# CS/ECE 374 Sec A ✧ Spring 2025

# Homework 1

Due Wednesday, Jan 29, 2025 at 9am

---

- You can work in a group of up to **three** students. Read the instructions on the course website for additional details.
- **Submit your solutions electronically on the course Gradescope site 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, not just a phone camera).
- Late submissions will be accepted (for 75% credit) until midnight the day of the deadline.

---

## ☞ 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 including the policy on using AI tools.
- **Avoid the Two 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 specific problem your algorithm is supposed to solve.

---

**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 Ed.

---

0. Read the course policies and the instructions for this home work before you start.

1. Let $n$ be a positive integer represented in decimal notation as $a_{k-1}a_{k-2}\ldots a_1a_0$ where $k \geq 1$ and $a_{k-1} \geq 1$. Suppose we want to know whether $n$ is divisible by 11. There is a simple and nice rule. Let $n_1$ be the sum of the digits of $n$ in odd position and let $n_2$ be the sum of the digits in the even position. Then $n$ is divisible by 11 iff $n' = |n_1 - n_2|$ is divisible by 11. As an example if $n = 3311$ then $n' = 0$ and $n'$ is divisible by 11. If $n = 383974139871230$ then $n' = |(0+2+7+9+1+7+3+3) - (3+1+8+3+4+9+8)| = 4$ which is not divisible by 11 and hence $n$ is not divisible by 11. If $n = 909090909090909090909090$ then $n' = 108$. To check whether $n'$ is divisible by 11 we can apply the process again to obtain 9 which is not divisible by 11 and conclude that $n$ is not divisible by 11. We thus have a recursive algorithm where we stop the algorithm in the base case of $n$ being a single digit number, otherwise the algorithm computes $n'$ and recurses on $n'$. We would like to argue about the correctness and running time of this algorithm.

   (a) Prove that as long as $n$ has at least 2 digits, $n' < n$.

   (b) Prove that $n$ is divisible by 11 iff $n'$ is divisible by 11. You can give a direct proof or a proof by induction.

   (c) For any positive integer $x$ with decimal representation $b_{k-1}b_{k-2}\ldots b_1b_0$ with $b_{k-1} \geq 1$, prove that $k = O(\log x)$.

   (d) Using the preceding part show that the decimal representation of $n'$ has $O(\log k)$ bits where $k$ is the number of bits in the representation for $n$.

   (e) Assuming that summing $\ell$ decimal digits to compute the result of the sum in decimal representation takes $O(\ell)$ time, write a recurrence for the total time for the algorithm to terminate as a function of $k$, the number of digits in the representation of $n$. Using the recurrence, argue that the running time of the algorithm is $O(k)$.

   (f) **Not to submit for grading:** Write a recurrence for the depth of the recursion and show that it is $O(\log^* k)$ (you may want to look up and read about this function).

2. Consider the set of strings $L_1 \subseteq \{\texttt{0}, \texttt{1}\}^*$ defined recursively as follows:

   - The string $\varepsilon$ is in $L_1$.
   - For any string $x$ in $L_1$, the strings $\texttt{0}x\texttt{1}$ and the string $\texttt{1}x\texttt{0}$ are also in $L_1$.
   - For any two strings $x, y \in L_1$ with $|x|, |y| \geq 1$, the string $xy \in L_1$.
   - The only strings in $L_1$ are the ones generated by the preceding rules.

   Now consider the following language $L_2$ defined recursively as follows.

   - The string $\varepsilon$ is in $L_2$.
   - For any string $x$ in $L_2$ and strings $a, b$ with $|a| = |b| = 2$ where $ab$ is a string with equal number of $\texttt{0}$'s and $\texttt{1}$'s, the string $axb$ is in $L_2$.
   - For any two strings $x, y \in L_2$ with $|x|, |y| \geq 1$ the string $xy \in L_2$.
   - The only strings in $L_2$ are the ones generated by the preceding rules.

Let $L_{eq}$ be the set of binary strings that have an equal number of `0`'s and `1`'s. Let $L'_{eq}$ is the set of strings in $L_{eq}$ whose length is divisible by 4.

(a) Prove by induction that $L_1 \subseteq L_{eq}$.

(b) Prove by induction that $L_{eq} \subseteq L_1$.

(c) Argue that $L'_{eq} \not\subseteq L_2$ by describing a string that is in $L'_{eq}$ but is not in $L_2$. You should briefly justify why the string you describe is not in $L_2$.

(d) **Not to submit for grading:** Prove by induction that $L_2 \subseteq L'_{eq}$.

Let $\#(a, w)$ denote the number of times symbol $a$ appears in string $w$; for example,

$$\#(\texttt{0}, \texttt{101110101101011}) = 5 \quad \text{and} \quad \#(\texttt{1}, \texttt{101110101101011}) = 10.$$

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. Otherwise, your proofs must be formal and self-contained.

3. **For your reading, do not submit for grading:**

Consider the following recurrence.

$$T(n) = T(\lfloor n/2 \rfloor) + 2T(\lfloor n/3 \rfloor) + n^2 \quad n \geq 4, \text{ and } T(n) = 1 \quad 1 \leq n < 4.$$

One can prove that $T(n) = O(n^2)$. The goal of this problem is to show a more general statement and to refresh your induction skills.

Let $c_1, c_2, c_3$ be rational numbers such that $0 < c_1 \leq c_2 \leq c_3$ and $c_1^2 + c_2^2 + c_3^2 < 1$. Let $\gamma > 0$. Consider the recurrence

$$T(n) = T(\lfloor c_1 n \rfloor) + T(\lfloor c_2 n \rfloor) + T(\lfloor c_3 n \rfloor) + \gamma n^2, \quad n > 1/c_1, T(n) = 1 \quad n \leq 1/c_1.$$

- Prove by induction that $T(n) = O(n^2)$. More precisely show that $T(n) \leq an^2 + b$ for $n \geq 1$ where $a, b \geq 0$ are some fixed but suitably chosen constants (you get to choose and fix them based on $c_1, c_2, c_3, \gamma$). You may first want to try the concrete recurrence at the start of the problem. How does $a$ depend on $c_1, c_2, c_3, \gamma$?
- Consider the recursion tree for the recurrence. What is an asymptotic upper bound on the depth of the recursion tree? Express this as a function of $n$ and $c_1, c_2, c_3$. You do not need to prove correctness of your bound.
- We now consider a somewhat more general setting. Let $0 < c_1 \leq c_2 \ldots \leq c_k < 1$ be $k$ rationals such that $\sum_{i=1}^{k} c_i^2 < 1$. And $\gamma > 0$. Suppose we have a recurrence of the form

$$T(n) = \sum_{i=1}^{k} T(\lfloor c_i n \rfloor) + \gamma n^2, \quad n > 1/c_1, T(n) = 1 \quad n \leq 1/c_1.$$

  You can show that $T(n) = O(n^2)$ via induction as in the simpler case when $k = 3$. State the bound for $a$ in this more general setting and also the depth of the recursion as a function of $n, c_1, c_2, \ldots, c_k$. You do not need to prove correctness of your bound.

4. **For your reading, do not submit for grading:** Suppose $S$ is a set of $103$ integers. Prove that there is a subset $S' \subseteq S$ of at least $15$ numbers such that the difference of any two numbers in $S'$ is a multiple of $7$. *Hint:* See solved problem.

5. **For your reading, do not submit for grading:** Let $\Sigma$ be a *finite* alphabet and let $\mathcal{L}$ be the set of all *finite* languages over $\Sigma$. Prove that $\mathcal{L}$ is countable. Note that Cantor's diagonalization argument (review CS 173 material if you have forgotten about countability) shows that if $|\Sigma| \geq 2$ the set of all languages over $\Sigma$ is *not* countable.

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!

## Solved Problems

1. Suppose $S$ is a set of $n+1$ integers. Prove that there exist distinct numbers $x, y \in S$ such that $x - y$ is a multiple of $n$.

   **Solution:** We will use the pigeon hole principle. Let the $n+1$ numbers in $S$ be $a_1, a_2, \ldots, a_{n+1}$ and consider $b_1, b_2, \ldots, b_{n+1}$ where $b_i = a_i \mod n$. Note that each $b_i$ belongs to the set $\{0, 1, \ldots, n-1\}$. By the pigeon hole principle we must have two numbers $b_i$ and $b_j$, $i \neq j$ such that $b_i = b_j$. This implies that $a_i \mod n = a_j \mod n$ and hence $a_i - a_j$ is divisible by $n$.

   **Rubric:** 2 points for recognizing that the pigeon hole principle can be used. 2 points for the idea of using $\mod n$. 6 points for a full correct proof. Any other correct proof would also fetch 10 points.

   ■

2. Recall that the ***reversal*** $w^R$ 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 $\Sigma$.

   (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).

   In parts (b) and (c), you may assume without proof that $(x \cdot y)^R = y^R \bullet x^R$ and $(x^R)^R = x$ for all strings $x$ and $y$.

   **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 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 three cases in the definition):
   - 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 $w = w^R$.
- Suppose $w = axa$ for some symbol $a \in \Sigma$ and some palindrome $x \in P$. Then

$$\begin{aligned} w^R &= (a \cdot x \bullet a)^R \\ &= (x \bullet a)^R \bullet a && \text{by definition of reversal} \\ &= a^R \bullet x^R \bullet a && \text{You said we could assume this.} \\ &= a \bullet x^R \bullet a && \text{by definition of reversal} \\ &= a \bullet x \bullet a && \text{by the inductive hypothesis} \\ &= w && \text{by assumption} \end{aligned}$$

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 $y^R a = ya$.
  It immediately follows that $(y^R a)^R = (ya)^R$.
  Known properties of reversal imply $(y^R a)^R = a(y^R)^R = ay$ and $(ya)^R = ay^R$.
  It follows that $ay^R = ay$, and therefore $y = y^R$.
  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).

- No penalty for jumping from $aya = ay^R a$ directly to $y = y^R$.

■

**Rubric (induction):** For problems worth 10 points:

- + 1 for explicitly considering an *arbitrary* object
- + 2 for a valid induction hypothesis
- + 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 cases overlap (for example:
- + 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.