Homework 4. Due November 14, 9:59PM.

CS181: Fall 2022

GUIDELINES:

- Upload your assignments to Gradescope by 9:59 PM.
- Follow the instructions mentioned on the course webpage for uploading to Gradescope very carefully (including starting each problem on a new page and matching the pages with the assignments); this makes it easy and smooth for everyone. As the guidelines are simple enough, bad uploads will not be graded.
- You may use results proved in class without proofs as long as you state them clearly.
- Most importantly, make sure you adhere to the policies for academic honesty set out on the course webpage. The policies will be enforced strictly. Homework is a stepping stone for exams; keep in mind that reasonable partial credit will be awarded and trying the problems will help you a lot for the exams.
- Note that we have a **modified grading scheme for this assignment**: A sincere attempt will get you 100% of the credit and a reasonable attempt will get you 50% for each problem. Nevertheless, please attempt the problems honestly and write down the solutions the best way you can - this is really the most helpful way to flex your neurons in preparation for the exam.
- All problem numbers correspond to our text 'Introduction to Theory of Computation' by Boaz Barak. So, exercise a.b refers to Chapter a, exercise b.

Remark: To receive full credit, first explain at a high-level how the machine works (e.g., how we described the machine for Palindrome/Minority of 0's in class). You should then provide a few more details about each step as to how you implement them **but** you don't have to dig in and write down all the individual transitions, etc. For example, it would be ok to say "Scan right until you reach end and enter state $****$". But saying "Pair up 0s and 1s until none are left" for the PALINDROME problem would not a valid solution. If you are not sure about this, contact the TAs or ask on edstem for clarification. (The idea is to make it easy for you to write down the solution without cumbersome notation and make it easy for the readers to grade the solutions.)

1. Design a TM that recognizes $L = \{x :$ number of 1's in x is at least thrice the number of 0's.$\}$. So for instance $1101, 11110, 011011111 \in L$, whereas $10, 11100 \notin L$. Do not use HOC here but describe the TM in pseudocode as we did in class for Maj. [1 point]

2. Show that there is a simple constant size TM (or program in your favorite language) $Fermat$ such that the program $Fermat$ terminates (on any input) if and only if the Fermat's last theorem is false (which we know it is not ... but don't assume that for this problem - just show equivalence). [1 point]

3. Exercise 9.5. Please replace NAND++ program with a Turing machine for the problem.

   In other words, prove that the following function $\mathsf{Finite} : \{0,1\}^* \to \{0,1\}$ is uncomputable. On input $P \in \{0,1\}^*$ (the description of the TM), we define $\mathsf{Finite}(P) = 1$ if and only if P is a string that represents a TM such that there are only a finite number of inputs $x \in \{0,1\}^*$ on which the TM outputs 1. [1 point]

   [Note that this problem needs material to be covered in the lecture on November 9th.]

   [Hint: One approach is to use a reduction from NOTHALTONZERO.]

4. Exercise 9.9. Please replace NAND-RAM program with a Turing machine for the problem. That is, consider the case where $F : \{0,1\}^* \to \{0,1\}$ takes two Turing machines P, M as input and $F(P, M) = 1$ if and only if there is some input x such that P halts on x but M does not halt on x. Prove that F is uncomputable. [1 point]

   [Note that this problem needs material to be covered in the lecture on November 9th.]

   [Hint: Use a reduction from HALTONZERO].

Practice problems. Do not submit.

1. Design a TM that computes the function $DecbyOne : \{0,1\}^* \to \{0,1\}^*$ that takes the binary representation of an integer as input, and returns the binary representation of the number minus 1. The answer should have the same the number of bits as the input (so you may end up padding with zeros if needed). So for instance, $Decbyone(1100) = 1011$, $Decbyone(0010) = 0010$, $Decbyone(1000) = 0111$. You can assume the input is greater than 0. Do not use HOC here but describe the TM in pseudocode as we did in class for Maj. [1 point]

2. This problem essentially shows that TMs can actually implement indexable arrays.

   Define a function $Ind : \{0,1\}^* \circ \{\#\} \circ \{0,1\}^* \to \{0,1\}$. That is the input is of the form $i\#x$ where $i \in \{0,1\}^*$, $x \in \{0,1\}^*$. We interpret i as the binary representation of an integer and $Ind(i\#x) = x[i]$. For example, $Ind(0\#101010) = 1$ as the bit in the 0'th position of x is 1. Similarly, $Ind(1\#101010) = 0$, $Ind(11\#101010) = 0$ as the bit in the third position of x is 1. (Remember indexing starts from 0.)

   Give a TM that computes Ind. That is, for instance, when the tape is loaded with $i\#x$, the TM ends with $x[i]$ on its tape. You can assume without loss of generality that the integer whose binary representation is i is less than the length of x (i.e., don't worry about border cases). Do not use HOC here but describe the TM in pseudocode as we did in class for Maj. [1 point]

   [Hint: You can use the idea behind Decbyone as a building ingredient. You can even give a two tape TM if that simplifies your pseudocode. Imagine keeping a separate head at the start of x, how many times do you have to move it to the right to get the right answer?]

grade the solutions.)

1. Design a TM that recognizes $L = \{x :$ number of 1's in x is at least thrice the number of 0's.$\}$. So for instance $1101, 11110, 011011111 \in L$, whereas $10, 11100 \notin L$. Do not use HOC here but describe the TM in pseudocode as we did in class for Maj. [1 point]

1. Scan to the right of the tape until "0" is found
2. If no "0" is found
    a) cleanup the tape and return 1
3. If a "0" is found
    a) if "0" is marked
        i) return to stage 1 and continue scanning
    b) if "0" is unmarked
        i) mark "0" as seen
        ii) scan tape until 3 unmarked "1"s are found
            - if found, mark the "1"s as seen, and go to stage 1
            - if not found, cleanup and return 0

2. Show that there is a simple constant size TM (or program in your favorite language) *Fermat* such that the program *Fermat* terminates (on any input) if and only if the Fermat's last theorem is false (which we know it is not ... but don't assume that for this problem - just show equivalence). [1 point]

```
import random

def isFermat(n)

    While True:
        a = randint(0, 1000)
        b = randint(0, 1000)
        c = randint(0, 1000)
        if (a**n + b**n) != c**n:
            return False
        else:
            return True

def main():
    n = 0
    while n < 1000:
        if isFermat(n) == False:
            return n
        n += 1
```

3. Exercise 9.5. Please replace NAND++ program with a Turing machine for the problem.

In other words, prove that the following function $\mathsf{Finite} : \{0,1\}^* \to \{0,1\}$ is uncomputable. On input $P \in \{0,1\}^*$ (the description of the TM), we define $\mathsf{Finite}(P) = 1$ if and only if P is a string that represents a TM such that there are only a finite number of inputs $x \in \{0,1\}^*$ on which the TM outputs 1. [1 point]

[Note that this problem needs material to be covered in the lecture on November 9th.]

[Hint: One approach is to use a reduction from NOTHALTONZERO.]

The proof is by reduction of NOTHALTONZERO. Suppose towards the, there was an Algorithm A such that $A(M) = \text{FINITE}(P)$ for every $P \in \{0,1\}^*$. Then we will construct an algorithm B that solves NOTHALTONZERO.

Given a turing machine M (which is the input to NOTHALTONZERO), our algorithm B does the following

1. Construct a TM M on which input $x \in \{0,1\}^*$, first runs $M(0)$ and then outputs 1.
2. Return $A(M)$

Now if M halts on input 1, then the TM M computes FINITE(P), and hence under our assumption that A computes FINITE(P), $A(M) = 1$.

If M does not halt on input 1, then the TM M will not compute FINITE(P). Hence under our assumption that A computes FINITE(P), $A(M) = 0$.

Henceforth, we see that $\text{FINITE}(P) = \text{NOTHALTONZERO}(M)$ and hence the value that Algorithm B returns in step 2 is equal to NOTHALTONZERO(M) proves FINITE(P) is computable, and reduced by NOTHALTONZERO.

However, we know that NOTHALTONZERO is uncomputable, therefore by association, FINITE(P) cannot be computed as well. That is, if FINITE(P) is solvable, NOTHALTONZERO would be too.

4. Exercise 9.9. Please replace NAND-RAM program with a Turing machine for the problem. That is, consider the case where $F : \{0,1\}^* \to \{0,1\}$ takes two Turing machines P, M as input and $F(P, M) = 1$ if and only if there is some input x such that P halts on x but M does not halt on x. Prove that F is uncomputable. [1 point]

[Note that this problem needs material to be covered in the lecture on November 9th.]

[Hint: Use a reduction from HALTONZERO].

We need to prove that every semantic non-trivial function F is at least as hard to compute as HALTONZERO (HOZ).

If a function is non-trivial, then there are 2 TMs P, M such that $F(P) = 0$ and $F(M) = 1$. The goal would be to take a machine N and map it onto a machine $Q = R(N)$, such that

i) if N halts on zero, then $Q = F(P) \neq F(M)$

ii) if N does not halt on zero, then $Q = F(M) = F(P)$

Assume an Algorithm A to compute $F(P,M)$. Then we can describe an Algorithm B that reduces A and computes $F(P,M)$ by the following

1. Construct machine N on input $x \in \{0,1\}^*$ do
   a) Run $N(0)$
   b) return $A(M)$
2. Return $1 - A(M)$

B outputs the right answer — suppose N does not halt on zero, then the program gets stuck into a loop and never reach step 2. $N = INF$ on this, but since algorithm B is fed into A, $F(N) = F(INF) = 0 \Rightarrow F(P) = F(M)$

If N does halt on zero, then $F(P,M) = 1 - F(N) = 1 - 0 = 1 \Rightarrow F(P) \neq F(M)$

Therefore, if N halts on zero $F(P) \neq F(M)$ and if it doesn't $F(P) = F(M)$. Henceforth, we see that $F(P,M) = \text{HALTONZERO}(P)$

However, we know that HALTONZERO is uncomputable, therefore by association, $F(P,M)$ cannot be computed as well.

That is, if $F(P,M)$ is solvable, HALTONZERO would be too.