T(n/2) + n^2$.
   (d) $T(n) = 7T(n/2) + n^3$