Homework 3

Due 8:00pm, May 28
(accepted until 9:59 pm, no credit after)

This homework has 8 questions, for a total of 100 points and 0 extra-credit points.
Unless otherwise stated, each question requires *clear*, *logically correct*, and *sufficient* justification to convince the reader.
For bonus/extra-credit questions, we will provide very limited guidance in office hours and on Piazza, and we do not guarantee anything about the difficulty of these questions.
We strongly encourage you to typeset your solutions in LaTeX.
If you collaborated with someone, you must state their name(s). You must *write your own solution* for all problems and *may not use any other student's write-up*.

A few note about this homework:

- This homework is due on **Tuesday, May 28 at 8pm**. The solution will be posted at 10pm on the same day.
- While some parts in this homework are optional and ungraded, **they are within the scope of the midterm**.
- After each question (or in some cases question part), we've indicated which lecture number we expect to cover the relevant material. So "(L8)" indicates that we expect to cover the material in lecture 8.

(0 pts) 0. **Before you start; before you submit.**

If applicable, state the name(s) and uniqname(s) of your collaborator(s).

Solution: None.

(10 pts) 1. **Self assessment.**

Carefully read and understand the posted solutions to the previous homework. Identify one part for which your own solution has the most room for improvement (e.g., has unsound reasoning, doesn't show what was required, could be significantly clearer or better organized, etc.). Copy or screenshot this solution, then in a few sentences, explain what was deficient and how it could be fixed.

(Alternatively, if you think one of your solutions is significantly *better* than the posted one, copy it here and explain why you think it is better.)

Solution:

Then the recurrence relationship is for all $m \in \{1, 2, \ldots, |A|\}$ and $n \in \{1, 2, \ldots, |B|\}$:

$$dis[m][n] = \min \begin{cases} dis[m-1][n-1] & \text{if } A[m] = B[n] \\ min \begin{cases} dis[m-1][n-1] + c_s \\ dis[m-1][n] + c_d \\ dis[m][n-1] + c_i \end{cases} & \text{if } A[m] \neq B[n] \end{cases}$$

We can justify the correctness of the recurrence relationship by induction.
Base case: When $m = 0$, the edit distance between the substring of A and B is the cost of inserting all n characters in the substring $B[0 : n-1]$. When $n = 0$, the edit distance between the substring of A and B is the cost of deleting all m characters in the substring $A[0 : m-1]$. These are optimal for sure.

Inductive step: Assume that for all $\{1, 2, \ldots, m\}$ and $\{1, 2, \ldots, n\}$, $dis[m][n]$ is the optimal edit distance between the substring of $A[0 : m-1]$ and $B[0 : n-1]$.
So there are two cases for $dis[m+1][n+1]$.
Case 1: $A[m+1] = B[n+1]$, then the char need not to be changed. So $dis[m+1][n+1] = dis[m][n]$.
Case 2: $A[m+1] \neq B[n+1]$, then three choices may produce the optimal edit distance for this problem: 1) change the char in A to the char in B, which costs c_s; 2) delete the char in A, which costs c_d; 3) insert the char in B to A, which costs c_i. So we pick

I think I did fair enough in hw 2, giving mostly correct answers, especially in the greedy algorithm part. I think I could have done better in the dynamic programming part. In problem 4, I took the fact that in the case when $A[i] = B[i]$, $ED(i-1, j-1)$ is the most efficient way but did not rigorously prove it. Next time I should think more carefully.

2. **Decision Problems and Languages.** (L8)

Recall that the goal of a *decision problem* is to determine whether a given input "object" has a certain "property", e.g., determine whether a given integer is prime, or whether a given string is a palindrome. In class, we said that any decision problem is equivalent to a *membership problem* for a corresponding *language*. That is, determining whether a given object has the property is equivalent to determining whether its encoding (as a string) is a member of the language.

For each of the following decision problems, (i) define a reasonable (finite) alphabet Σ, (ii) give the encoding (over the alphabet) of a representative input object, (iii) analyze the length of the encoding in terms of the value of the input, and (iv) define a language L for the corresponding decision problem. Use the notation $\langle X \rangle$ for the encoding of object X as a string over the alphabet. As examples, we have provided solutions to the first two parts.

(a) **Example:** "Does a given non-negative integer k have 3 as its last digit, when written in base 10?"

Solution:

(i) A reasonable alphabet consists of the decimal digits, $\Sigma = \{\texttt{0}, \ldots, \texttt{9}\}$. Alternatively, we could use the binary digits $\{0, 1\}$, or hexadecimal digits (0 through 9 and A through F), or the digits from any other base (including unary), or some entirely different-looking alphabet like $\{\star, \otimes, \perp\}$ (though the latter would not be very human-friendly). No matter the choice of (nonempty, finite) alphabet, there is a way to unambiguously encode (represent) non-negative integers as strings of characters from that alphabet.

Note that it would *not* be valid to include all of the non-negative integers in the alphabet, because an alphabet must be a *finite* set, and there are infinitely many non-negative integers.

(ii) Example encoding: For an integer k, write it in base 10 in the usual way, as a string of digits. For example, if k is the integer forty-seven, then $\langle k \rangle = \texttt{47}$. We stress that this is a *string* of the characters `4` and `7`, which *represents* the number forty-seven.

(iii) The length of this encoding would be the number of digits in the value, which is $\Theta(\log k)$.

(iv) A corresponding language would be $L_{EndsWith3} = \{\langle k \rangle : k \bmod 10 = 3\}$. It would also be acceptable to copy the phrasing of the decision problem, i.e., $L_{EndsWith3} = \{\langle k \rangle : k \text{ written in base 10 has 3 as the last digit}\}$

Alternatively, we could also encode integers in binary, using the alphabet $\Sigma = \{0, 1\}$. For example, if $k = 5$, then $\langle k \rangle = \texttt{101}$. The length of this encoding is still $\Theta(\log k)$. The above two definitions of the language hold without modification, because they both describe membership only in terms of the *value* of k, not its encoding.

(b) **Example:** "Is a given array of non-negative integers sorted?"

Solution:

(i) A reasonable alphabet Σ is the set of ASCII characters, or more selectively, the decimal digits along with some separator characters: $\Sigma = \{\texttt{0}, \ldots, \texttt{9}, \texttt{[}, \texttt{,}, \texttt{]}\}$. (Notice that a comma is one of these characters.)

Note that we can't just use the decimal digits alone if we also want some special symbols to indicate the start/end of the array and to separate the elements.

(ii) Example encoding: For an array A, encode its elements as in the previous part, and list those encodings with appropriate separators. For example, the array A with entries one, two, three, and four would have $\langle A \rangle = \texttt{[1,2,3,4]}$.

(iii) The length of this encoding is $\Theta(n + e)$, where n is the number of elements in the array, and e is the total length of the encodings of the elements.

(iv) The corresponding language is

$$L_{\text{sorted}} = \{\langle A \rangle : A \text{ is a sorted array of non-negative integers.}\}.$$

(4 pts) (c) "Given two strings compost of English alphabets, do they have a common subsequence of length k?"

Solution:

(i) A reasonable alphabet Σ is the set of 26 English letters both in their upper and lower case forms, together with a quotation mark for : $\Sigma = \{\texttt{a}, \ldots, \texttt{z}, \texttt{A}, \ldots, \texttt{Z}, \texttt{[}, \texttt{,}, \texttt{]}\}$.

(ii) Example encoding: For an array A, encode its elements as in the previous part, and list those encodings with appropriate separators. For example, the array A with entries one, two, three, and four would have $\langle A \rangle = \texttt{[1,2,3,4]}$.

(iii) The length of this encoding is $\Theta(n + e)$, where n is the number of elements in the array, and e is the total length of the encodings of the elements.

(iv) The corresponding language is

$$L_{\text{sorted}} = \{\langle A \rangle : A \text{ is a sorted array of non-negative integers.}\}.$$

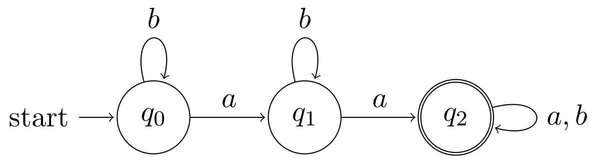
(4 pts) (d) "Does a given undirected graph G has a Hamiltonian cycle?" *Hint: Consider adjacency matrix.*

Solution:

3. **DFAs.** (L8)

For this problem, check out this tool to generate `tikzpicture` script for your DFA.

(4 pts) (a) Consider the following DFA:



Determine the (i) alphabet, (ii) state-transition function and (iii) language decided by the DFA. You may describe the language in English or regular expression, whichever you prefer.

Solution:

(5 pts) (b) Give a DFA that decides language L over the alphabet $\{a, b\}$ corresponding to the regular expression $a^*ba^*(ba^*ba^*)^*$. You may represent the DFA as a state-transition function or as a diagram, whichever you prefer.

Solution:

(6 pts) (c) Give a DFA that decides the following language over the alphabet $\Sigma = \{a, b\}$

$$L = \{s \in \Sigma^* : s \text{ contains at least two } a\text{'s } \textbf{and} \text{ odd number of } b\text{'s.}\}.$$

You may represent the DFA as a state-transition function or as a diagram, whichever you prefer.

Solution:

4. **Turing Machines Potpourri.** (L9, L11)

Determine whether the following statements about Turing Machines are always/ sometimes/ never true. While justification is **not required** to received for full credit, providing it is highly encouraged.

(3 pts) (a) A language that is decidable by a DFA is (always/ sometimes/ never) decidable by a Turing machine.

Solution:

(3 pts) (b) A Turing machine will (always/ sometimes/ never) halt before reading in its full input.

Solution:

(3 pts) (c) If a Turing machine does not halt on a given input, then the number of cells written to on its tape for that input is (always/ sometimes/ never) unbounded.

Solution:

(3 pts) (d) If a given Turing machine is guaranteed to only write to a finite number of cells on a tape, then it would (always/ sometimes/ never) halt.

Solution:

(3 pts) (e) A Turing machine can (always/ sometimes/ never) decide its own halting problem.

Solution:

5. **A Powerful Diagonalization.** (L10)

(7 pts) (a) The *power set* of a set A, denoted $\mathcal{P}(A)$, is defined as the *set of all subsets* of A. For example, $\mathcal{P}(\{0,1\}) = \{\emptyset, \{0\}, \{1\}, \{0,1\}\}$.

Use diagonalization to prove that for any *countably infinite* set A, the power set $\mathcal{P}(A)$ is *uncountably* infinite.

Hint: View each subset as an infinite binary sequence representing whether each element of A is included in the subset or not.

Solution:

(7 pts) (b) Prove that any infinite language $L \subseteq \Sigma^*$ has an undecidable subset $L' \subseteq L$.

Hint: Part (a) will be useful.

Solution:

6. **Decidability and Set Operations.** (L11)

Consider four languages where L_{D1} and L_{D2} are decidable, and L_{U1} and L_{U2} are undecidable. Determine whether the following languages are always/ sometimes/ never decidable. The case of $L_{D1} \cap L_{D2}$ is provided as an example. While justification is **not required** to received for full credit, providing it is highly encouraged.

(a) **Example:** $L_A = L_{D1} \cap L_{D2}$

Solution: Always. Let M_1 be a TM that decides L_{D1}, and M_2 be a TM that decides L_{D2}. We use these to construct the following Turing Machine that decides $L_{D1} \cap L_{D2}$:

```
$M'$ = on input ($x$):
1: Run $M_1$ on $x$ and run $M_2$ on $x$
2: If both $M_1$ and $M_2$ accepted $x$ then accept
3: Else reject
```

Analysis:

- $x \in (L_{D1} \cap L_{D2}) \implies x \in L_{D1} \wedge x \in L_{D2} \implies M_1$ accepts $\wedge\, M_2$ accepts $\implies$ M accepts
- $x \notin (L_{D1} \cap L_{D2}) \implies x \notin L_{D1} \vee x \notin L_{D2} \implies M_1$ rejects $\vee\, M_2$ rejects $\implies$ M rejects

Since M_1 and M_2 are deciders, they necessarily halt on all inputs. Thus, M' must also halt on all inputs. In addition, since M' accepts all $x \in (L_{D1} \cap L_{D2})$ and rejects all $x \notin (L_{D1} \cap L_{D2})$, it follows that M' decides $L_{D1} \cap L_{D2}$. As we have constructed a decider for $(L_{D1} \cap L_{D2})$, we can conclude that $(L_{D1} \cap L_{D2})$ is necessarily decidable.

(4 pts) (b) $L_B = L_{D1} \cup L_{D2}$

Solution:

(4 pts) (c) $L_C = L_{D1} \cap L_{U1}$

Solution:

(4 pts) (d) $L_D = L_{D1} \cup L_{U1}$

Solution:

(4 pts) (e) $L_E = L_{U1} \cap L_{U2}$

Solution:

(4 pts) (f) $L_F = L_{U1} \cup L_{U2}$

Solution:

7. **Can You Decide?**

For each of the following language, determine with proof whether the language is decidable or not. If decidable, describe and analyze a Turing machine that decides it. If undecidable, prove that reduce it from some language L of your choice that we have previously proved is undecidable.

(6 pts) (a) $L_{\text{ODD-5}} = \{x \in \{0, 1, 2, 3, 4\}^* : x \text{ represents an odd number in base } 5\}$. (L11)

Solution:

(6 pts) (b) $L_{\text{AND-LOOP}} = \{(\langle M \rangle, x, y) : M \text{ is a TM that loops on input } x \text{ and loops on input } y\}$ (L11)

Solution:

(6 pts) (c) Consider a language $L_{\text{I'MFINITE}}$ that contains a finite number of strings. Is $L_{\text{I'MFINITE}}$ (always/ sometimes/ never) decidable? Justify your answer. (L11)

Solution:

(d) *This part is optional and ungraded for this homework, but it's* **within the scope of the midterm**. *Therefore, we highly encourage you to attempt the problem.*
Define the language

$$L_{\text{FINITE}} = \{\langle M \rangle : M \text{ is a Turing machine and } L(M) \text{ is finite}\}.$$

Determine whether L_{FINITE} is decidable or undecidable. If decidable, describe and analyze a Turing machine that decides it. If undecidable, prove that $L \leq_T L_{\text{FINITE}}$ for some language L of your choice that we have previously proved is undecidable. (L12)

Solution: