Homework 1

Simon Zheng
260744353

September 22nd, 2017

1 Stable marriage matching problem

a)

$$m_1 : w_1, w_2, w_3, w_4$$
$$m_2 : w_2, w_1, w_3, w_4$$
$$m_3 : w_3, w_2, w_1, w_4$$
$$m_4 : w_4, w_2, w_3, w_1$$
$$w_1 : w_4, w_2, w_3, w_1$$
$$w_2 : w_1, w_4, w_3, w_2$$
$$w_3 : w_1, w_2, w_4, w_3$$
$$w_4 : w_1, w_2, w_3, w_4$$

Simply assume each man has a different woman as first choice (and they are respectively the worst choice of the woman). Since the algorithm is ran on the men, each proposes to their first choice one after the other and the women must accept. The men also do not break up any pairings as they are all different. The algorithm terminates, so the rest doesn't even matter.

b) If every woman has their best choice, then there are two cases. One, every man had a different woman as their first choice, and they are respectively also the first choice of their woman. Second case, every man must have proposed to every woman to all get their last choices. So if we jump to the last man at his last choice where he proposes to a woman, this woman breaks up (as he must be her first choice to end up like we want) with her previous choice. But then this means a man is now free, despite the men having gone through all their choices. The algorithm was proven to never be able to arrive at this state. Therefore, this is not a valid final state if we use this algorithm.

c)

$$\begin{aligned} m_1 &: m_2, m_3, m_4 \\ m_2 &: m_3, m_4, m_1 \\ m_3 &: m_4, m_2, m_1 \\ m_4 &: m_1, m_2, m_3 \end{aligned}$$

While the algorithm terminates, it is not a stable matching.

$$\begin{aligned} m_1 &: \cancel{m_2}, \underline{m_3}, \cancel{m_4} \\ m_2 &: \cancel{m_3}, \underline{m_4}, \cancel{m_1} \\ m_3 &: \cancel{m_4}, \cancel{m_2}, \underline{m_1} \\ m_4 &: \cancel{m_1}, \underline{m_2}, \cancel{m_3} \end{aligned}$$

m_2 prefers m_3 and vice-versa, yet they are not matched together.

2

a)

$$\begin{aligned} \sqrt{n} + n\sqrt{n} &= \sqrt{n}(1+n) \\ &\leq n(1+n) \\ &\leq n(n+n) = n^2 + n^2 = 2n^2 \end{aligned}$$

Thus, for $c = 2$ and $n_0 = 3$.

b) On one hand,

$$\begin{aligned} (n + \log_2 n)^5 \geq n^5 + (\log_2 n)^5 &\geq n^5 \\ n^5 &= O((n + \log_2 n)^5) \end{aligned}$$

For $c = 1, n_0 = 1$. On the other hand,

$$\begin{aligned} (n + \log_2 n)^5 \leq (n+n)^5 &= 2^5 n^5 \\ (n + \log_2 n)^5 &= O(n^5) \end{aligned}$$

For $c = 2^5, n_0 = 1$.

c)

$$\begin{aligned} n! &= 1 \times 2 \times ... \times (n-1) \times n \\ n^n &= n \times n \times ...n\ times... \times n \times n \end{aligned}$$

So,

$$\begin{aligned}\lim_{n\to\infty}\frac{n!}{n^n} &= \lim_{n\to\infty}\frac{1\times 2\times ...\times (n-1)\times n}{n\times n\times ...\times n \qquad \times n}\\ &= 0\end{aligned}$$

Therefore, $n! = o(n^n)$.

d) Let us use de L'Hôpital's rule.

$$\begin{aligned}\lim_{n\to\infty}\frac{\log_2 n}{n^{1/100}} &= \lim_{n\to\infty}\frac{1/n}{1/(100n^{99/100})}\\ &= \lim_{n\to\infty}\frac{100n^{99/100}}{n}\\ &= 100\lim_{n\to\infty}\frac{1}{n^{1/100}}\\ &= 0\end{aligned}$$

3

a) This is false, as $2^{2^n} = o(2^{2^{n+1}})$:

$$\begin{aligned}&\lim_{n\to\infty}\frac{2^{2^n}}{2^{2^{n+1}}}\\ &= \lim_{n\to\infty}\frac{2^{2^n}}{2^{2^n\cdot 2}}\\ &= \lim_{n\to\infty}\frac{2\times ...(2^n\ times)...\times 2}{2\times ...(2\cdot 2^n\ times)...\times 2}\\ &= \lim_{n\to\infty}\frac{1}{2^{2^n}}\\ &= 0\end{aligned}$$

for $n_0 = 0, c = 1$.

b) By using limits and little-oh to disprove this,

$$\begin{aligned}\lim_{n\to\infty}\frac{\log_2 n^5}{(\log_2 n)^5} &= \lim_{n\to\infty}\frac{5\log_2 n}{(\log_2 n)^5}\\ &= 5\lim_{n\to\infty}\frac{1}{(\log_2 n)^4}\\ &= 0\end{aligned}$$

Thus, if $\log_2 n^5$ is $o((\log_2 n)^5)$, then $(\log_2 n)^5$ cannot be $O(\log_2 n^5)$.

c)

d)

4

a) Big-Oh of a sum of function is the same as big-Oh of the fastest growing function. We can see $O(f(n))$ as the set of all functions that are upper-bounded by $f(n)$. $O(f(n)+g(n)) = O(max(f(n), g(n)))$ So on one hand if $f(n) > g(n)$ (as in grows faster) then

$$\begin{aligned} O(f(n)+g(n)) &= O(f(n)) \\ O(f(n)) &= f(n) + O(g(n)) \\ &= f(n) \end{aligned}$$

as $f(n) \geq g(n)$ so any function that is $O(g(n))$ will still be dominated by the faster growing $f(n)$.

On the other hand, if $g(n)$ dominates over $f(n)$, then

$$\begin{aligned} O(f(n)+g(n)) &= O(g(n)) \\ O(g(n)) &= f(n) + O(g(n)) \\ &= O(g(n)) \end{aligned}$$

as since $g(n)$ grows faster than $f(n)$, there will be at least one function $O(g(n))$ that exists that grows faster than $f(n)$ thus $f(n)$ doesn't matter and is dominated by the set of functions $O(g(n))$.

b) For the product, constants (adding to or multiplying the function) are ignored as they don't change the growth order of the function, but multiplying functions do. Any function $O(g(n))$ does not grow faster than $g(n)$ thus $f(n) \times O(g(n))$ will never grow faster than $f(n) \times g(n)$. Therefore, $f(n) \times O(g(n)) = O(f(n) \times g(n)$.

5

From the base case, we can see that it should apply for any $c \geq 1$.

$$\begin{aligned} n^2 &= O(n) \\ n^2 &\leq c \cdot n \\ 1 &\leq c \cdot 1 \end{aligned}$$

so we set $n_0 = 1$.

But in the induction step, c isn't taken into account, and it breaks down even for the simple case of $c = 1$:

$$\begin{aligned}(n+1)^2 = n^2 + 2n + 1 = O(n) + 2n + 1 &= O(n+1)\\ c \cdot n + 2n + 1 &= c \cdot (n+1)\\ n + 2n + 1 &= n + 1 && (c = 1)\\ &\textit{Contradiction.}\end{aligned}$$

Therefore, the induction step is flawed, as it did not take the correct definition of big-oh into account. Since the base case sets $c = 1$ or $c \geq 1$, then it is given that this must apply to the induction step which depends on this base case, yet we arrive at a contradiction.