TCSS 343 - Week 1 - Thursday

Jake McKenzie

January 13, 2019

Asymptotics

"You're always you, and that don't change, and you're always changing, and there's nothing you can do about it."

...

Neil Gaiman

"If one proves the equality of two numbers a and b by showing first that $a \leqslant b$ and then that $b \geqslant a$, it is unfair; one should instead show that they are really equal by disclosing the inner ground for their equality".

...

Emmy Noether

"These are not my stars. Even the heavens are denied me here".

...

Alidar Jarok

0. Prove the following theorem. Use a **direct proof** to find constants that satisfy the definition of big Θ or use the **limit test**. Make sure your proof is tidy. That means it is complete, concise, clear and precise.

Theorem 0. $21n - 71 \in \Theta(n)$

1. Prove the following theorem. Use a **direct proof** to find constants that satisfy the definition of big Θ or use the **limit test**. Make sure your proof is tidy. That means it is complete, concise, clear and precise.

Theorem 1. $\log n^n + 21\sqrt{n} \in \Theta(n \log n)$

2. Prove the following theorem. Use a **direct proof** to find constants that satisfy the definition of big Θ or use the **limit test**. Make sure your proof is tidy. That means it is complete, concise, clear and precise.

Theorem 2. $\log(2^n + 2) \in \Theta(n)$

3. Use the definition of big Oh notation to **prove** or **disprove** that if $f(n) \in O(g(n))$ and $g(n) \in O(f(n))$ then $f(n) = g(n) \forall n$.

4. Use the definition of big Oh notation to **prove** or **disprove** that if $f(n) \in O(h(n))$ and $g(n) \in O(h(n))$ then $f(n) + g(n) \in O(h(n)) \forall n$.

Algorithm Convert_to_Binary (n) ;
Input: n (a positive integer).
Output: b (an array of bits corresponding to the binary representation of n).

```
begin
    t := n ; { we use a new variable t to preserve n }
    k := 0 ;
    while t > 0 do
        k := k + 1 ;
        b[k] := t mod 2 ;
        t := t div 2 ;
end
```

Figure 2.6 Algorithm *Convert_to_Binary*.

5. For the following problem we will explore an induction and the notion of an **invariant** of an algorithm, which put simply, is a statement about a variable correct independent of the number of times a loop is executed. For the purposes of this algorithm the expression: $n = t \times 2^k + m$ is a loop invariant. Loop invariants can be hard to find but are often the heart of any algorithm.
   Induction Hypothesis: if m is the integer represented by the binary array $b[1 \ldots k]$, then $n = t \times 2^k + m$. To prove the correctness of this algorithm we must prove three conditions:

   (a) The hypothesis is true at the beginning of the loop.

   (b) The truth of the hypothesis at steps k implies the step $k + 1$.

   (c) When the loop terminates, the hypothesis implies the correctness of the algorithm.

   Attempt to complete the proof given the information I've given you to start.