Recitation Problems - Com S 311

Week of Jan $22^{th} - 27^{th}$

1. Prove or disprove each of the following statements. For all problems assume the domain of the function is $\mathbb{N}$ (i.e., the set of natural numbers.)

   (a) $f(n) \in O(g(n))$, where $f(n) = n^5 - 1001n^4 + 30n^3$ and $g(n) = n^5$.

   Solution: Notice that $f(n) = n^5 - 1001n^4 + 30n^3 \leq 2n^5 = 2 \cdot g(n)$ for all $n \in \mathbb{N}$. Therefore, with $c = 2$ and $n_0 = 1$, $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$.

   (b) $f(n) \in O(g(n))$, where $f(n) = 2^{2^{n+2}}$ and $g(n) = 2^{2^{n+1}}$.

   Solution (1): We know if $f(n) \in O(g(n))$, then, $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$. Let's assume $f(n) \in O(g(n))$, therefore we have:

   $$\lim_{n\to\infty} \frac{2^{2^{n+2}}}{2^{2^{n+1}}} = \lim_{n\to\infty} 2^{2^{n+2}-2^{n+1}} = \lim_{n\to\infty} 2^{2^{n+1}(2-1)} = \lim_{n\to\infty} 2^{2^{n+1}} = \infty$$

   which contradicts our assumption. Thus, $f(n) \notin O(g(n))$.

   Solution (2): By contradiction, let's assume $f(n) \in O(g(n))$. So, there exists $c > 0$ and $n_0 \in \mathbb{N}$ such that $f(n) \leq c \cdot g(n)$ for all $n > n_0$. Therefore, we have

   $$\begin{aligned} 2^{2^{n+2}} \leq c \cdot 2^{2^{n+1}} &\to \log 2^{2^{n+2}} \leq \log c \cdot 2^{2^{n+1}} = \log c + \log 2^{2^{n+1}} \\ &\to 2^{n+2} \leq \log c + 2^{n+1} \\ &\to 2^{n+2} - 2^{n+1} \leq \log c \\ &\to 2^{n+1} \leq \log c, \end{aligned}$$

   since 2^{n+1} is a strictly increasing function, no constant c will satisfy this. This contradicts our initial assumption, hence $f(n) \notin O(g(n))$.

   (c) $f(n) \in O(g(n))$, where $f(n) = \log n$ and $g(n) = \sqrt{n}$.

   Solution: Observe that $f(n) = \log n = \log n^{2\times\frac{1}{2}} = 2\log\sqrt{n} \leq 2 \cdot \sqrt{n} = 2g(n)$ for all $n \geq 1$. Therefore, with $c = 2$ and $n_0 = 1$, $f(n) \leq c \cdot g(n)$ holds for all $n \geq n_0$.

   (d) If $f(n) \in O(g(n))$ and h is a any positive-valued function, then $f \cdot h(n) \in O(g \cdot h(n))$.

   Solution: Since $f(n) \in O(g(n))$, there exists $c > 0$ and $n_0 \in \mathbb{N}$ such that, for all $n \geq n_0$, $f(n) \leq c \cdot g(n)$. So, we can choose $c' = c$ and $n_0' = n_0$ such that, for all $n \geq n_0'$ and any positive-valued function $h(n)$, $f \cdot h(n) \leq c \cdot g \cdot h(n)$. Thus, $f \cdot h(n) \in O(g \cdot h(n))$.

2. Formally derive the runtime of each algorithm below as a function of n and determine its Big-O upper bound.

   (a)
```
for(i=1 to n)
    for(j=i to n)
        [some constant atomic operations]
```

   Solution: we will consider atomic operations take unit time:

$$\sum_{i=1}^{n}\sum_{j=i}^{n} 1 = \sum_{i=1}^{n} n-i+1 = \sum_{i=1}^{n}(n+1) - \sum_{i=1}^{n} i = n^2+n-\frac{n^2+n}{2} = \frac{n^2+n}{2} \in O(n^2)$$

   (b)
```
i=1
while(i < n){
    [some constant atomic operations]
    i *= 2
}
```

   Solution: this while loop iterate k times, where k is the largest natural number that $2^k < n$. Therefore, the runtime of this code snippet is $O(\log_2 n)$