Term Test—Sample Solutions

CSC165H1 / LEC0101/0201 — Danny Heap

November 21st, 1:40 OR 3:10 — Duration: **80 minutes**

Question 1. short answers [9 marks]

Part (a) step counting [1 mark]

Read over function $f(n)$ below and state how many times the loop iterates when $f(11)$ is called.

```python
def f(n: int) -> int:
    """Assume i >= 0"""
    i = 5
    while i < 5 * n:
        i = i * i
```

sample solution: 2 times

Part (b) *i* in terms of *s* [1 mark]

For function $f(n)$ above, find a formula for $i(s)$, the value of i after s iterations of the loop body.

sample solution: $i(s) = 5^{2^s}$

Part (c) step counting formula [1 mark]

Use your work in the previous parts to find a formula for the exact number of iterations of the loop if $f(n)$ is called, for some positive natural number n. Use floor or ceiling to make sure that your formula specifies the appropriate integer.

sample solution: $\lceil \log_2(\log_5(n) + 1) \rceil$

Part (d) asymptotic comparisons [3 marks]

Let $f(n) = 5n^3 + 2n$ and let $g(n) = 16\log(n)$. **Circle** each **true** statement below. Do **nothing** to **false** statements. You gain points for each statement correctly circled, or correctly left uncircled.

$f(n) + g(n) \in \Theta(f(n))$ $\qquad$ $g(n) \in \Theta(f(n) + g(n))$ $\qquad$ $f(n) \cdot g(n) \in \mathcal{O}(g(n))$

$f(n) \in \mathcal{O}(f(n) \cdot g(n))$ $\qquad$ $f(n) \cdot g(n) \in \Omega(2^n)$ $\qquad$ $2^n \in \Omega(f(n) \cdot g(n))$

sample solution: Circle only $f(n) + g(n) \in \Theta(f(n))$, $f(n) \in \mathcal{O}(f(n) \cdot g(n))$, and $2^n \in \Omega(f(n) \cdot g(n))$.

Part (e) binary numbers [3 marks]

Theorem 4.2 of the course notes guarantees a unique binary representation with the left-most bit being 1, for each positive natural number. Beneath each quantity below, write the number of bits (**bi**nary digi**ts**) in its unique binary representation.

2^{15} $\qquad$ $2^{15} - 1$ $\qquad$ $4 \times (2^{15} - 1)$

sample solution: 2^{15} has 16 bits, $2^{15} - 1$ has 15 bits, $4 \times (2^{15} - 1)$ has 17 bits.

Question 2. algorithm analysis [11 marks]

Read over function **has_mod_3**. Assume that **integer_list** contains only entries from $\{0, 1, 2\}$, with duplicates allowed. Define n as the length of **integer_list** and $WC_{has\_mod\_3}(n)$ as the largest number of "steps," for all **integer_list** of length n.

In what follows, if **has_mod_3** returns **True** right after examining k entries in **integer_list**, count this as k steps. If **has_mod_3** returns **False** after examining all n entries in **integer_list**, count this as $n + 1$ steps.

```
def has_mod_3(integer_list) -> bool:
    for i in range(len(integer_list)):
        if integer_list[i] % 3 == 0:
            return True
    return False
```

Part (a) lower bound [2 marks]

Find and prove a lower bound, $L(n)$ for $WC_{has\_mod\_3}(n)$. Your lower bound should be in the same asymptotic complexity class as the upper bound you find in the next question.

sample solution: $L(n) = n$ is a lower bound on $WC_{has\_mod\_3}(n)$, the worst run-time for inputs of size n with entries taken from $\{0, 1, 2\}$.

header: Let $n \in \mathbb{N}$. Let **LL** the list where each of the n entries is a 1. I want to show that **has_mod_3(LL)** takes at least n steps.

body: **has_mod_3(LL)** examines all n entries from **LL** at a cost of n steps. This means $WC_{has\_mod\_3}(n) \geq L(n)$ ■

Part (b) upper bound [2 marks]

Find and prove an upper bound, $U(n)$ for $WC_{has\_mod\_3}(n)$. Your upper bound should be in the same asymptotic complexity class as the lower bound you found in the previous question.

sample solution: $U(n) = n + 1$ is an upper bound on the run-times of **has_mod_3(x)** for all lists x of length n with entries taken from $\{0, 1, 2\}$.

header: Let $n \in \mathbb{N}$. Let **AL** be an arbitrary list consisting of n elements from $\{0, 1, 2\}$. I want to show that **has_mod_3(AL)** takes no more than $n + 1$ steps.

body: **has_mod_3(AL)** examines no more than n entries from **AL** and may then return **False**, for a total of $n + 1 \leq U(n)$ steps. Since **AL** is arbitrary this means $WC_{has\_mod\_3}(n) \leq U(n)$ ■

Term Test—Sample Solutions

CSC165H1 / LEC0101/0201 — Danny Heap

November 21st, 1:40 OR 3:10 — Duration: **80 minutes**

Part (c) average for length 3 [3 marks]

What is the average number of steps taken by **has_mod_3** for lists of length 3? Show your calculations, and explain them, to arrive at this result. Assume each input list is equally likely.

sample solution: The number of lists that have 0 in the first position, and hence return **True** after one step is $1 \times 3 \times 3$ or 9. The number of lists that have 1 or 2 in their first position, and 0 in their second position, and hence return **True** after 2 steps is $2 \times 1 \times 3$ or 6. The number of lists that have some combination of 1 and 2 in their first two positions and 0 in their third position is $2 \times 2 \times 1$ or 4. Finally, the number of lists that have no 0s and take $n+1$ steps is $2 \times 2 \times 2$ or 8. The total number of lists of length 3 is $3 \times 3 \times 3$ or 27. Thus the average number of steps is:

$$\frac{1 \times 9 + 2 \times 6 + 3 \times 4 + 4 \times 8}{27} = \frac{65}{27}, \text{ a bit more than 2 steps}$$

Part (d) average for length n [4 marks]

Find a closed formula for the average number of steps taken by **has_mod_3** for lists of length n. Show your calculations to arrive at this result. You may find the following formula helpful (although you are not required to use it):

$$\sum_{i=0}^{i=n-1} ir^i = \frac{nr^n}{r-1} + \frac{r - r^{n+1}}{(r-1)^2}$$

Assume each input list is equally likely.

sample solution: In general, if the first 0 is encountered at position i there there are $i-1$ positions filled with 1s and 2s, so 2^{i-1} possibilities, and $n-i$ positions filled with 1s, 2s, or 0s, so 3^{n-i} possibilities. The lists with no 0s fill all positions with 1s and 2s, so 2^n possibilities. Summing up the number of steps taken by each subset of lists, and dividing by 3^n total lists gives:

$$(1/3^n)\left(\sum_{i=1}^{n} i2^{i-1}3^{n-i}\right) + \frac{2^n(n+1)}{3^n} = \left(\sum_{i=1}^{n} i2^{-1}\frac{2^i}{3^i}\right) + \left(\frac{2}{3}\right)^n (n+1)$$

factor out 3^n

$$= (1/2)\left(\sum_{i=0}^{n} i\frac{2^i}{3^i}\right) + \left(\frac{2}{3}\right)^n (n+1)$$

factor out 1/2, add 0 to summation

$$= (1/2)\left(\frac{(n+1)(2/3)^{n+1}}{(2/3)-1} + \frac{(2/3)-(2/3)^{n+2}}{((2/3)-1)^2}\right) + \left(\frac{2}{3}\right)^n (n+1)$$

substitute n with $n+1$ in formula, so it applies to this case

$$= (1/2)\left(\frac{(n+1)(2/3)^{n+1}}{-(1/3)} + \frac{(2/3)-(2/3)^{n+2}}{(-1/3)^2}\right) + \left(\frac{2}{3}\right)^n (n+1)$$

calculate denominators

$$= \quad -(n+1)\left(\frac{2}{3}\right)^n + \frac{(1/3) - (2^{n+1}/3^{n+2})}{(-1/3)^2} + \left(\frac{2}{3}\right)^n (n+1)$$

calculate first term, which cancels last

$$= \quad \frac{(1/3) - (2^{n+1}/3^{n+2})}{1/9} = 3 - 2(2/3)^n$$

Question 3. induction [7 marks]

Part (a) proof [5 marks]

Prove the following statement using induction:

$$\forall n \in \mathbb{N}, n \geq 4 \Rightarrow 3^n > n^3 + n$$

sample solution: Define $P(n) : n \geq 4 \Rightarrow 3^n > n^3 + n$. I will prove $\forall n \in \mathbb{N}, P(n)$ by induction.

base case, $P(4)$: $3^4 = 81 > 68 = 4^3 + 4$, which verifies $P(4)$.

inductive step: Let $n \in \mathbb{N}$ and assume $n \geq 4$ and $P(n)$, that is $3^n > n^3 + n$. I want to show $P(n+1)$, that is $3^{n+1} > (n+1)^3 + (n+1)$.

body: Notice:

$$\begin{aligned}
3^{n+1} &= 3 \times 3^n \\
&> 3n^3 + 3n = n^3 + n^3 + n^3 + 3n \qquad \text{\# by IH} \\
&\geq n^3 + 3n^2 + 9n + 3n \qquad \text{\# since } n \geq 4 \geq 3 \\
&= n^3 + 3n^2 + 3n + 9n = n^3 + 3n^2 + 3n + 4n + n + 4n \\
&\geq n^3 + 3n^2 + 3n + 1 + n + 1 \qquad \text{\# since } n \geq 4 \geq 1/4 \\
&= (n+1)^3 + (n+1) \qquad \blacksquare
\end{aligned}$$

Part (b) analysis [2 marks]

Explain why the hypothesis $n \geq 4$ is needed, or else explain why it is not needed.

sample solution: The hypothesis is needed because $3^3 < 3^3 + 3$, and the claim is only true for integers greater than, or equal to, 4.

Question 4. big-Omega [5 marks]

In what follows use the following definition for $f \in \Omega(g)$:

$$\exists c, n_0 \in \mathbb{R}^+, \forall n \in \mathbb{N}, n \geq n_0 \Rightarrow f(n) \geq cg(n)$$

Define $f(n) = n^3$ and $g(n) = 2^n$. Prove that $f \notin \Omega(g)$. You may not use techniques of calculus such as limits, and you may not use Theorem 5.1 from the course notes. You may assume, without proof, that for any integer k greater than 4, $2^k > 6k$ (although you are not required to use this).

Term Test—Sample Solutions

CSC165H1 / LEC0101/0201 — Danny Heap

November 21st, 1:40 OR 3:10 — Duration: **80 minutes**

sample solution: I will prove that $f \notin \Omega(g)$, that is:

$$\forall c, n_0 \in \mathbb{R}^+, \exists n \in \mathbb{N}, n \geq n_0 \wedge f(n) < cg(n)$$

header: Let $c, n_0 \in \mathbb{R}^+$. Let

$$n = 2^{1+\lceil \max(\lg(n_0), 4, 1+\lg(1/c)) \rceil}$$

I want to show that $n \geq n_0 \wedge c2^n > n^3$. (My choice of n is motivated by Problem Set #3, 2(a)).

body: By choice of n we have $n > 2^{lg(n_0)} = n_0$. Also, by choice of n we have $n > 2^{1+\lg(\lg(1/c))} = 2\lg(1/c)$, so $n/2 > \lg(1/c)$, so $2^{n/2} > 1/c$. Finally, by choice of n we have $n \geq 2^5$ and n is an integer power of 2, so $n > 6\lg(n)$, or $n/2 > 3\lg(n)$ and $2^{n/2} > n^3$. Putting these together, and raising to powers of 2, we have:

$$\begin{aligned}
2^{n/2} &> n^3 \\
2^n &> 2^{n/2}n^3 \quad (*) \qquad \text{\# multiply by } 2^{n/2} \\
2^{n/2} &> 1/c \\
2^{n/2}n^3 &> n^3 1/c \quad (**) \qquad \text{\# multiply by } n^3 \\
2^n &> n^3 1/c \qquad \text{\# } (*) \text{ and } (**) \\
c2^n &> n^3 \qquad \blacksquare
\end{aligned}$$