CS/ECE 374 A ✧ Fall 2019
Homework 0
Due Tuesday, September 3, 2019 at 8pm

- **Each student must submit individual solutions for this homework.** For all future homeworks, groups of up to three students can submit joint solutions.
- **Submit your solutions electronically to Gradescope as PDF files.** Submit a separate PDF file for each numbered problem. If you plan to typeset your solutions, please use the LaTeX solution template on the course web site. If you must submit scanned handwritten solutions, please use a black pen on blank white paper and a high-quality scanner app (or an actual scanner).
- You are ***not*** required to sign up on Gradescope or Piazza with your real name and your illinois.edu email address; you may use any email address and alias of your choice. However, to give you credit for the homework, we need to know who Gradescope thinks you are. **Please fill out the web form linked from the course web page.**

☞ Some important course policies ☜

- **You may use any source at your disposal**—paper, electronic, or human—but you ***must*** cite ***every*** source that you use, and you ***must*** write everything yourself in your own words. See the academic integrity policies on the course web site for more details.
- The answer ***"I don't know"*** (and *nothing* else) is worth 25% partial credit on any required problem or subproblem on any homework or exam. We will accept synonyms like "No idea" or "WTF" or "\(ó_ò)/", but you must write *something*.

  On the other hand, only the homework problems you submit actually contribute to your overall course grade, so submitting "I don't know" for an entire numbered homework problem will almost certainly hurt your grade more than submitting nothing at all.
- **Avoid the Three Deadly Sins!** Any homework or exam solution that breaks any of the following rules will be given an ***automatic zero***, unless the solution is otherwise perfect. Yes, we really mean it. We're not trying to be scary or petty (Honest!), but we do want to break a few common bad habits that seriously impede mastery of the course material.
  - Always give complete solutions, not just examples.
  - Always declare all your variables, in English. In particular, always describe the precise problem your algorithm is supposed to solve.
  - Never use weak induction.

See the course web site for more information.

If you have any questions about these policies,
please don't hesitate to ask in class, in office hours, or on Piazza.

---

1. The infamous Scottish computational arborist Seòras na Coille has a favorite 26-node binary tree, whose nodes are labeled with the letters of the English alphabet. Preorder and inorder traversals of his tree yield the following letter sequences:

   Preorder: U X M Z I W J E O V N H R D T K G L Y A F S Q P C B
   Inorder: Z M X E J V O W I N D R T H U G K F A Q P S Y C B L

   (a) List the nodes in Professor na Coille's tree according to a postorder traversal.

   (b) Draw Professor na Coille's tree.

   You do *not* need to prove that your answers are correct. *[Hint: It may be easier to write a short Python program than to figure this out by hand.]*

2. For any string $w \in \{0,1\}^*$, let $sort(w)$ denote the string obtained by sorting the characters in w. For example, $sort(010101) = 000111$. The $sort$ function can be defined recursively as follows:

$$sort(w) := \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ 0 \cdot sort(x) & \text{if } w = 0x \\ sort(x) \bullet 1 & \text{if } w = 1x \end{cases}$$

   (a) Prove that $\#(0, sort(w)) = \#(0, w)$ for every string $w \in \{0,1\}^*$.

   (b) Prove that $sort(w \bullet 1) = sort(w) \bullet 1$ for every string $w \in \{0,1\}^*$.

   (c) Prove that $sort(sort(w)) = sort(w)$ for every string $w \in \{0,1\}^*$.

   Think about these two problems on your own; do not submit solutions:

   (d) Prove that $x \bullet 0 \neq y \bullet 1$ for all strings $x, y \in \{0,1\}^*$.

   (e) Prove that $sort(w) \neq x \bullet 10 \bullet y$, for all strings $w, x, y \in \{0,1\}^*$.

   You may assume without proof that $\#(a, uv) = \#(a, u) + \#(a, v)$ for any symbol a and any strings u and v, or any other result proved in class, in lab, or in the lecture notes. Your proofs for later parts of this problem can assume earlier parts even if you don't prove them. Otherwise, your proofs must be formal and self-contained.

3. Consider the set of strings $L \subseteq \{0,1\}^*$ defined recursively as follows:

   - The empty string ε is in L.
   - For any strings x in L, the strings $0x1$ and $1x0$ are also in L.
   - For any two *nonempty* strings x and y in L, the string $x \bullet y$ is also in L.
   - These are the only strings in L.

   This problem asks you to prove that L is the set of all strings w where the number of 0s is equal to the number of 1s. More formally, for any string w, let $\Delta(w) = \#(1,w) - \#(0,w)$, or equivalently,

$$\Delta(w) = \begin{cases} 0 & \text{if } w = \varepsilon \\ \Delta(x) - 1 & \text{if } w = 0x \\ \Delta(x) + 1 & \text{if } w = 1x \end{cases}$$

   (a) Prove that the string 11011100101000 is in L.

   (b) Prove that $\Delta(w) = 0$ for every string $w \in L$.

   (c) Prove that L contains every string $w \in \{0,1\}^*$ such that $\Delta(w) = 0$.

   You can assume the following properties of the Δ function, for all strings w and z.

   - Addition: $\Delta(wz) = \Delta(w) + \Delta(z)$.
   - Downward interpolation: If $\Delta(wz) > 0$ and $\Delta(z) < 0$, then there are strings x and y such that $w = xy$ and $\Delta(yz) = 0$.
   - Upward interpolation: If $\Delta(wz) < 0$ and $\Delta(z) > 0$, then there are strings x and y such that $w = xy$ and $\Delta(yz) = 0$.

   The interpolation properties are a type of "intermediate value theorem". **Think about how to prove these properties yourself.**

   You can also assume any other result proved in class, in lab, or in the lecture notes. Otherwise, your proofs must be formal and self-contained.

Solved Problems

Each homework assignment will include at least one solved problem, similar to the problems assigned in that homework, together with the grading rubric we would apply if this problem appeared on a homework or exam. These model solutions illustrate our recommendations for structure, presentation, and level of detail in your homework solutions. Of course, the actual **content** *of your solutions won't match the model solutions, because your problems are different!*

4. The ***reversal*** $\boldsymbol{w^R}$ of a string w is defined recursively as follows:

$$w^R := \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ x^R \bullet a & \text{if } w = a \cdot x \end{cases}$$

A ***palindrome*** is any string that is equal to its reversal, like `AMANAPLANACANALPANAMA`, `RACECAR`, `POOP`, `I`, and the empty string.

(a) Give a recursive definition of a palindrome over the alphabet Σ.

(b) Prove $w = w^R$ for every palindrome w (according to your recursive definition).

(c) Prove that every string w such that $w = w^R$ is a palindrome (according to your recursive definition).

You may assume without proof the following statements for all strings x, y, and z:

- Reversal reversal: $(x^R)^R = x$
- Concatenation reversal: $(x \bullet y)^R = y^R \bullet x^R$
- Right cancellation: If $x \bullet z = y \bullet z$, then $x = y$.

Solution:

(a) A string $w \in \Sigma^*$ is a palindrome if and only if either

- $w = \varepsilon$, or
- $w = a$ for some symbol $a \in \Sigma$, or
- $w = axa$ for some symbol $a \in \Sigma$ and some *palindrome* $x \in \Sigma^*$.

Rubric: 2 points = ½ for each base case + 1 for the recursive case. No credit for the rest of the problem unless this part is correct.

(b) Let w be an arbitrary palindrome.
Assume that $x = x^R$ for every palindrome x such that $|x| < |w|$.
There are three cases to consider (mirroring the definition of "palindrome"):

- If $w = \varepsilon$, then $w^R = \varepsilon$ by definition, so $w = w^R$.
- If $w = a$ for some symbol $a \in \Sigma$, then $w^R = a$ by definition, so $w = w^R$.
- Finally, if $w = axa$ for some symbol $a \in \Sigma$ and some palindrome $x \in P$,

then

$$\begin{aligned} w^R &= (a \cdot x \bullet a)^R \\ &= (x \bullet a)^R \bullet a && \text{by definition of reversal} \\ &= a^R \bullet x^R \bullet a && \text{by concatenation reversal} \\ &= a \bullet x^R \bullet a && \text{by definition of reversal} \\ &= a \bullet x \bullet a && \text{by the inductive hypothesis} \\ &= w && \text{by assumption} \end{aligned}$$

In all three cases, we conclude that $w = w^R$. ■

Rubric: 4 points: standard induction rubric (scaled)

(c) Let w be an arbitrary string such that $w = w^R$.
Assume that every string x such that $|x| < |w|$ and $x = x^R$ is a palindrome.
There are three cases to consider (mirroring the definition of "palindrome"):

- If $w = \varepsilon$, then w is a palindrome by definition.
- If $w = a$ for some symbol $a \in \Sigma$, then w is a palindrome by definition.
- Otherwise, we have $w = ax$ for some symbol a and some *non-empty* string x.
  The definition of reversal implies that $w^R = (ax)^R = x^R a$.
  Because x is non-empty, its reversal x^R is also non-empty.
  Thus, $x^R = by$ for some symbol b and some string y.
  It follows that $w^R = bya$, and therefore $w = (w^R)^R = (bya)^R = ay^R b$.

  [At this point, we need to prove that $a = b$ and that y is a palindrome.]

  Our assumption that $w = w^R$ implies that $bya = ay^R b$.
  The recursive definition of string equality immediately implies $a = b$.

  Because $a = b$, we have $w = ay^R a$ and $w^R = aya$.
  The recursive definition of string equality implies $y^R a = ya$.
  Right cancellation implies that $y^R = y$.
  The inductive hypothesis now implies that y is a palindrome.

  We conclude that w is a palindrome by definition.

In all three cases, we conclude that w is a palindrome. ■

Rubric: 4 points: standard induction rubric (scaled).

Standard induction rubric. For problems worth 10 points:

+ 1 for explicitly considering an *arbitrary* object.
+ 2 for a valid ***strong*** induction hypothesis
    - **Deadly Sin!** Automatic zero for stating a weak induction hypothesis, unless the rest of the proof is *absolutely perfect*.
+ 2 for explicit exhaustive case analysis
    - No credit here if the case analysis omits an infinite number of objects. (For example: all odd-length palindromes.)
    - −1 if the case analysis omits an finite number of objects. (For example: the empty string.)
    - −1 for making the reader infer the case conditions. Spell them out!
    - No penalty if the cases overlap (for example: even length at least 2, odd length at least 3, and length at most 5.)
+ 1 for cases that do not invoke the inductive hypothesis ("base cases")
    - No credit here if one or more "base cases" are missing.
+ 2 for correctly applying the ***stated*** inductive hypothesis
    - No credit here for applying a ***different*** inductive hypothesis, even if that different inductive hypothesis would be valid.
+ 2 for other details in cases that invoke the inductive hypothesis ("inductive cases")
    - No credit here if one or more "inductive cases" are missing.

For (sub)problems worth less than 10 points, scale and round to the nearest half-integer.