Mason Blanford
CS 325: Algorithms
Assignment 1, Page 1

Problem 1

Prove the statements True or False. If True, graph the functions, marking c and n_0.

1a (True)

$$\frac{n(n+1)}{2} \in O(n^3)$$

Simplify the limit.

$$\lim_{n\to\infty} \frac{T(n)}{g(n)} = \lim_{n\to\infty} \frac{\frac{n(n+1)}{2}}{n^3} = \lim_{n\to\infty} \frac{n(n+1)}{2n^3} = \lim_{n\to\infty} \frac{n^2+n}{2n^3} \text{ by product of quotient's reciprocal, then distribution}$$

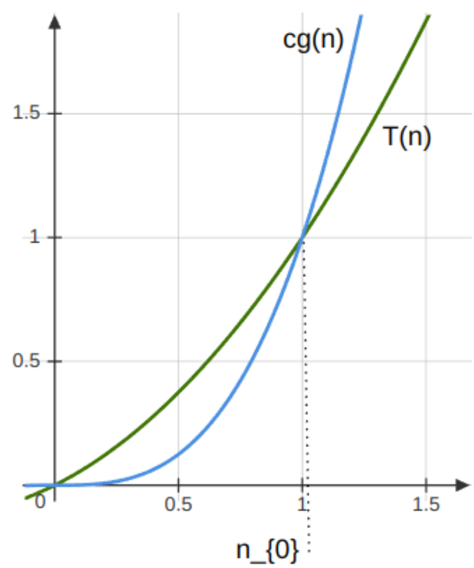
Direct substitution yields indeterminate form $\frac{\infty}{\infty}$. Use L'Hôpital's rule.

$$\lim_{n\to\infty} \frac{T'(n)}{g'(n)} = \frac{2n+1}{6n^2}$$

$$\lim_{n\to\infty} \frac{T''(n)}{g''(n)} = \frac{2}{12n}$$

$$\lim_{n\to\infty} \frac{T'''(n)}{g'''(n)} = \frac{0}{12} = 0$$

The limit approaches 0, so $\frac{n(n+1)}{2} \in O(n^3)$.



Continued on next page:

1b (True)

$$\frac{n(n+1)}{2} \in \Theta(n^2)$$

Simplify the limit.

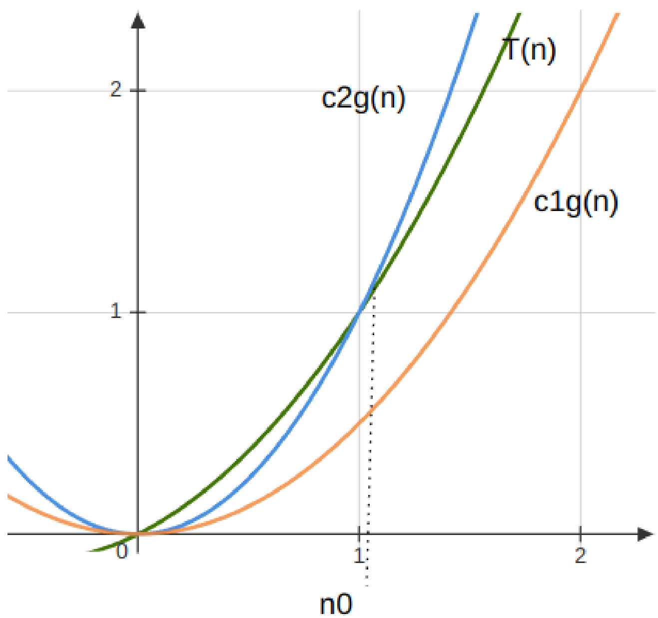
$$\lim_{n\to\infty} \frac{T(n)}{g(n)} = \lim_{n\to\infty} \frac{\frac{n(n+1)}{2}}{n^2} = \lim_{n\to\infty} \frac{n(n+1)}{2n^2} = \lim_{n\to\infty} \frac{n^2+n}{2n^2} \text{ by product of quotient's reciprocal, then distribution}$$

Direct substitution yields indeterminate form $\frac{\infty}{\infty}$. Use L'Hôpital's rule.

$$\lim_{n\to\infty} \frac{T'(n)}{g'(n)} = \frac{2n+1}{4n}$$

$$\lim_{n\to\infty} \frac{T''(n)}{g''(n)} = \frac{2}{4} = \frac{1}{2}$$

The limit approaches $\frac{1}{2}$, a constant, so $\frac{n(n+1)}{2} \in \Theta(n^2)$.



Continued on next page:

1c (False)

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

$$\lim_{n\to\infty} \frac{T(n)}{g(n)} = \lim_{n\to\infty} \frac{10n-6}{78n+2020}$$

Direct substitution yields indeterminate form $\frac{\infty}{\infty}$. Use L'Hôpital's rule.

$$\lim_{n\to\infty} \frac{T'(n)}{g'(n)} = \frac{10}{78} = \frac{5}{39}$$

The limit approaches constant $\frac{5}{39}$, so $10n - 6 \notin \Omega(78n + 2020)$. A true alternative is $10n - 6 \in \Theta(78n + 2020)$.

1d (True)

$$n! \in \Omega(0.00001n)$$

$$\lim_{x\to\infty} \frac{T(n)}{g(n)} = \lim_{x\to\infty} \frac{n!}{0.00001n}$$

Use factorial expansion, where $n! = n(n-1)(n-2)(3)(2)(1)$.

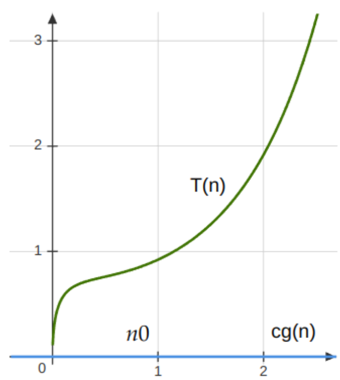
$$\lim_{x\to\infty} \frac{T(n)}{g(n)} = \lim_{x\to\infty} \frac{n \cdot (n-1) \cdot (n-2) \text{ ... } 3 \cdot 2 \cdot 1}{0.00001n}$$

$$= \lim_{x\to\infty} \frac{(n-1) \cdot (n-2) \text{ ... } 3 \cdot 2 \cdot 1}{0.00001} \text{ by canceling } n$$

$$= \lim_{x\to\infty} \frac{(\infty-1) \cdot (\infty-2) \text{ ... } 3 \cdot 2 \cdot 1}{0.00001} \text{ by direct substitution}$$

$$= \lim_{x\to\infty} \frac{\infty}{0.00001} = \infty$$

The limit approaches ∞, so $n! \in \Omega(0.00001n)$.



Problem 2

Consider the pseudocode for this algorithm:

```
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?
The algorithm finds the range of elements in an input array.

b) What is its basic operation (i.e., the line or operation executed the most)?
The loop's beginning is executed the most:

```
for i = 1 to n-1:
```

c) How many times is the basic operation executed?
The basic operation is executed $n - 1$ times, where n is the input array size.

d) What is the algorithm's time complexity?
The algorithm's time complexity is $O(n)$.

Problem 3

Prove this non-recursive algorithm by induction.

```
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 function's loop invariant.
At the start of each loop iteration for array Arr, the sub-array $Arr[i]$ to $Arr[j]$ holds the original array's elements but in reverse order.

b) Write an induction proof.

Initialization (Base case):
Assume case $P(1)$, where $n = len(Arr) = 1$.

$$i = \rfloor \frac{n-1}{2} \lfloor = \rfloor \frac{1-1}{2} \lfloor = \rfloor \frac{0}{2} \lfloor = 0$$

$$j = \rfloor \frac{n}{2} \lfloor = \rfloor \frac{1}{2} \lfloor = 0$$

The loop begins, because $(i = 0) \geq 0$ and $(j = 0) \leq ((n = 1) - 1)$, and $0 = 0$ in both events.

Since $Arr[i] = Arr[j]$, the loop returns the same Arr, which is Arr reversed when $len(Arr) = 1$.

Continued on next page:

Problem 3 (Continued)

Maintenance (Inductive case):
From the loop invariant, the sub-array $Arr[i]$ to $Arr[j]$ is reversed for $P(n)$, where $n =$ length Arr.

The same is true for $P(n + 1)$:

$$i = \rfloor \frac{(n+1)-1}{2} \lfloor = \rfloor \frac{n}{2} \lfloor$$

$$j = \rfloor \frac{n+1}{2} \lfloor$$

The loop begins, as $i \geq 0$ and $j \leq (n - 1)$.

If Arr is even-length, the loop begins at these indices. The elements swap.

$$Arr[i] = Arr[\frac{n}{2} - 1]$$

$$Arr[j] = Arr[\frac{n}{2}]$$

If Arr is odd-length, the loop begins at the same indice. The element swaps with itself.

$$Arr[i] \text{ and } Arr[j] = Arr\left[\frac{n}{2} - 1\right]$$

Then, i decrements by 1, and j increments by 1.

Termination: (Conclusion)
The loop terminates when $i = -1$ and $j = n$. By the loop invariant, Arr is reversed from $Arr[i]$ to $Arr[j]$, which encompasses the entire length.