CSC165H1: Problem Set 3

Due Friday March 6 2020, before 4pm

General instructions

Please read the following instructions carefully before starting the problem set. They contain important information about general problem set expectations, problem set submission instructions, and reminders of course policies.

- Your problem sets are graded on both correctness and clarity of communication. Solutions that are technically correct but poorly written will not receive full marks. Please read over your solutions carefully before submitting them.
- Each problem set may be completed in groups of up to three. If you are working in a group for this problem set, please consult https://github.com/MarkUsProject/Markus/wiki/Student_Groups for a brief explanation of how to create a group on MarkUs.

  Exception: Problem Set 0 must be completed individually.
- Solutions must be typeset electronically, and submitted as a PDF with the correct filename. **Hand-written submissions will receive a grade of ZERO.**

  The required filename for this problem set is **problem_set3.pdf**.
- Problem sets must be submitted online through MarkUs. If you haven't used MarkUs before, give yourself plenty of time to figure it out, and ask for help if you need it! If you are working with a partner, you must form a group on MarkUs, and make one submission per group. "I didn't know how to use MarkUs" is not a valid excuse for submitting late work.
- Your submitted file(s) should not be larger than 9MB. You might exceed this limit if you use a word processor like Microsoft Word to create a PDF; if it does, you should look into PDF compression tools to make your PDF smaller, although please make sure that your PDF is still legible before submitting!
- Submissions must be made *before* the due date on MarkUs. You may use *grace tokens* to extend the deadline; please see the Homework page for details on using grace tokens.
- The work you submit must be that of your group; you may not use or copy from the work of other groups, or external sources like websites or textbooks.

Additional instructions

- When doing a proof by induction, always label the step(s) that use the induction hypothesis.
- You may not use forms of induction we have not covered in lecture.
- Please follow the same guidelines as Problem Set 2 for all proofs.

1. **[8 marks] Proofs by induction.**

   (a) Let $x \in \mathbb{R}$. We define the following recursive sequence of numbers:

$$a_n = \begin{cases} x, & \text{if } n = 0 \\ x \cdot \prod_{i=0}^{n-1} a_i, & \text{otherwise} \end{cases}$$

   Find a closed-form expression for a_n (in terms of x and n), and prove that your expression is correct using induction.

   (b) Consider the following definitions.

   Definition 1 (ternary string)**.** A **ternary string** is a string over the set of characters $\{0, 1, 2\}$. For example, 0210 is a ternary string of length 4, and 020202 is a ternary string of length 6. The empty string, denoted ϵ, is a ternary string of length 0.

   Definition 2 (digit sum, E_n, O_n)**.** Let s be a ternary string. We say that the **digit sum** of s is the sum of its characters. For example, the digit sum of 0210 is 3, the digit sum of 020202 is 6, and the digit sum of ϵ is 0.

   Let $n \in \mathbb{N}$. We define $\boldsymbol{E_n}$ to be the number of ternary strings of length n whose digit sum is even, and $\boldsymbol{O_n}$ to be the number of ternary strings of length n whose digit sum is odd.

   Find closed-form expressions for E_n and O_n in terms of n, and prove that your expressions are correct using induction. Use the same approach as counting subsets from Worksheet #9. You may use this fact about divisibility:

$$\forall x, y \in \mathbb{Z},\ \mathit{Even}(x + y) \Leftrightarrow (\mathit{Even}(x) \land \mathit{Even}(y)) \lor (\mathit{Odd}(x) \land \mathit{Odd}(y))$$

   HINT: we're asking you to prove a statement of the form $\forall n \in \mathbb{N},\ E_n = \_\_ \land O_n = \_\_$; use a single induction predicate that involves both E_n and O_n.

2. **[9 marks] Binary representations.** On Worksheet #10, we looked at representing real numbers between zero and one in binary. However, we saw that some numbers had infinitely repeating binary representations, meaning they cannot be represented exactly by computers (which have only a finite number of bits to work with). In this question, we will consider what happens when we have to truncate these numbers in order to store them in a finite number of bits.

   For simplicity, we'll restrict our attention in this question to real numbers between 0 and 1.
   We define the set $S = \{a \mid a \in \mathbb{R} \text{ and } 0 \leq a < 1\}$.

   Definition 3 (Finite fractional binary representation)**.** Let $x \in S$. We say that a **finite fractional binary representation of** x is a number k and bits $b_1, b_2, \ldots, b_k \in \{0, 1\}$ that satisfy:

$$x = \sum_{i=1}^{k} \frac{b_i}{2^i}$$

   In this case, we write $x = (0.b_1 b_2 \ldots b_k)_2$.

   We also define the following predicate FB over $\mathbb{Z}^+ \times S$:

$$FB(n, x): \text{ “}x\text{ has a finite fractional binary representation with }n\text{ bits.”}$$

   Equivalently, $FB(n, x)$ means there exist bits $b_1, \ldots, b_n \in \{0, 1\}$ such that $x = \displaystyle\sum_{i=1}^{n} \frac{b_i}{2^i}$.

   (a) Consider the number $0.2 = 1/5$. It turns out this number does not have a finite fractional binary representation,[1] and in fact we can apply the technique we learned on Worksheet #10 to obtain the representation $0.2 = (0.\overline{0011})_2$. In other words, we represent 0.2 using the infinite sequence of bits $b_1 = 0, b_2 = 0, b_3 = 1, b_4 = 1, b_5 = 0, b_6 = 0, b_7 = 1, b_8 = 1, \ldots$.

   For all $n \in \mathbb{Z}^+$, we define z_n to be the number obtained by taking the first n bits from the infinite binary representation of 0.2. For example:

   - $z_1 = (0.0)_2 = 0$
   - $z_2 = (0.00)_2 = 0$
   - $z_3 = (0.001)_2 = 1/8 = 0.125$
   - $z_4 = (0.0011)_2 = 3/16 = 0.1875$
   - $z_8 = (0.00110011)_2 = 51/256 = 0.19921875$

   We say that these z_n's are **approximations** of 0.2, and that the **approximation error of** z_n is the quantity $0.2 - z_n$.

   Find, with proof, a closed-form formula for the approximation error in terms of n, when n is a multiple of 4. That is, fill in the blank in the following predicate logic statement, and then prove it:

$$\forall n \in \mathbb{Z}^+,\ 4 \mid n \Rightarrow 0.2 - z_n = \underline{\quad}$$

   You may use the following formulas, as long as you explicitly show how you use them:

   - For all $n \in \mathbb{Z}^+$ and $r \in \mathbb{R}$ if $r \neq 1$: $\displaystyle\sum_{i=0}^{n-1} r^i = \frac{1 - r^n}{1 - r}$
   - For all $r \in \mathbb{R}$ if $|r| < 1$: $\displaystyle\sum_{i=0}^{\infty} r^i = \frac{1}{1 - r}$

[1] In fact, it is a good exercise to prove that the *only* numbers in S that have a finite fractional representation are the ones that can be written as a fraction where the denominator is a power of two.)

(b) Prove the following statement by induction on n:

$$\forall n \in \mathbb{Z}^+,\ \forall x \in S,\ \exists x_1 \in S,\ FB(n, x_1) \land 0 \leq x - x_1 \leq \frac{1}{2^n}$$

Your proof should follow the same structure as the example we did in lecture in Week 6.

3. **[10 marks] Asymptotic notation.**

   Prove or disprove each of the following statements.

   (a) $n^4 + 165n^3 \in \mathcal{O}(n^4 - n^2)$

   (b) $\exists f : \mathbb{N} \to \mathbb{R}^+,\ \big(\forall g : \mathbb{N} \to \mathbb{R}^{\geq 0},\ g \in \mathcal{O}(f) \lor g \in \Omega(f)\big)$

   HINTS: note the codomain of f is $\mathbb{R}^+$, not $\mathbb{R}^{\geq 0}$. Also, you can define a function's behaviour based on cases, e.g.

$$f(n) = \begin{cases} ..., & \text{if } ... \\ ..., & \text{otherwise} \end{cases}$$

4. **[5 marks] Little-Oh.** Recall the definition of Big-Oh:

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

Here is one variation of this definition.

Definition 4 (little-oh)**.** Let $f, g : \mathbb{N} \to \mathbb{R}^{\geq 0}$. We say that g is **little-oh** of f, and write $g \in o(f)$, when:

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

This is a *stronger* property than Big-Oh, in the sense that if $g \in o(f)$, then $g \in \mathcal{O}(f)$.[2] While $g \in \mathcal{O}(f)$ says (colloquially) that "It is possible to scale up f so that it eventually dominates g," $g \in o(f)$ says that "No matter how you scale f (up or down), it will always eventually dominate g." Or, in terms of rates of growth, $g \in \mathcal{O}(f)$ means that g grows at most as quickly as f, while $g \in o(f)$ means that g grows strictly slower than f.

Prove the following statements about little-oh, using only the definitions of little-oh and Big-Oh. You may not use any external properties of Big-Oh in this question.

(a) **[Do not hand in—this question part is not graded.]** Prove that for all positive real numbers a and b, if $a < b$ then $n^a \in o(n^b)$.

(b) Prove that for all functions $f, g : \mathbb{N} \to \mathbb{R}^+$, if $g \in o(f)$ then $f \notin O(g)$.
HINT: the codomain of f and g is $\mathbb{R}^+$, not $\mathbb{R}^{\geq 0}$.

[2] The only difference between these two definitions is in the quantification of c, with the switch to universal quantification leading to a more powerful claim. In general, if the statement $\forall c \in \mathbb{R}^+,\ P(c)$ is true, then $\exists c \in \mathbb{R}^+,\ P(c)$ is also true.