Assignment: Asymptotic Notations and Correctness of Algorithms

[You may include handwritten submission for the parts of the assignment that are difficult to type, like equations, rough graphs etc., but make sure it is legible for the graders. Regrade requests due to the illegible parts of the work will not be accepted.]

1. **Identify and compare the order of growth**: Identify if the following statements are true or false. Prove your assertion using any of the methods shown in the exploration. Draw a rough graph marking the location of c and n_0, if the statement is True. [A generic graph would do for this purpose. You **don't** have to find the values of c and n_0. On the graph you can just write c and n_0 without mentioning their values. The idea is that you know how it looks graphically.].
   a. $n(n+1)/2 \in O(n^3)$
      i False

      1) a) $n(n+1)/2 \in O(n^3)$

      For all $n \geq 1$: $n^2 \leq n^3$ (False statement)

      For all $n \geq 1$: $\frac{n^2}{2} \leq \frac{n^3}{2}$ (divide by 2)

      For all $n \geq 1$: $\frac{n^2}{2} + \frac{n}{2} \leq \frac{n^3}{2} + \frac{n}{2}$ (add $n/2$)

      If we choose $c = \frac{1}{2}$ then:

      $\frac{n^2}{2} + \frac{n}{2} \leq c(n^3+n)$

      ~~Therefore, $n(n+1)/2 \notin$~~

      therefore, $n(n+1)/2 \in O(n^3)$ is false,

      ii n
   b. $n(n+1)/2 \in \Theta(n^2)$
      i True

ii

b) $n(n+1)/2 \in O(n^2)$

$n \leq n^2$ for all $n \leq 1$

$\frac{1}{2} n \leq \frac{1}{2} n^2$ for all $n \leq 1$ (multiply by $\frac{1}{2}$)

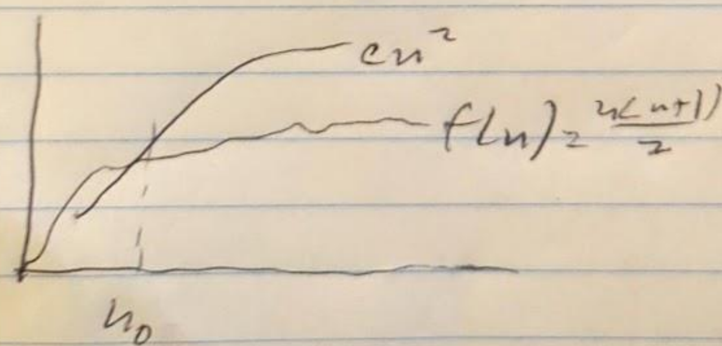
$\frac{1}{2} n^2 + \frac{1}{2} n \leq \frac{1}{2} n^2 + \frac{1}{2} n^2$ for all $n \leq 1$

(add $\frac{1}{2} n^2$)

Hence we can choose $c = 1$,

$n(n+1)/2 \leq cn^2$

$n(n+1)/2 \in O(n^2)$



c. 10n-6 $\in \Omega$(78n + 2020)

i False

ii

c) $10n-6 \in \Omega(78n+2020)$

$10n-6 \geq 78n + (-68n - 2014) + 2020$

$10n-6 \geq 78n+2020$ (false statem

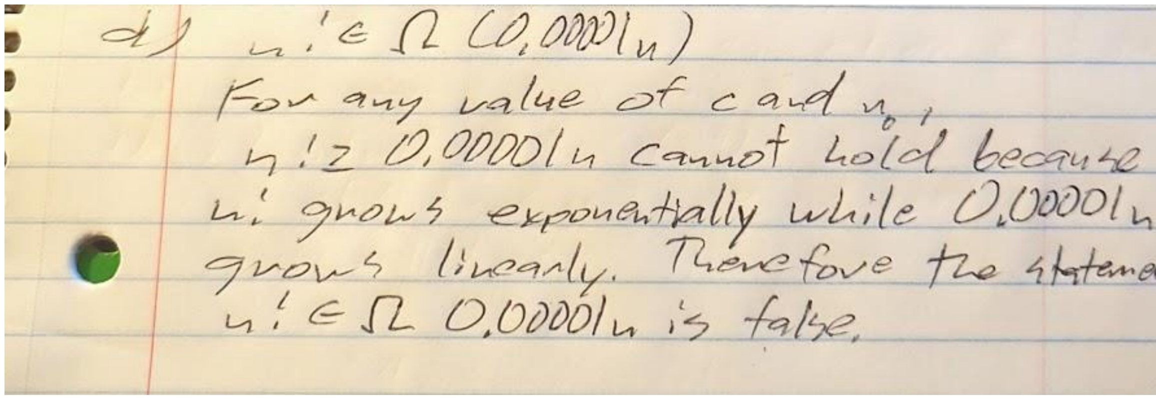
For all terms, regardless of the values

c and n_0, $10n-6$ is not greater than

$c \cdot 78n$.

d. n! ∈ Ω (0.00001n)

i False

ii 

d) $n! \in \Omega$ (0.00001n)
For any value of c and n_0,
$n! \geq 0.00001n$ cannot hold because
n! grows exponentially while 0.00001n
grows linearly. Therefore the statem
$n! \in \Omega$ 0.00001n is false.

2. **Read and Analyze Pseudocode:** Consider the following algorithm (In the algorithm, A[0..n-1] refers to an array of n elements i.e. A[0], A[1]… A[n-1])

```
Classified(A[0..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?
   i This algorithm finds the minimum and maximum values in an array.
b. What is its basic operation (i.e. the line of code or operation that is executed maximum number of times)?
   i If A[i] < minval and if A[i] > maxval
c. How many times is the basic operation executed?
   i The basic operation is executed n-1 times, where n is the number of elements in the array.
d. What is the time complexity of this algorithm?
   i O(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.
   i At the start of each iteration of the while loop, the subarray Arr[0:i+1] is reversed.
b. Prove correctness of reverse_array function using induction.
   i Loop invariant: At the start of each iteration of the while loop, the subarray Arr[0:i+1] is reversed.
   ii Initialization: At the start of the first iteration the loop invariant states: At the start of each iteration of the while loop, the subarray Arr is reversed. i is set to (n-1)//2 and j is set to n//2.
   iii Maintenance: Assuming at the start of iteration (i>= 0 and j <= (n-1)) holds true, temp holds value Arr[i] and the algorithm performs the element swapping operation Arr[i] = Arr[j] and Arr[j] = temp. After the swap, the subarray Arr[0:i+1] is still reversed because the element at index i is now swapped with the element at index j, maintaining the reversed order.
   iv Termination: The loop terminates when the condition i >= 0 and j <= (n-1) becomes false, which means that i becomes negative or j exceeds the array bounds.

----------------------(Ungraded question: you can try this question if time permits)---------------

Any number greater than 8 can written in terms of three or five.

a. Write a pseudocode of algorithm that that takes a number greater than 8 and returns a tuple (x,y); where x represents number of threes and y represent number of fives make that number
   If number = 8 your pseudocode should return (1,1)
b. Code your pseudocode into python and name your file ThreeAndFive.py