Instructor Name: Siddhartha Nath
Email: s.nath@northeastern.edu

Program Structure and Algorithms (INFO 6205)
Homework #1 – 100 points

Student NAME: Xijing Zhang

Student ID: 002205911

Notes:

- Please submit two files.
- The first file MUST be a PDF that contains your solutions to all questions except the coding question.
- The second file is your solution to the coding question with either .py or .cpp or .java extension.

Question 1 (25 points). *Please prove the following with regards to asymptotic growth of functions.*

(a) *(5 points) Show that $f(x) = x^2 + 4x$ is $O(x^2)$.*

Proof:

A function $f(x)$ is $O(g(x))$ if and only if there exist positive real numbers c and x_0 such that:

$$0 \leq f(x) \leq c \cdot g(x) \quad \textit{for every} \quad x \geq x_0$$

Consider $c = 2$ and $x_0 = 4$. For any $x \geq x_0$, note that $4x \leq x^2$. Adding x^2 to both sides, we have:

$$x^2 + 4x \leq x^2 + x^2 = 2x^2.$$

Since $x \geq x_0 = 4$, it follows that:

$$0 \leq x^2 + 4x \leq 2x^2 = c \cdot x^2.$$

Thus, $f(x) = x^2 + 4x$ satisfies the definition of Big-O with $c = 2$ and $x_0 = 4$. Therefore:

$$f(x) = O(x^2).$$

(b) *(5 points) Show that $f(x) = x^2$ is NOT $O(\sqrt{x})$.*

Proof:

By the definition of Big-O, $f(x) = x^2$ would be $O(\sqrt{x})$ if there exist constants $c > 0$ and $x_0 > 0$ such that:

$$0 \leq f(x) \leq c \cdot \sqrt{x} \quad \textit{for all} \quad x \geq x_0.$$

Substituting $f(x) = x^2$ into the inequality, this implies:

$$x^2 \leq c \cdot \sqrt{x}.$$

Dividing through by $\sqrt{x}$ (valid for $x > 0$) gives:

$$x^{3/2} \leq c.$$

As $x \to \infty$, $x^{3/2} \to \infty$, which means the inequality cannot hold for any constant $c > 0$. Therefore, there are no constants $c > 0$ and $x_0 > 0$ that satisfy the definition of Big-O.

*Hence, $f(x) = x^2$ is **not** $O(\sqrt{x})$.*

(c) *(5 points) Show that $f(x) = x$ is $\Omega(\log x)$.*

Proof:

For a given function $g(n)$, we denote by $\Omega(g(n))$ (pronounced "big-Omega of $g(n)$") the set of functions:

$$\Omega(g(x)) = \{f(x) : \textit{there exist positive constants } c \textit{ and } x_0 \textit{ such that } 0 \leq c \cdot g(x) \leq f(x) \textit{ for all } x \geq x_0\}.$$

Consider $c = 1$ and $x_0 = 2$. Then, for any $x \geq x_0$, we need to show that:

$$x \geq c \cdot \log x.$$

For $c = 1$, this reduces to:

$$x \geq \log x \quad \textit{for all} \quad x \geq 2.$$

Since x grows faster than $\log x$ asymptotically as $x \to \infty$, this inequality holds true. In fact, for any $x \geq 2$, we can confirm that:

$$x \geq \log x.$$

Therefore, for $c = 1$ and $x_0 = 2$, we have:

$$x \geq \log x \quad \textit{for all} \quad x \geq 2.$$

Thus, $f(x) = x$ satisfies the definition of $\Omega(\log x)$.

(d) *(10 points) Show that $f(x) = (2x^2 - 3)/((3x^4 + x^3 - 2x^2 - 1)$ is $\Theta(x^{-2})$.*

Proof:

To prove that $f(x) = \Theta(x^{-2})$, we need to show that there exist positive constants c_1, c_2, and x_0 such that:

$$c_1 x^{-2} \leq f(x) \leq c_2 x^{-2}, \quad \forall x \geq x_0.$$

For large x, the highest-order terms dominate. Thus:

$$f(x) = \frac{2x^2 - 3}{3x^4 + x^3 - 2x^2 - 1} \approx \frac{2x^2}{3x^4}.$$

Simplifying the dominant terms, we have:

$$\frac{2x^2}{3x^4} = \frac{2}{3x^2}.$$

This shows that $f(x)$ asymptotically behaves like $\frac{2}{3x^2}$, which suggests that $f(x) \in \Theta(x^{-2})$. Now, we formalize the bounds.
Derive upper and lower bounds, we start with the expression:

$$f(x) = \frac{2x^2 - 3}{3x^4 + x^3 - 2x^2 - 1}.$$

For sufficiently large x, we observe that:

$$2x^2 - 3 \geq x^2, \quad \textit{and} \quad 3x^4 + x^3 - 2x^2 - 1 \leq 4x^4.$$

Thus:

$$f(x) \geq \frac{x^2}{4x^4} = \frac{1}{4x^2}.$$

For sufficiently large x, we observe that:

$$2x^2 - 3 \leq 2x^2, \quad \textit{and} \quad 3x^4 + x^3 - 2x^2 - 1 \geq 3x^4.$$

Thus:

$$f(x) \leq \frac{2x^2}{3x^4} = \frac{2}{3x^2}.$$

From the above, for sufficiently large x, we have:

$$\frac{1}{4x^2} \leq f(x) \leq \frac{2}{3x^2}.$$

This satisfies the definition of $\Theta(x^{-2})$, where $c_1 = \frac{1}{4}$, $c_2 = \frac{2}{3}$, and x_0 is sufficiently large.
We have shown that:

$$f(x) = \frac{2x^2 - 3}{3x^4 + x^3 - 2x^2 - 1} \in \Theta(x^{-2}).$$

Question 2 (15 points)**.** *Please rank the following functions based on their $O(\cdot)$ complexity of running time. The function that has the least complexity should be ranked 1. Please explain your answer to get full credit.*

$f_1(x) = x \log_2 x$
$f_2(x) = 3^x$
$f_3(x) = \sqrt{x}$
$f_4(x) = x!$
$f_5(x) = 2^x$

Solution:

We rank the functions based on their growth rates (smaller growth rate = higher rank):

1. $f_3(x) = \sqrt{x}$: $O(\sqrt{x})$ (slowest growth rate)
2. $f_1(x) = x \log_2 x$: $O(x \log x)$ (quasilinear)
3. $f_5(x) = 2^x$: $O(2^x)$ (exponential with base 2)
4. $f_2(x) = 3^x$: $O(3^x)$ (exponential with base 3)
5. $f_4(x) = x!$: $O(x!)$ (super-exponential, fastest growth rate)

Explanation:

1. $f_1(x) = x \log_2 x$: This function grows faster than x (linear growth) but slower than polynomial functions like x^2. It is classified as **quasilinear**.

$$O(f_1(x)) = O(x \log x)$$

2. $f_2(x) = 3^x$: This function is **exponential** with base 3. It grows faster than polynomial functions and 2^x.

$$O(f_2(x)) = O(3^x)$$

3. $f_3(x) = \sqrt{x}$: This function is **sublinear** (grows slower than linear functions x).

$$O(f_3(x)) = O(\sqrt{x})$$

4. $f_4(x) = x!$: The factorial function grows faster than any exponential function. It is **super-exponential**.

$$O(f_4(x)) = O(x!)$$

5. $f_5(x) = 2^x$: This function is **exponential** with base 2. It grows slower than 3^x but faster than polynomial or quasilinear functions.

$$O(f_5(x)) = O(2^x)$$

Question 3 (60 points). *Suppose you are given a string consisting of alphanumeric and parenthesis characters as input. Your goal is to determine if all the open-parenthesis have a corresponding close-parenthesis when you reach the end of the string. If yes, then your algorithm should return True, else False.*

For example, if the input is "I { love [the {rains}()]}", then the output is True. Whereas, if the input is "I { love [the {rains] ()", then the output is False.

(a) *(15 points) Please describe an efficient algorithm in English using a data structure such as array / linked list / stack / queue to solve this problem.*

(b) *(5 points) What is the asymptotic upper bound of complexity of running time for your algorithm?*

(c) *(40 points) Please write a program in either Python / Java / C++ that realizes your algorithm in (a). To receive full credit, please structure your code, write comments and show the output for the above two examples.*

(a)
1. Initialize an empty stack.
2. Traverse each character in the input string.
a. If it is an opening parenthesis ('(', '', or '['), push it onto the stack.
b. If it is a closing parenthesis (')', '', or ']'):
i. Check if the stack is empty. If it is, return False.
ii. If the stack is not empty, pop the top element and check if it matches the corresponding opening parenthesis.
3. After processing all characters, if the stack is empty, return True. Otherwise, return False.

(b)
The algorithm processes each character in the input string exactly once, and each stack operation (push and pop) takes constant time $O(1)$. Therefore, the time complexity is:

$$O(n)$$

where n is the length of the input string.

(c)
HW1.py