CSE 551
Assignment 1

30^{th} August, 2022

Submission Instructions: Deadline is **11:59pm on 09/07/2022**. Late submissions will be penalized, therefore please ensure that you submit (file upload is completed) before the deadline. Additionally, you can download the submitted file to verify if the file was uploaded correctly. **Please TYPE UP YOUR SOLUTIONS and submit a PDF** electronically, via *Canvas*. Furthermore, please note that the graders will grade 2 out of the 4 questions randomly. Therefore, if the grader decides to check questions 1 and 4,and you haven't answered question 4, you'll lose points for question 4. Hence, please answer all the questions.

1. Prove or disprove the following assertions: **(8+8+9)**

   (i) If $f(n) = O(g(n))$ then $log_2 f(n) = O(log_2 g(n))$
   True

   Given $f(n) \leq c.g(n)$ for all $n \geq n0$ and c is a constant.
   Applying log_2 on both sides.
   $log_2 f(n) \leq log_2 c + log_2 g(n)$ for all $n \geq n0$
   Choose n0 such that $log_2 g(n) \geq log_2 c$ for all $n \geq n0$
   From above 2 we can say $log_2 f(n) \leq log_2 g(n) + log_2 g(n) = 2.log_2 g(n)$
   Hence we can say $log_2 f(n) = O(log_2 g(n))$.

   (ii) If $f(n) = O(g(n))$ then $3^{f(n)} = O(3^{g(n)})$
   False

   Here is a counter example. Consider $f(n) = 2log_3 n$ and $g(n) = log_3 n$. This satisfies $f(n) = O(g(n))$

   $3^{f(n)} = n^2$ and $3^{g(n)} = n$ but n^2 is not $O(n)$.

   (iii) If $f(n) = O(g(n))$ then $f(n)^3 = O(g(n)^3)$
   True

   Proof simiar to (i) where we instead cube on both sides.

2. Algorithm A_1 takes $10^{-4} \times 2^n$ seconds to solve a problem instance of size n and Algorithm A_2 takes $10^{-2} \times n^3$ seconds to do the same on a particular machine. **(8+8+9)**
(i) What is the size of the largest problem instance A_2 will be able solve in one year ?
Ans: 1 year = 31536000 seconds. Assume we can solve a problem size x. Then $10^{-2} * x^3 = 31536000$. Solving it we get x = 1466.

(ii) What is the size of the largest problem instance A_2 will be able solve in one year on a machine one hundred times as fast ?
Ans: From given info we can say A2 is O(n^3) algorithm. From the size of largest problem instance solvable in 1 Hour table we filled in class A2 can solve 4.64N size problem in 1hour on a machine 100 times faster. 4.64 * 1466.64 is approximately 6805.

(iii) Which algorithm will produce results faster, in case we are trying to solve problem instances of size less than 20 ?
Ans: For $n = 20$ A1 takes $10^{-4} * 2^{20} = 104.856$ seconds and A2 takes $10^{-2} * n^3 = 80$ seconds. For $n = 19$ A1 takes 52.43 seconds and A2 takes 68.59 seconds. So both the plots intersect somewhere between 19 and 20. So for problem sizes less than 20 A1 solves the problem faster.

3. Prove or disprove the following with valid arguments: **(8+8+9)**

(i) $3n^3 + 1000 = O(n^2)$.

Ans: False

R.T.P. $3n^3 + 1000 = O(n^2)$

or, $3n^3 + 1000 \leq c.n^2$ where $n \geq n0$

or, $3n + \frac{1000}{n} \leq c$

or, $\frac{3n^2+1000}{n} \leq c$

The L.H.S. is a function of n and its value increases with n. There are no c value for which the above condition is true.

(ii) $2n^2 log(n) = \Theta(n^2)$.

Ans: False

R.T.P. $n^2.log(n) = \Theta(n^2)$

Alternatively, we need to prove :

Case 1 : $n^2.log(n) = O(n^2)$ and

Case 2 : $n^2.log(n) = \Omega(n^2)$

For Case 1 :

R.T.P. : $n^2.log(n) \leq c_1.n^2$

Dividing both L.H.S and R.H.S by n^2 we get :

R.T.P. : $log(n) \leq c_1$

The L.H.S. is a function of n and its value increases with n. There are no c_1 value for which the above condition is true for sufficiently large value of n.

(iii) $3^n n^4 + 8 * 4^n n^3 = O(3^n n^4)$.

ANS: False

R.T.P. $3^n n^4 + 8 * 4^n n^3 = O(3^n n^4)$.

or, $3^n n^4 + 8 * 4^n n^3 \leq c.3^n n^4 foralln \geq n0$

Dividing both sides by $3^n n^4$

$1 + \frac{8*4^n n^3}{3^n n^4} \leq c$

$1 + \frac{8*(1.3)^n}{n} \leq c$

Here, we have reached a contradiction as the L.H.S. becomes infinity as n approaches ∞. So no such constant c exists.

4. Take the following list of functions and arrange them in ascending order of growth rate. That is, if function g(n) immediately follows f(n) in your list, then it should be the case that f(n) is O(g(n)). **(25)**

(i) $f_1(n) = n^{4.6}$.
(ii) $f_2(n) = (2n)^{1.6}$.
(iii) $f_3(n) = n^{4.5} + 87$.
(iv) $f_4(n) = 40^n$.
(v) $f_5(n) = 120^n$

ANS: $f_2 < f_3 < f_1 < f_4 < f_5$ Polynomial functions have less order of growth compared to exponential functions. Polynomial functions have order of growth in the order of their highest exponents. In exponential functions 40^n has lower order of growth than 120^n.