Assignment: Asymptotic Notations and Correctness of Algorithms

1. **Identify and compare the order of growth**: Use the definitions of O, Ω and Θ to identify if the following statements are true or false. Prove your assertion. Draw a rough graph marking c and n_0 values if the statement is True. [You don't have to find the values of c and n_0].

   a. n(n+1)/2 ∈ O(n^3) True

   $$\lim_{n\to\infty} \frac{\frac{n(n+1)}{2}}{n^3}$$

   $$= \lim_{n\to\infty} \frac{n^2+n}{2n^3}$$

   $$= \lim_{n\to\infty} \frac{2n+1}{6n^2}$$ Applying L'Hospital rule

   $$= \lim_{n\to\infty} \frac{2}{12n}$$ Applying L'Hospital rule

   $$= 0$$

   ⇒ f(n) ∈ O (g(n))

   b. n(n+1)/2 ∈ Θ(n^2) True

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

   $$= \lim_{n\to\infty} \frac{n^2+n}{2n^2}$$

   $$= \lim_{n\to\infty} \frac{2n+1}{4n}$$ Applying L'Hospital rule

   $$= \lim_{n\to\infty} \frac{2}{4}$$ Applying L'Hospital rule

   = Constant

   ⇒ f(n) ∈ Θ (g(n))

   c. 10n-6 ∈ Ω(78n + 2020) True

   $$\lim_{n\to\infty} \frac{10n-6}{78n+2020}$$

   $$= \lim_{n\to\infty} \frac{10}{78}$$ Applying L'Hospital rule

= Constant

$\Rightarrow$ f(n) $\in \Theta$ (g(n))

Since Θ is stronger notation than Ω-notation. Θ-notation implies Ω-notation. Hence, True

d. n! $\in \Omega$ (0.00001n) True

$$\lim_{n\to\infty} \frac{n!}{0.00001n}$$

$$= \lim_{n\to\infty} \frac{\sqrt{2\pi n}\left(\frac{n}{e}\right)^n}{0.00001n} \qquad \text{using Stirling's Formula}$$

$$= \frac{\sqrt{2\Pi}}{0.00001} \lim_{n\to\infty} \frac{\sqrt{n}}{n} \left(\frac{n}{e}\right)^n$$

$$= \frac{\sqrt{2\pi}}{0.0000^1} \lim_{n\to\infty} \frac{\sqrt{n}}{(e)^n} x \frac{n^n}{n}$$

$$= \frac{\sqrt{2\pi}}{0.00001} \lim_{n\to\infty} \frac{\sqrt{n}}{e^n} x^{n^{n-1}}$$

$$= (something) * (\infty)^\infty = \infty$$

$\Rightarrow$ f(n) $\in \Omega$ (g(n))

Or

$$\lim_{n\to\infty} \frac{n!}{0.00001n}$$

$$= \lim_{n\to\infty} \frac{n(n-1)(n-2)\ldots.}{0.00001n}$$

$$= \lim_{n\to\infty} \frac{(n-1)(n-2)\ldots.}{0.00001} \quad \text{[simplifying]}$$

$$= \infty$$

2. **Read and Analyze Pseudocode:** Consider the following algorithm

```
Classified(A...n-1):
    minval = A[0]
    maxval = A[0]
    for i = 1 to n-1:
        if A[i] < minval:
            minval = A[i]
```

```
        if A[i] > maxval
            maxval = A[i]
    return maxval - minval
```

a. What does this algorithm compute? The difference between the largest and the smallest element in an array.

b. What is its basic operation (i.e. the line of code or operation that is executed maximum number of times)?
   The if conditions
   `if A[i] < minval; if A[i] > maxval`

c. How many times is the basic operation executed?
   $\sum_{i=1}^{n-1} 1 = (n-1)$

d. What is the time complexity of this algorithm?
   $\theta(\mathrm{n})$

3. **Using mathematical induction prove below non-recursive algorithm:**

```
def reverse_array(Arr):
    n = len(Arr)
    i = (n-1)//2
    j = n//2
    while(i>= 0 and j <= (n-1)):
        temp = Arr[i]
        Arr[i] = Arr[j]
        Arr[j] = temp
        i = i-1
        j = j+1
```

a. Write the loop invariant of the reverse_array function.
   Before the iteration of the while loop, for values i and j, Arr[i+1 : j-1] is reversed. That is values in the Arr from 'i+1' to 'j-1' are reversed.

b. Prove correctness of reverse_array function using induction.

   Loop invariant: For i and j, Arr[i+1 : j-1] is reversed

   Initialization: Before the entry into the while loop j-1 < i+1, hence the part of the Arr[i+1 : j-1] does not contain any elements. The invariant is true by default.

   Maintenance: At the start of k^{th} iteration, assume that Arr[i+1 : j-1] is reversed.

   In in the previous iteration i is decremented and j is incremented. In the k^{th} iteration, in the body of the loop Arr[i] and Arr[j] are swapped.

   Thus at the start of $(k+1)^{th}$ iteration Arr[i : j] is reversed.

   Termination: After the loop terminates, i = -1 and j = n.

   Based on loop invariant Arr[i+1 : j-1] is reversed. Therefore, Arr[0:n-1], which is our entire array, is reversed.