# Assignment 1 pg. 1

Thursday, September 26, 2024 10:21 AM

Liam Geraghty - 300356748

1. a. True

Let $f(n) = 2n^2 + 10n^3$, $g(n) = n^3$
Let $c$ and $n_0$ be constants

$n^2 \leq n^3 < 2n^2 + 10n^3$ for all $n \geq 1$
$\Rightarrow 2n^2 + 10n^3 \leq 2n^3 + 10n^3$
$\leq 12n^3$ for all $n \geq 1$

$\Rightarrow f(n) \leq c \cdot g(n)$ for all $n \geq n_0$,
where $c = 12$ and $n_0 = 1$
$\Rightarrow 2n^2 + 10n^3$ is $O(n^3)$

b. True

Let $f(n) = \sqrt{n} + 10\log_2 n$, $g(n) = n$
Let $c$ and $n_0$ be constants

$\log_2 n \leq \sqrt{n}$ for all $x \geq 16$
$\Rightarrow \sqrt{n} + 10\log_2 n \leq \sqrt{n} + 10\sqrt{n}$
$\leq 11\sqrt{n}$

$\sqrt{n} \leq n$ for all $n \geq 1$
$\Rightarrow 11\sqrt{n} \leq 11n$ for all $n \geq 16$

$\Rightarrow f(n) \leq c \cdot g(n)$ for all $n \geq n_0$, where
$c = 11$ and $n_0 = 16$
$\Rightarrow \sqrt{n} + 10\log_2 n$ is $O(n)$

c. False

Assume $3^n + n^3$ is $\Theta(2^n)$

$\Rightarrow c' \cdot 2^n \leq 3^n + n^3 \leq c'' \cdot 2^n$ for all $n \geq n_0$.
where $c' > 0$ and $c'' > 0$

$\Rightarrow$ There exist $c'' > 0$ and $n_0$ s.t.
$3^n + n^3 \leq c'' \cdot 2^n$ for all $n \geq n_0$
$\Rightarrow \dfrac{3^n + n^3}{2^n} \leq c''$

$\lim_{n\to\infty} \dfrac{3^n + n^3}{2^n} = \lim_{n\to\infty} \left( (3/2)^n + \dfrac{n^3}{2^n} \right)$
$= \infty$

$\Rightarrow \infty \leq c''$ $\longrightarrow$ This is a contradiction. $\infty$ has no upper bound.

By contradiction, $3^n + n^3$ cannot be $\Theta(2^n)$

# Assignment 1 pg. 2

Sunday, September 29, 2024 12:20 PM

Liam Geraghty - 300356748

d.

Assume $\log_{10} n + 10n^3 + 100 \text{ is } \Omega(n^2)$

$\Rightarrow$ There exists $c > 0$ and $n_0$ s.t.
$\log_{10} n + 10n^3 + 100 \geq c \cdot n^2$ for all $n \geq n_0$

$$\Rightarrow \frac{\log_{10} n}{n^3} + 10 + \frac{100}{n^3} \geq \frac{c}{n}$$

$$\lim_{n\to\infty}\left(\frac{\log_{10} n}{n^3} + 10 + \frac{100}{n^3}\right) = \lim_{n\to\infty} \frac{1}{3n^3 \ln 10} + 10$$
$$= 10$$

$$\lim_{n\to\infty} \frac{c}{n} = 0$$

$\Rightarrow 10 \geq 0$

$$\lim_{n\to 1^+}\left(\frac{\log_{10} n}{n^3} + 10 + \frac{100}{n^3}\right) = 0 + 10 + 100$$
$$= 110$$

$$\lim_{n\to 1^+} \frac{c}{n} = c$$

$\Rightarrow 110 \geq c$

$\Rightarrow$ If $c = 1$ and $n_0 = 1$,

$\log_{10} n + 10n^3 + 100 \geq c \cdot n^2, \quad n \geq n_0$

$\Rightarrow \log_{10} n + 10n^3 + 100 \text{ is } \Omega(n^2)$

2. a. $T(n)$ is $O(\sqrt{n})$

This is the case where a large prime number is inputted. Line 5 will never evaluate to true, so the loop will continue until $x^2 \leq n$. Because x increments by 1 each iteration, it will take $\lfloor \sqrt{n} \rfloor$ iterations before the loop is complete. Then the function returns true.

b. $B(n)$ is $O(1)$

This is the case where n is any even number. On the first iteration of the loop, $x=2$. Line 5 will evaluate $n \% x == 0$ to true as all even numbers are divisible by 2. This will trigger the 'return false' and the function will end after a constant number of operations

# Assignment 1 pg. 3

Sunday, September 29, 2024 1:04 PM

Liam Geraghty - 300356748

3.

```
Algorithm sortStack(Stack s):                                   
Input: An n-element stack of integers                           
  Stack tempStack                                               1
  int tmp                                                       1

  tempStack.push(s.pop())                                       1
  while !s.isEmpty()                                            n+1
    if s.peek() >= tempStack.peek()                             n
      tempStack.push(s.pop())                                   n
    else
      tmp = s.pop()                                             n
      while !tempStack.isEmpty() and tmp < tempStack.peek()     1+2+3+...+n
        s.push(tempStack.pop())                                 1+2+3+...+n-1
      tempStack.push(tmp)                                       n

  while !tempStack.isEmpty()                                    n+1
    s.push(tempStack.pop())                                     n
```

⇒ Worst case of sortStack() is $O(n^2)$. This would occur if the stack was sorted with the smallest elements on the bottom.

4. a. The lineup would be implemented using a Deque. Cashiers would be implemented using a class with a single instance variable representing the customer they are helping.

```
Algorithm serve(int i):
Input: integer i representing cashier 1 or 2
  if lineup is empty
    throw error

  if i == 1
    cashier1.setCustomer(lineup.removeFirst())
  else if i == 2
    cashier2.setCustomer(lineup.removeremoveFirst())

Algorithm interruptService(int i):
Input: integer i representing cashier 1 or 2
  if i == 1
    lineup.addLast(cashier1.removeCustomer())
  else if i == 2
    lineup.addLast(cashier2.removeCustomer())

Algorithm newCustomer(Customer p)
Input: Customer to add
  lineup.addLast(p)

Algorithm giveUp(int n)
Input: integer n representing last n Customers in line
  for i <- 0 to n do
    lineup.removeLast()
```

# Assignment 1 pg. 4

Sunday, September 29, 2024 5:55 PM

Liam Geraghty - 300356748

b.

| Operations: | Output: |
|---|---|
| Newcustomer(A); | |
| Newcustomer(B); | |
| Newcustomer(C); | |
| PrintLineup(); | A, B, C |
| Serve(2); | |
| Serve(1); | |
| Serve(1); | |
| Newcustomer(D); | |
| Newcustomer(E); | |
| PrintLineup(); | D, E |
| InterruptService(2); | |
| InterruptService(1); | |
| PrintLineup(); | D, E, A, C |
| Newcustomer(F); | |
| GiveUp(2); | |
| PrintLineup(); | D, E, A |