Introduction to the Theory of Computation 2022 — Final Exam

Solutions

Problem 1 (15 pts). Consider

$$f(n) = e^{n^2} + 528n,$$

where $e \approx 2.71828$ is known as Euler's number.

(a) (5 pts) Prove

$$f(n) = 2^{O(n^2)}$$

by using the definition of big-O with $c = 2$.

(b) (10 pts) Prove

$$f(n) = o(4^{n^2})$$

by using the definition of limit. That is, you need to find out a real number n_0 for each $c > 0$.

Solution.

(a) Our target is finding a positive integer n_0 such that

$$\begin{aligned} &\log_2(e^{n^2} + 528n) \leq c \cdot n^2 = 2 \cdot n^2 \\ \Leftrightarrow &2^{2 \cdot n^2} \geq e^{n^2} + 528n \\ \Leftrightarrow &4^{n^2} \geq e^{n^2} + 528n \end{aligned} \tag{1}$$

for all $n > n_0$. The inequality (1) implies that

$$4^{n^2} = (3+1)^{n^2} \geq \underbrace{3^{n^2}}_{A} + \underbrace{n^2 \cdot 3^{n^2-1}}_{B} \tag{2}$$

For the term A, since $3 > e$, 3^{n^2} is always greater than e^{n^2} for all n. For the term B, we can pick $n_0 = 528$ such that

$$n^2 \cdot 3^{n^2-1} > 528n$$

holds for all $n > n_0 = 528$. Hence, we can derive (2) to

$$4^{n^2} > e^{n^2} + 528n$$

for all $n > n_0 = 528$. Therefore,

$$f(n) = 2^{O(n^2)}.$$

Common mistake: Some directly wrote, for example,

$$2^{2n^2} \geq 528n, \ \forall n \geq 3,$$

but this is exactly what we want you to prove.

(b) Given a real number $c > 0$, we can use (2) to help us to show that

$$e^{n^2} + 528n < c \cdot (3^{n^2} + n^2 \cdot 3^{n^2-1}) \leq c \cdot 4^{n^2},\ \exists n_0 \in \mathbb{R},\ \forall n > n_0. \tag{3}$$

Let us separate it to two parts:

(i) $e^{n^2} < c \cdot 3^{n^2}$. If $c \geq 1$, we can pick $n_0 = 1$. If $c < 1$, we can then find n_0 by

$$\begin{aligned} & n^2 < \log c + n^2 \cdot \log 3 \\ \Rightarrow & n^2(1 - \log 3) < \log c \\ \Rightarrow & n^2 > \frac{\log c}{(1 - \log 3)} \\ \Rightarrow & n > \sqrt{\frac{\log c}{(1 - \log 3)}} \end{aligned}$$

for which we can take

$$n_0 = \sqrt{\frac{\log c}{(1 - \log 3)}} + 1.$$

(ii) $528n < c \cdot n^2 \cdot 3^{n^2-1}$. This inequality implies

$$\frac{528}{n \cdot 3^{n^2-1}} \leq \frac{528}{n} < c.$$

Thus, we can take

$$n_0 = \frac{528}{c} + 1$$

such that

$$528n < c \cdot n^2 \cdot 3^{n^2-1}$$

holds for all $n > n_0$

Combine (bi) and (bii), we can pick

$$n_0 = \begin{cases} \max\left(1, \frac{528}{c} + 1\right) & \text{if } c \geq 1, \\ \max\left(\sqrt{\frac{\log c}{(1-\log 3)}} + 1, \frac{528}{c} + 1\right) & \text{otherwise.} \end{cases}$$

such that (3) holds.

Common mistake: You need to separately check the cases of $c \geq 1$ and $c < 1$.

Problem 2 (30 pts). Consider the language

$$L = \{a^i b^j c^k \mid i = j \text{ or } j = k, \text{ where } i, j, k \geq 0 \text{ are integers.}\}$$

We can design a two-tape machine with the following steps:

i) Go through all a until the first tape's head pointer is located at the first b. The second tape's head pointer stays at the first position.

ii) Copy all b into the second tape. After this step, the first and second tape's head pointers are located at the first c and the end, respectively.

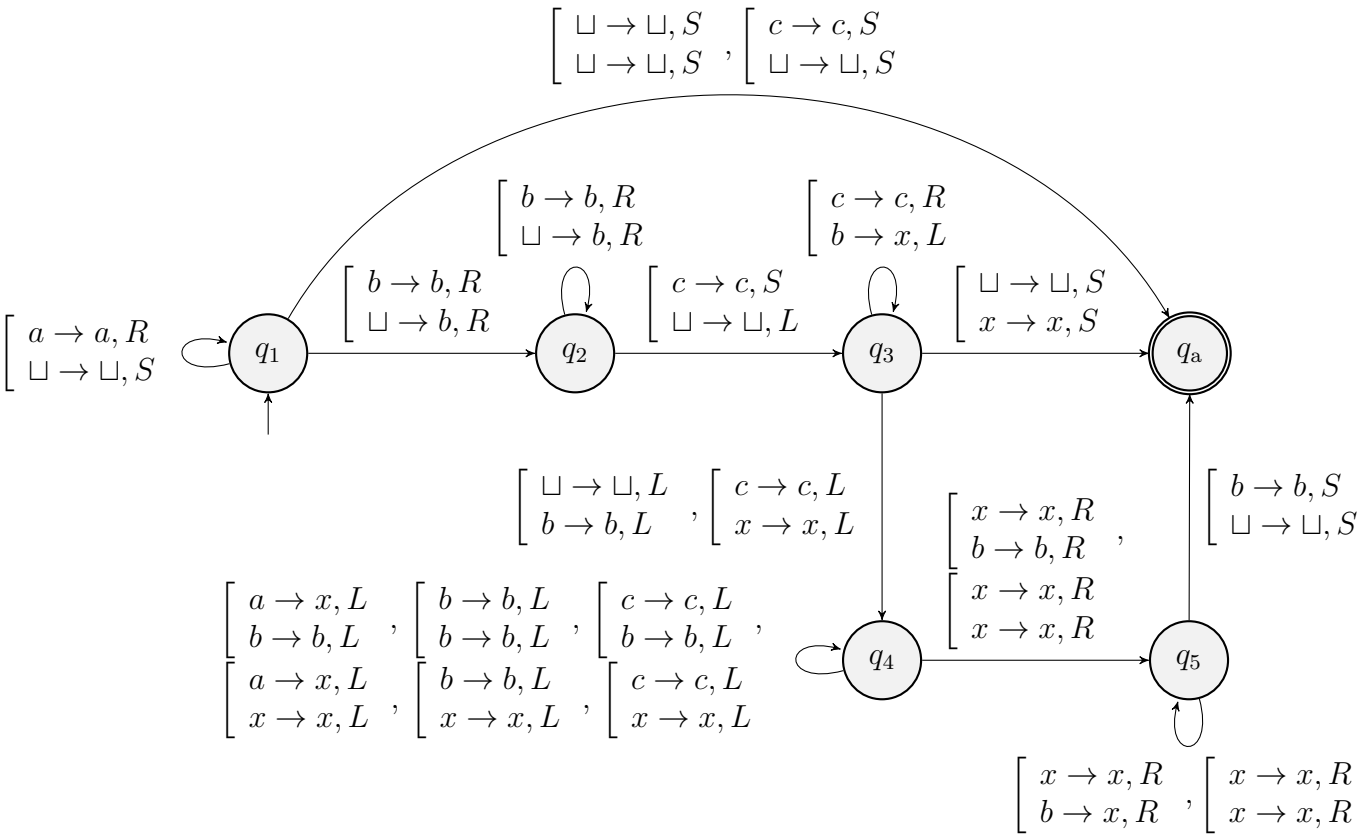
iii) Compare the numbers of c and b, where the first and second tape's head pointers move right and left, respectively. If $j = k$, accept the string. Note that you may mark a compared b as x in the second tape.

iv) In this step, we have known that $j \neq k$, so move both head pointers to the leftest location. Note that we need to mark a traversed a as x to make sure we are in the leftest location.

v) Compare the numbers of a and b. Remember we have marked some b to x in the step (iii), and have further marked all a to x in the step (iv). If $i = j$, accept the string.

vi) In the final, we focus on the special case $j = 0$.

Here is a (possibly incomplete) diagram of the TM:



In Problem 2, we assume the input order has been checked. For example,

ccbba

should not be considered.

(a) (5 pts) Please use the above diagram to simulate (with details) the string

aabbccc.

For easy grading, please also tell us the total number of needed operations.

(b) (10 pts) Does the TM really handle the case $j = 0$? If yes, please prove it. If not, please fix the diagram. You also need to explain the idea of your changes. For the diagram, you only need to show the changes. Further, you are allowed introduce at most one additional state.

(c) (10 pts) What is the time complexity of this two-tape TM? You need to show the details.

(d) (5 pts) Suppose that we can only use one-tape TM to decide L. Can you roughly give an idea for a new algorithm such that the time complexity is $O(n \log n)$? Is this $O(n \log n)$ the best complexity we can get? If yes, please show that your algorithm reaches the best big-O. If not, please prove it. You can directly use the knowledge that you learned in this course. Please give the description within roughly 100 words. Hint: L is not regular.

Solution.

(a) Here is the simulation:

$$\begin{array}{l} q_1\,a\,a\,b\,b\,c\,c\,c \\ q_1\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} a\,q_1\,a\,b\,b\,c\,c\,c \\ q_1\sqcup\ \ \sqcup\sqcup\sqcup\sqcup\sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} a\,a\,q_1\,b\,b\,c\,c\,c \\ q_1\sqcup\sqcup\ \ \sqcup\sqcup\sqcup\sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} a\ \ \ a\,b\,q_2\,b\,c\,c\,c \\ b\,q_2\sqcup\sqcup\ \ \sqcup\sqcup\sqcup\sqcup \end{array}$$

$$\Rightarrow \begin{array}{l} a\,a\ \ \ b\,b\,q_2\,c\,c\,c \\ b\,b\,q_2\sqcup\sqcup\ \ \sqcup\sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} a\ \ \ a\,b\,b\,q_3\,c\,c\,c \\ b\,q_3\,b\sqcup\sqcup\ \ \sqcup\sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} a\,a\,b\,b\,c\,q_3\,c\,c \\ q_3\,b\,x\sqcup\sqcup\sqcup\ \ \sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} a\,a\,b\,b\,c\,c\,q_3\,c \\ q_3\,x\,x\sqcup\sqcup\sqcup\sqcup\ \ \sqcup \end{array}$$

$$\Rightarrow \begin{array}{l} a\,a\,b\,b\,c\,q_4\,c\,c \\ q_4\,x\,x\sqcup\sqcup\sqcup\ \ \sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} a\,a\,b\,b\,q_4\,c\,c\,c \\ q_4\,x\,x\sqcup\sqcup\ \ \sqcup\sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} a\,a\,b\,q_4\,b\,c\,c\,c \\ q_4\,x\,x\sqcup\ \ \sqcup\sqcup\sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} a\,a\,q_4\,b\,b\,c\,c\,c \\ q_4\,x\,x\ \ \sqcup\sqcup\sqcup\sqcup\sqcup \end{array}$$

$$\Rightarrow \begin{array}{l} a\,q_4\,a\,b\,b\,c\,c\,c \\ q_4\,x\ \ \ x\sqcup\sqcup\sqcup\sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} q_4\,a\,x\,b\,b\,c\,c\,c \\ q_4\,x\,x\sqcup\sqcup\sqcup\sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} q_4\,x\,x\,b\,b\,c\,c\,c \\ q_4\,x\,x\sqcup\sqcup\sqcup\sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} x\,q_5\,x\,b\,b\,c\,c\,c \\ x\,q_5\,x\sqcup\sqcup\sqcup\sqcup\sqcup \end{array}$$

$$\Rightarrow \begin{array}{l} x\,x\,q_5\,b\,b\,c\,c\,c \\ x\,x\,q_5\sqcup\sqcup\sqcup\sqcup\sqcup \end{array} \Rightarrow \begin{array}{l} x\,x\,q_a\,b\,b\,c\,c\,c \\ x\,x\,q_a\sqcup\sqcup\sqcup\sqcup\sqcup \end{array}$$

The total number of needed operations is 17.

(b) Since the TM accepts the string

$$ac,$$

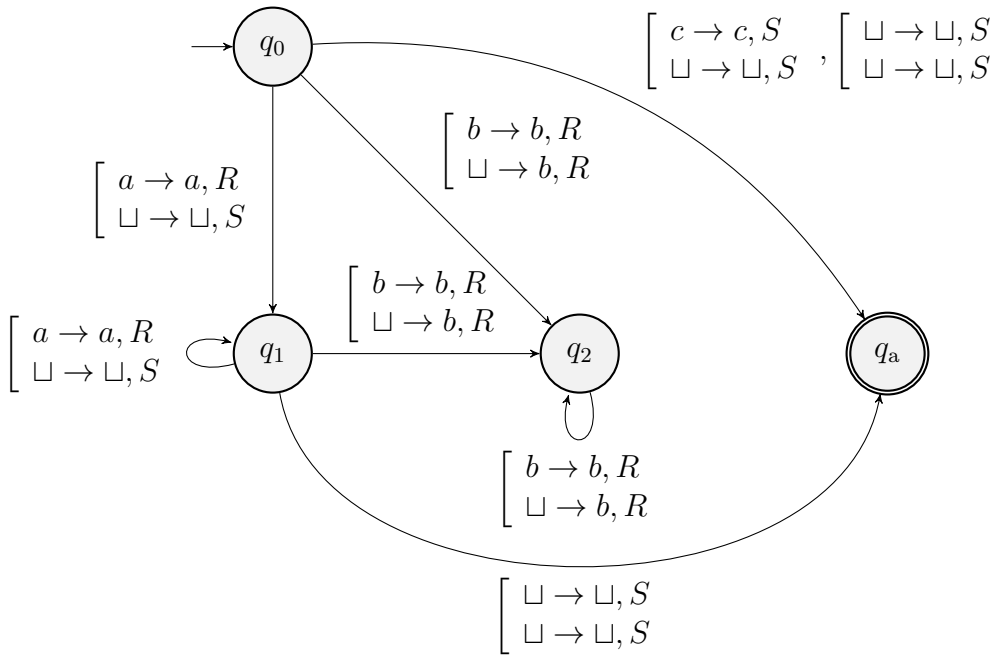
we know that this TM is wrong in the case $j = 0$. To fix the TM, first let us think what went wrong. The TM has

- $j = 0$.
$$q_1 \to \cdots \to q_1 \to q_a$$
- $j > 0$.
$$q_1 \to \cdots \to q_1 \to q_2$$

For ac, the first setting goes wrong. Recall the main task at q_1 is to handle a^i. Thus we introduce q_0 to share the task at q_1. At q_0, we consider two cases.

- $i > 0$. We have $q_0 \to q_1$ and run the original TM. A link to q_a remains for handling the situation of $j = k = 0$. Now ac is no longer accepted.
- $i = 0$. If $j > 1$, we have $q_0 \to q_2$ and run the original TM. Otherwise, $i = j = 0$, so we have links to q_a.

Here is the difference of the fixed diagram:



Some students argued that

bbccc

will be stucked by infinity loop in q_4. To fix this issue, we need to modify

$$\begin{bmatrix} b \to b, R \\ b \to b, R \end{bmatrix}, \begin{bmatrix} b \to b, R \\ x \to x, R \end{bmatrix}$$

to

$$\begin{bmatrix} b \to \tilde{b}, R \\ b \to b, R \end{bmatrix}, \begin{bmatrix} b \to \tilde{b}, R \\ x \to x, R \end{bmatrix}$$

in the self-loop of q_4, and change

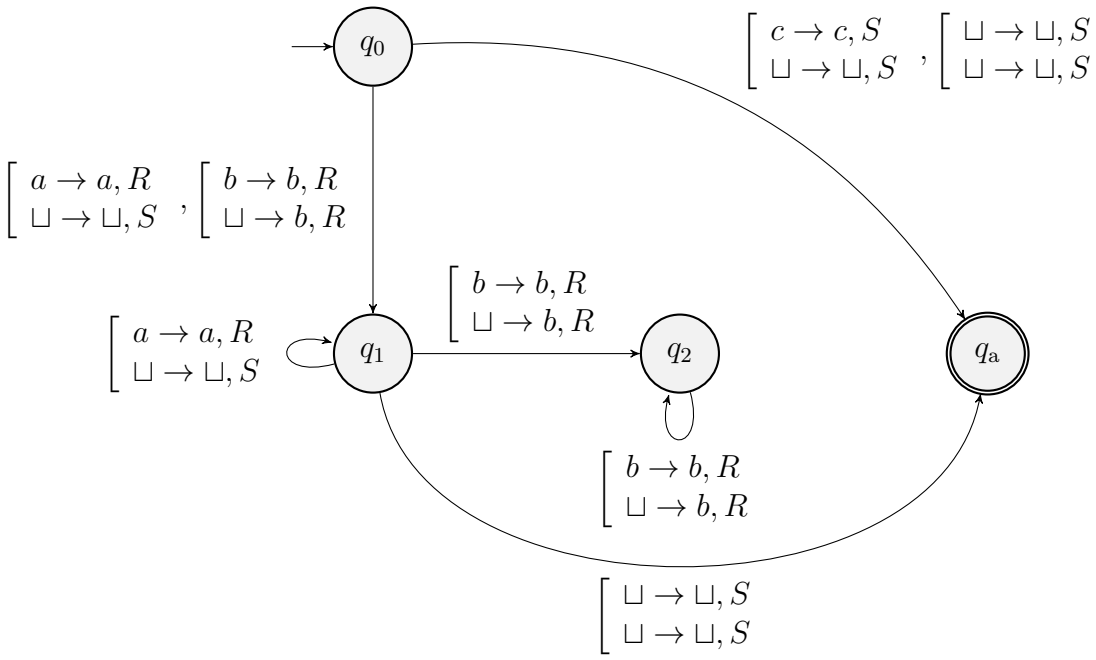
$$\begin{bmatrix} b \to b, S \\ \sqcup \to \sqcup, S \end{bmatrix}$$

to

$$\begin{bmatrix} \tilde{b} \to \tilde{b}, S \\ \sqcup \to \sqcup, R \end{bmatrix}$$

in the path q_5 to q_a. However, we have no time to correct the problem (this issue was reported at 12:50pm.) Also, although this issue is part of fixing the TM, we do not discuss it in the problem statement.

Some students instead do



This is fine because we have assumed that the input is

$$a^i b^j c^k.$$

Common mistake: You cannot just say that the diagram cannot handle the case of $j = 0$ without giving reasons.

(c) We can discuss two cases:

- $j \neq 0$. For the nodes q_0, q_1, q_2 and q_3, they scan the input string one time. If $j \neq k$, this TM will go back to the head of string by node q_4, which also traverses the string one time. In node q_5, it checks whether $i = j$, so the steps are less than the string length. Overall, the steps we need are less than three times of the input string length for deciding it, which is $O(n)$.
- $j = 0$. The nodes q_0 and q_1 scan the input string at most one time to do the decision, which is $O(n)$.

Therefore, the total complexity is $O(n)$.

Common mistake: The complexity needs to be a function of n, the input length.

(d) In the pages 7-9 of our slides "chap7_bigO2.pdf", we learned how to design a one-tape TM that decides the language $\{a^i b^j \mid i = j\}$ with $O(n \log n)$. Thus, given a string $a^i b^j c^k$, we can check whether $i = j$ with this $O(n \log n)$ algorithm first. If not, we restore all $\underline{b}$ to b, which requires $O(n)$. Then, check whether $j = k$ with this $O(n \log n)$ algorithm again, and the total complexity is still $O(n \log n)$. Moreover, because L is not regular, it implies "L cannot decide in $o(n \log n)$ on a single-tape TM." Therefore, $O(n \log n)$ is the best complexity that an algorithm for deciding L with a single-tape TM can reach.

Problem 3 (25 pts). Let $\Sigma = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, \cdot\}$, i.e., Σ contains all digits and **the dot character**. Consider the language

$$L = \{0 \cdot \boldsymbol{w} \mid \boldsymbol{w} \in \{0, 1, \ldots, 9\}^*\}.$$

At first glance, L looks like to be the set of real numbers between 0 and 1 if we regard each element in L as a floating point number. One student further proves that the set L is uncountable and gives a proof using the diagonalization method:
Proof. Assume for contradiction that L is countable, i.e., there exists a correspondence $f : \mathbb{N} \to L$, say

$$\begin{aligned} f(1) &= 0 \cdot \boldsymbol{w}_1 = 0 \cdot w_{11}w_{12}\dots w_{1|\boldsymbol{w}_1|} \\ f(2) &= 0 \cdot \boldsymbol{w}_2 = 0 \cdot w_{21}w_{22}\dots w_{2|\boldsymbol{w}_2|} \\ &\vdots \\ f(n) &= 0 \cdot \boldsymbol{w}_n = 0 \cdot w_{n1}w_{n2}\dots w_{n|\boldsymbol{w}_n|} \\ &\vdots \end{aligned}$$

where $\boldsymbol{w}_i \in \{0, 1, \dots, 9\}^*$ for all $i \in \mathbb{N}$ and $w_{ij} \in \{0, 1, \dots, 9\}$ for all $i \in \mathbb{N}$, $1 \le j \le |\boldsymbol{w}_i|$. We construct an element

$$0 \cdot \hat{\boldsymbol{w}} = 0 \cdot \hat{w}_1\hat{w}_2\dots\hat{w}_n\dots,$$

which is not paired with anything in $\mathbb{N}$. This is done by choosing $\hat{w}_i$ to be different from w_{ii}. If $|\boldsymbol{w}_i| < i$, which means there's no w_{ii}, we choose $\hat{w}_i$ as 1. Besides, we avoid selecting the digits 0 or 9 when constructing $0 \cdot \hat{\boldsymbol{w}}$ to make sure there's no issue like $0 \cdot 1999\dots$ and $0 \cdot 2000\dots$. With this construction, it's obvious that $0 \cdot \hat{\boldsymbol{w}} \neq f(n)$ for all $n \in \mathbb{N}$. Therefore, f is not a correspondence so that L is uncountable. □

(a) (10 pts) We've learned that Σ^* is countable, so it seems impossible that L, being a proper subset of Σ^*, is uncountable. Please point out the problem in the above proof with explanation. (*Hint: Think about the definition of the star operation.*)

(b) (5 pts) Can you represent each element in L as a fraction number? Denote L_{frac} to be the set of real numbers whose decimal representations (in the string format) are in L. Please write L_{frac} by a general mathematical form. That is,

$$L_{\text{frac}} = \{s \mid s = \frac{q}{p}, \text{ where } p \text{ and } q \text{ are certain values}\}.$$

(c) (10 pts) We'd like to show L is indeed countable by giving an explicit correspondence from $\mathbb{N}$ to L. (Here $\mathbb{N}$ starts from 1.) This can be done by listing strings in L according to the length of strings in ascending order. If two strings have the same length, list the one that represents smaller decimal number first (e.g., "$0 \cdot 0$" should be counted before "$0 \cdot 1$"). Moreover, in our counting, "$0 \cdot 0$" is considered different from "$0 \cdot 00$", "$0 \cdot 000$", ..., etc. Thus, each "$0 \cdot 0\dots0$" should be counted. For such a correspondence, which positive integer maps to the string "$0 \cdot 2022$"? That is, which n such that your $f(n)$ is "$0 \cdot 2022$".

Solution.

(a) The problem is that the constructed $\hat{\boldsymbol{w}}$ has an infinite length. It doesn't belong to

$$\{0, 1, \dots, 9\}^* = \bigcup_{k \ge 0} \{0, 1, \dots, 9\}^k.$$

Common Mistake: You must point out where the proof goes wrong.

(b) $L = \{s \mid s = \frac{q}{p}, \text{ where } p = 10^s \text{ for some } s \in \mathbb{N} \text{ and } 0 \le q \le p - 1. \}$

(c) We have

$$
\begin{aligned}
&f(1) = 0 \cdot \ \ , \\
&f(2) = 0 \cdot 0,\ f(3) = 0 \cdot 1,\ \ldots,\ f(11) = 0 \cdot 9, \\
&f(12) = 0 \cdot 00,\ f(13) = 0 \cdot 01,\ \ldots,\ f(111) = 0 \cdot 99, \\
&f(112) = 0 \cdot 000,\ f(113) = 0 \cdot 001, \ldots,\ f(1111) = 0 \cdot 999, \\
&f(1112) = 0 \cdot 0000,\ \ldots, f(3134) = 0 \cdot 2022,\ \ldots
\end{aligned}
$$

Problem 4 (30 pts). Consider the following ID masking problem that is related to the student list on our page of HW and exam scores.[1]

Let D be the set $\{0, 1, \ldots, 9\}$. Suppose we have a set of distinct student IDs, all with length l:

$$S = \{I^{(1)}, I^{(2)}, \ldots, I^{(C)}\} \text{ where } I^{(i)} \in D^l \text{ for } i = 1, 2, \ldots, C$$

We want to mask away some digits of every student ID, and the masked positions can be different for each student. However, we only deal with masking $I^{(1)}$ in this problem. Once we know how to mask $I^{(1)}$, we can apply the same algorithm to every other student independently. Therefore, considering only the masking of $I^{(1)}$ is adequate here.

Our goal is to keep only a few digits of $I^{(1)}$, say m digits, and mask away the remaining digits. When performing this masking, we need to make sure all other students' IDs are different from $I^{(1)}$ in at least one of these m digits. Otherwise, some students might mistakenly think that the masked $I^{(1)}$ is themselves.

Example 1. Suppose we have these student IDs:

$$
\begin{aligned}
S &= \{I^{(1)}, I^{(2)}, I^{(3)}\} \\
I^{(1)} &= 011 \\
I^{(2)} &= 002 \\
I^{(3)} &= 101
\end{aligned}
$$

We can mask away the first and the last digit of $I^{(1)}$ so that it is displayed as

$$\_1\_$$

on the webpage. The student with ID $I^{(1)}$ can recognize it as himself since the second digit matches his ID. On the other hand, students with ID $I^{(2)}$ or $I^{(3)}$ know that it's not them since their second digit is not 1. □

Let us restate our requirements more formally with the following definition. We will use I_j to denote the jth digit of a student ID I.

Definition 1. Given a set of IDs S and an ID $\tilde{I} \in S$. We say that *student $\tilde{I}$ can be uniquely identified with m digits* if there exists a set of indices $U \subseteq \{1, 2, \ldots, l\}$ satisfying

1. $|U| = m$
2. For all $I \in S \setminus \{\tilde{I}\}$, we have $\tilde{I}_j \neq I_j$ for some $j \in U$.

[1] https://www.csie.ntu.edu.tw/~cjlin/courses/comptheory2022/scores.html

□

Let us repeat Example 1 but with rigorous definitions.

Example 2. With the same student IDs from Example 1, we choose $U = \{2\}$ with $|U| = 1$, so we have $I_2^{(2)} \neq I_2^{(1)}$ and $I_2^{(3)} \neq I_2^{(1)}$. Therefore, $I^{(1)}$ can be uniquely identified with 1 digit. □

(a) (10 pts) Let us examine another example to familiarize ourselves with the problem. Assume we have these student IDs:

$$\begin{aligned} S &= \{I^{(1)}, I^{(2)}, I^{(3)}\} \\ I^{(2)} &= 55 \\ I^{(3)} &= 66 \end{aligned}$$

Show that we can pick an $I^{(1)} \in D^2$ such that the selected $I^{(1)}$ cannot be uniquely identified with 1 digit but can be uniquely identified with 2 digits. Specifically, to prove your result you need to select an $I^{(1)}$ and apply Definition 1 in your arguments like what we did in Example 2.

(b) We can further formally formulate the problem with the following language:

$$\begin{aligned} L = \{\langle \left(I^{(1)}, I^{(2)}, \ldots, I^{(C)}\right), m\rangle \mid C > 0, l \geq m > 0, I^{(i)} \in D^l \text{ for all } i = 1, 2, \ldots, C \\ \text{and } I^{(1)} \text{ can be uniquely identified with } m \text{ digits.}\} \end{aligned}$$

Notice that in our definition, the ID length l and the number of students C are not constants, but a variable that varies between strings in L.

We are going to design a polynomial time verifier V in subproblems (b1) and (b2) to show that $L \in$ NP. To discuss the algorithm design, let us make the following assumptions:

- Each ID $I^{(i)}$ is a string in $\{0, 1\}^l$. The same result should hold for general values as in subproblem (a), but we make this assumption for simplicity.
- Each ID $I^{(i)}$ is represented on the tape of TM by placing each digit sequentially in the tape cells.
- To represent m on the tape, we fill m tape cells with the tape symbol 1.
- To make the algorithm design easy, we further assume to have some marker symbols between, before and after the representations of $I^{(i)}$ and m.

(1) (5 pts) Design the certificate c to be used by your verifier. You are required to represent c as a string that is not longer than l cells (recall that l is the length of IDs).

(2) (10 pts) Write down your verifier algorithm **in clear steps of high-level descriptions**. Then, argue that your algorithm runs in polynomial time by giving the time complexity in big-O for each step of your algorithm (giving some arbitrarily large order is deemed incorrect). The time complexity should be calculated by running the algorithm on a single-tape TM and expressed in terms of the input length $n = |w|$, where w is the input string of the TM:

$$\langle (I^{(1)}, I^{(2)}, \ldots, I^{(C)}), m\rangle.$$

You can ignore checking the input format of w. Note that the verifier runs on $\langle w, c\rangle$

(c) (5 pts) Show that $L \in$ NP again, but by designing an NTM that uses the verifier V in subproblem (b2) as a subroutine. Show that it decides L in polynomial time by giving the time complexity for each step. Note that you need to also consider the generation of the certificate.

Solution.

(a) We can choose $I^{(1)}$ so that each of its digits is the same as some other IDs. For example, let

$$I^{(1)} = 56$$
$$I^{(2)} = 55$$
$$I^{(3)} = 66.$$

To uniquely identify $I^{(1)}$ with 1 digit, we must have

$$U = \{1\} \text{ or } U = \{2\}.$$

If $U = \{1\}$, we have $I_1^{(1)} = 5 = I_1^{(2)}$. If $U = \{2\}$, we have $I_2^{(1)} = 6 = I_2^{(3)}$. In both cases, $I^{(1)}$ is not uniquely identified so other students may confuse the masked $I^{(1)}$ with their own ID. It is clear that $I^{(1)}$ can be uniquely identified with 2 digits. Let $U = \{1, 2\}$. We have $I_2^{(1)} \neq I_2^{(2)}$ and $I_1^{(1)} \neq I_1^{(3)}$.

Common mistake: You need to show that it can be uniquely identified with 2 digits.

(b) (1) According to Definition 1, there exists a set U iff $I^{(1)}$ can be uniquely identified. Therefore, we can use the set U as the certificate. To represent it as a string, an intuitive way is to use a string of 0s and 1s:

$$c = c_1 c_2 \dots c_l \text{ where } c_i = \begin{cases} 1 & i \in U \\ 0 & i \notin U \end{cases}$$

This certificate is of the same length as $I^{(1)}$. Therefore, it has a length of $O(n)$ where n is the length of the input w.

Note: Some try to directly use U as the certificate. Then, $U \subseteq \{1, 2, \dots, l\}$ and l is a variable. However, our Σ is finite.

(2) Our verification works according to Definition 1. We loop over each position in U and make sure that every other student is different from $I^{(1)}$ in one of these positions.

On input $\langle w, c \rangle$:

i. Count the number of 1s in c to make sure that it is equal to m. Recall that we represented m as m cells of 1s on the tape. Therefore, we can apply the algorithm from page 7 of lecture slide "chap7_bigO2.pdf" to make sure the number of 1s in c is equal to m. We only have to slightly modify it to ignore the marker symbols and 0s in c. Therefore, this step can be done in $O(n \log n)$.

ii. Go through the certificate c and perform the following step for each c_j (repeats $|c| = O(n)$ times):

- Go through each $I^{(i)}$ and mark it if $c_j = 1$ and it is different from $I^{(1)}$ in the jth position. This can be done in $O(n)$.

This step takes $O(n^2)$ in total.

iii. Check whether $I^{(2)}, I^{(3)}, \dots, I^{(C)}$ are all marked. If yes, accept. Otherwise, reject. This can be done in $O(n)$.

Overall, the verification can be done in $O(n^2)$.

(c) On input $\langle w \rangle$:

i. Nondeterministically generate a certificate $c \in \{0,1\}^l$. Because l is not a constant, we can use $I^{(C)}$ as a reference. For each c_j being generated, we mark the jth digit of $I^{(C)}$. After all digits of $I^{(C)}$ are marked we know that c has been generated. Going back and forth between $I^{(C)}$ and c takes $O(n)$ and we have to perform it for $l = O(n)$ times. Therefore, this step takes $O(n^2)$.

ii. Run the verifier V on $\langle w, c \rangle$. From subproblem (b2), this step takes $O(n^2)$.

Overall, the NTM indeed decides L in polynomial time.

Common mistake: Many think that the cost of generating a certificate is $O(n)$. However, we do not see a way of doing it in $O(n)$.