Solutions of Problems 1 and 2 of Assignment #1

(Course: CS 401)

Problem 1:

(a) Take the following functions and arrange them in *descending* order of growth rates. That is, if a function $g_i(n)$ immediately follows function $g_j(n)$ in your list, then it should be the case that $g_i(n) = O\left(g_j(n)\right)$. **Justify your answer clearly**.

$$g_1(n) = 2^{(\log n)^{1/3}}$$

$$g_2(n) = 2^{n^2}$$

$$g_3(n) = n^{5/3}$$

$$g_4(n) = \frac{\sqrt{\log n}}{\log\log n}$$

$$g_5(n) = n^{(\log n)^2}$$

$$g_6(n) = 2^{(11/9)^n}$$

$$g_7(n) = (\log n)^{\log n}$$

(b) Give examples of two continuous functions $f(n)$ and $g(n)$ of positive real inputs n such that $f(n) \neq O(g(n))$, $g(n) \neq O(f(n))$, $f(n) \neq \Omega(g(n))$ and $g(n) \neq \Omega(f(n))$.

Solution:

(a)

$$g_4 = O(g_1) \equiv \frac{\sqrt{\log n}}{\log\log n} = O\left(2^{(\log n)^{1/3}}\right) \underbrace{\equiv \frac{1}{2}\log\log n - \log\log\log n = O\left((\log n)^{1/3}\right)}_{\text{taking log of both sides}} \equiv \log\log n = O\left((\log n)^{1/3}\right)$$

$$g_1 = O(g_3) \equiv 2^{(\log n)^{1/3}} = O\left(n^{5/3}\right) \underbrace{\equiv \log\log n = O\left(\log n\right)}_{\text{taking log of both sides}}$$

$$g_3 = O(g_7) \equiv n^{5/3} = O\left((\log n)^{\log n}\right) \underbrace{\equiv \log n = O\left(\log n \log\log n\right)}_{\text{taking log of both sides}}$$

$$g_7 = O(g_5) \equiv (\log n)^{\log n} = O\left(n^{(\log n)^2}\right) \underbrace{\equiv \log n \log\log n = O\left((\log n)^3\right)}_{\text{taking log of both sides}} \equiv \log\log n = O\left((\log n)^2\right)$$

$$g_5 = O(g_2) \equiv n^{(\log n)^2} = O\left(2^{n^2}\right) \underbrace{\equiv (\log n)^2 = O\left(n^2\right)}_{\text{taking log of both sides}}$$

$$g_2 = O(g_6) \equiv 2^{n^2} = O\left(2^{(11/9)^n}\right) \underbrace{\equiv n^2 = O\left((11/9)^n\right)}_{\text{taking log of both sides}}$$

Thus, the correct order is $g_6, g_2, g_5, g_7, g_3, g_1, g_4$.

(b)

$$f(n) = \begin{cases} n^2, & \text{if } n \text{ is an even positive integer} \\ n^4, & \text{if } n \text{ is an odd positive integer} \\ f(\lfloor n \rfloor) + \Big(f(\lceil n \rceil) - f(\lfloor n \rfloor)\Big)\Big(n - \lfloor n \rfloor\Big), & \text{otherwise} \end{cases}$$

$f(n)$ is a continuous function of n since the function $h(n) = f(\lfloor n \rfloor) + \Big(f(\lceil n \rceil) - f(\lfloor n \rfloor)\Big)\Big(n - \lfloor n \rfloor\Big)$ satisfy $h(\lfloor n \rfloor) = f(\lfloor n \rfloor)$ and $h(\lceil n \rceil) = f(\lceil n \rceil)$. The function $g(n)$ can similarly be defined as

$$g(n) = \begin{cases} n^4, & \text{if } n \text{ is an even positive integer} \\ n^2, & \text{if } n \text{ is an odd positive integer} \\ g(\lfloor n \rfloor) + \Big(g(\lceil n \rceil) - g(\lfloor n \rfloor)\Big)\Big(n - \lfloor n \rfloor\Big), & \text{otherwise} \end{cases}$$

and a similar argument shows that $g(n)$ is also a continuous function of n.

Proof of $g(n) \neq O\big(f(n)\big)$ and $f(n) \neq \Omega\big(g(n)\big)$: Assume, for the sake of contradiction, that $g(n) = O\big(f(n)\big)$. This implies that there exists two positive constants n_0 and c such that $g(n) \leq cf(n)$ for all $n \geq n_0$. Let $n_1 = \max\big\{2, 2\lceil c \rceil\big\} \times n_0 \geq n_0$. Note that n_1 is an even integer and thus,

$$g(n_1) \leq c\, f(n_1) \equiv n_1^4 \leq c\, n_1^2 \equiv n_1 \leq \sqrt{c}$$

and the last inequality above contradicts the choice of n_1. The same proof also shows that $f(n) \neq \Omega\big(g(n)\big)$.

Proof of $f(n) \neq O\big(g(n)\big)$ and $g(n) \neq \Omega\big(f(n)\big)$: Assume, for the sake of contradiction, that $f(n) = O\big(g(n)\big)$. This implies that there exists two positive constants n_0 and c such that $f(n) \leq cg(n)$ for all $n \geq n_0$. Let $n_1 = \max\big\{3, 2\lceil c \rceil + 1\big\} \times n_0 \geq n_0$. Note that n_1 is an odd integer and thus,

$$f(n_1) \leq c\, g(n_1) \equiv n_1^4 \leq c\, n_1^2 \equiv n_1 \leq \sqrt{c}$$

and the last inequality above contradicts the choice of n_1. The same proof also shows that $g(n) \neq \Omega\big(f(n)\big)$.

Problem 2: Suppose that there are only two classifications of professional wrestlers: the "good wrestlers" and the "bad wrestlers". Further suppose that two wrestlers can wrestle only if one of them good and the other one is bad.

Your input is a list of n wrestlers and a list of r pairs of wrestlers that have rivalries. Your goal is to write an algorithm to determine, in $O(n+r)$ time, if it is possible to classify the n wrestlers as good or bad such that the two wrestlers in every rivalry pair can in fact wrestle.

For example, if the wrestlers are W_1, W_2, W_3 and the rivalry pairs are $\{W_1, W_2\}$, $\{W_1, W_3\}$, then classifying W_1 as good and W_2, W_3 as bad works. If, one the other hand, the rivalry pairs are $\{W_1, W_2\}$, $\{W_3, W_2\}$, $\{W_1, W_3\}$, then no matter how we classify the wrestlers, one rivalry pair will contain two wrestlers of the same type.

Solution: Build the following undirected unweighted graph $G = (V, E)$. There is a node v_i in V corresponding to the i^{th} wrestler ($i = 1, 2, \ldots, n$). There is an edge $\{v_i, v_j\}$ in E if the i^{th} and the j^{th} wrestler are in a pair of rivalry. Clearly, G has n nodes, r edges and can be built in $O(n+r)$ time.

Classifying the wrestlers into two classes is equivalent to checking if G is bipartite or not (the "good" wrestlers form the left side of nodes, and the "bad" wrestlers the right side of nodes in the bipartite graph). In the class, we have seen how to do this using BFS in $O(n+r)$ time.