1. True or False? (you must justify your claim using a limit argument, induction or some other method.)

   (a) $5(3^n) \in O(3(5^n))$

   Solution: True

   $$\lim_{n\to\infty} \frac{5(3^n)}{3(5^n)} = \frac{5}{3} \lim_{n\to\infty} \left(\frac{3}{5}\right)^n = 0$$

   (b) $(n^6 + 2n + 1)^2 \in O((3n^3 + 4n^2)^4)$

   Solution: True

   $$\lim_{n\to\infty} \frac{(n^6 + 2n + 1)^2}{(3n^3 + 4n^2)^4} = \lim_{n\to\infty} \frac{n^{12}}{3n^{12}} = \frac{1}{3}$$

   (c) $\log(n^{10}) \in \Omega(\log(n))$

   Solution: True

   $$\lim_{n\to\infty} \frac{\log(n^{10})}{\log(n)} = \lim_{n\to\infty} \frac{10\log(n)}{\log(n)} = 10$$

   (d) $\sum_{i=1}^{n} i^k \in \Omega(n^{k+1})$ for any fixed positive integer k.

   Solution: True

   By integral calculus, we have that for any $n > 1$,

   $$\begin{aligned} \sum_{i=1}^{n} i^k &> \int_0^n x^k dx \\ \sum_{i=1}^{n} i^k &> \left.\frac{x^{k+1}}{k+1}\right|_0^n \\ \sum_{i=1}^{n} i^k &> \frac{n^{k+1}}{k+1} = \left(\frac{1}{k+1}\right) n^{k+1} \end{aligned}$$

   The inequality shows that $\sum_{i=1}^{n} i^k > \left(\frac{1}{k+1}\right) n^{k+1}$ so $\sum_{i=1}^{n} i^k = \Omega\left(n^{k+1}\right)$

   (e) $\log n! \in \Omega(n \log n)$

   Solution:

   We can rewrite $\log n! = \sum_{i=1}^{n} \log(i)$. Clearly, this summation is greater than the only the last half of its terms:

   $$\log n! = \sum_{i=1}^{n} \log(i) > \sum_{i=n/2}^{n} \log(i).$$

   Then each term in this sum must be greater than $\log(n/2)$, so:

   $$\log n! > \sum_{i=n/2}^{n} \log(n/2) = n/2 \log(n/2).$$

   We can rewrite this as:

   $$n/2 \log(n/2) = n/2\,(\log n - \log 2) = \Omega(n \log n)$$

   and since $\log n! > n/2 \log(n/2)$, then $\log n! = \Omega(n \log n)$.

2. The following sequence of numbers: $T_0, T_1, \ldots$, are defined by

   $$T_0 = 1, T_1 = 1, \qquad T_n = 2T_{n-1} + 2T_{n-2}$$

   Prove that:

(a) $T_n = O(3^n)$

(b) $T_n = \Omega\left(2^n\right)$

Solution:

(a) Claim: $T_n \leq 3^n$ for all $n \geq 0$.
$T_0 = 1, 3^0 = 1$
$T_1 = 1, 3^1 = 3$
Suppose that for some arbitrary integer $n \geq 1$, that $T_k \leq 3^k$ for all $0 \leq k < n$.
Then:

$$\begin{aligned} T_n &= 2T_{n-1} + 2T_{n-2} \\ &\leq 2\left(3^{n-1} + 3^{n-2}\right) \\ &\leq 2\left(3^{n-2}\right)(3+1) \\ &= 8\left(3^{n-2}\right) \\ &\leq 9\left(3^{n-2}\right) \\ &= 3^n \end{aligned}$$

(b) Claim: $T_n \geq 2^n$ for all $n \geq 2$.
$T_2 = 4, 2^2 = 4$
$T_3 = 10, 2^3 = 8$
Suppose that for some arbitrary integer $n \geq XXXX$, that $T_k \geq 2^k$ for all $0 \leq k < n$.
Then:

$$\begin{aligned} T_n &= 2T_{n-1} + 2T_{n-2} \\ &\geq 2\left(2^{n-1} + 2^{n-2}\right) \\ &\geq 2\left(2^{n-2} + 2^{n-2}\right) \\ &= 4\left(2^{n-2}\right) \\ &= 2^n \end{aligned}$$

3. The indegree of a vertex u is the number of incoming edges into u, i.e., edges of the form (v, u) for some vertex v.

Consider the following algorithm that takes the adjacency list $A[v_1, v_2, \ldots, v_n]$ of a directed graph G as input and outputs an array containing all indegrees.
(An adjacency list $A[v_1, v_2, \ldots, v_n]$ is an array indexed by the vertices in the graph. Each entry $A[v_i]$ contains the list of neighbors of v_i.)

```
procedure Indegree(A[v_1, v_2, ..., v_n])
  initialize an array ID[v_1, ..., v_n] to 0
  for each i = 1 ... n:
    for each u ∈ A[v_i]:
      increment ID[u]
  return ID
```

(a) Prove the correctness of this algorithm by proving the loop invariant:

At tth iteration, $ID[u]$ is the number of incoming edges to u from $\{v_1, \ldots, v_t\}$.

Base Case: Before the first iteration ($t = 0$), the array ID is initialized to 0 which is the indegree to all vertices from the empty set.
Inductive Hypothesis: Suppose after $t-1$ iterations for some t with $t > 0$ that $ID[u]$ is equal to the number of incoming edges to u from $\{v_1, \ldots, v_{t-1}\}$. Show what happens after t iterations.

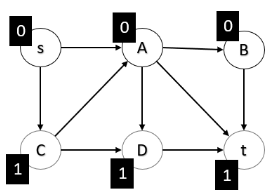
Inductive Step: During the t iteration, $i = t$, so the inner for loop considers all outgoing neighbors of v_t. The array value $ID[u]$ of each outgoing neighbor u of v_t will be incremented during this iteration. So along with the loop invariant being true after $t-1$ iterations, it will be true that $ID[u]$ is equal to the number of incoming edges to u from $\{v_1, \ldots, v_t\}$ so the loop invariant remains true.

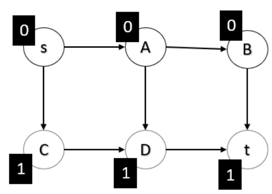
(b) Analyze the runtime of this algorithm assuming that G has n vertices and m edges.

Inside each outer iteration, the algorithm loops through each neighbor of v_i which is equal to $outdegree(v_i)$ iterations of the inner loop. So, over the course of the entire algorithm, there are $\sum_{v \in V} outdegree(v) = m$ many times we increment $ID(u)$. Along with the initialization of the array which takes $O(n)$ time, the entire algorithm will take $O(n+m)$ time.

4. Consider the following problem:
Given a directed graph G with vertex weights $w_v \in \{0, 1\}$ (in other words, each vertex is either labeled with 0 or 1), and vertices s and t, Determine if there is a (not necessarily simple) path in G from s to t such that the binary sequence of vertex weights in the path is of the form $(01)^i$ for any $i \geq 1$. (I believe if any path exists, a simple path will exist as well.)

For example, the algorithm should output TRUE for the first graph because of the path $s \to C \to A \to t$. The algorithm should output FALSE for the second graph because there is no path from s to t with a sequence of $(01)^i$.





Consider the following algorithm that claims to solve this problem:

Algorithm Description: First check to see if s is labeled with 0 and t is labeled with 1. If they are not then return FALSE. Otherwise proceed to the next step.
Create a graph G' by removing all edges that go from a 0-labeled vertex to a 0-labeled vertex and by removing all edges that go from a 1-labeled vertex to a 1-labeled vertex.
Run graphsearch on G' starting from s. If t is visited then return TRUE. Otherwise return FALSE

Either prove this algorithm is correct or disprove it with a counter-example.

(NOTE: *graphsearch* is an algorithm that takes a graph G and a vertex s and returns a list of vertices that are reachable from s by a directed path in G. You do not have to prove the correctness of *graphsearch*, we will do this in class.)

This algorithm is correct.
Justification of correctness:
($\leftarrow$) Suppose that there is a path from s to t such that the binary sequence of the vertex weights in the path is of the form $(01)^i$. Then that means that s is labeled with 0 and t is labeled with 1 so the first conditional is satisfied. Then since the path is alternating from 0 to 1 to 0 and so on, there will never be any edges in this path that go from 0 to 0 or 1 to 1. Therefore, this path will also be in G'. So when graphsearch is performed on G' starting at s, t will be visited and the algorithm will return TRUE.
($\rightarrow$) Suppose the algorithm outputs TRUE, then it must be the case that s is labeled with 0 and t is labeled with 1. Furthermore, there must be a path from s to t in G'. Since G' does not have any 0 to 0 or 1 to 1 edges, all paths in G' must be alternating. Therefore this path in G' from s to t corresponds to an alternating path of the form $(01)^i$.