

CS 181 Homework 4

Einar Balan

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 be 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 $L = \{x : \text{number of 1's in } x \text{ is at least thrice the number of 0's}\}$. 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]

Answer

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 true (which we know it is ... but don't assume that for this problem - just show equivalence). [1 point]

Answer

3. Exercise 9.5. Please replace NAND++ program with a Turing machine for the problem. [1 point]

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

Answer

4. Exercise 9.9. Please replace NAND-RAM program with a Turing machine for the problem. That is, consider the case where $F : \{0,1\}^* \rightarrow \{0,1\}$ 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]

[Hint: Use a reduction from HALTONZERO].

Answer

Practice problems. Do not submit.

1. Design a TM that computes the function $DecbyOne : \{0,1\}^* \rightarrow \{0,1\}^*$ 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 $Ind : \{0,1\}^* \circ \{\#\} \circ \{0,1\}^* \rightarrow \{0,1\}$. 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?]

3. Define a function $PowerTwo : 1^* \rightarrow 1^*$ that takes as input 1^n and outputs 1^{2^n} . For instance, $PowerTwo(1) = 11$, $PowerTwo(11) = 1111$, $PowerTwo(111) = 11111111$ and so on. (We do not care about non unary inputs.)

Describe a TM for computing $PowerTwo$. Describe the TM in pseudocode at a level of detail as we did in class for Maj. You may use multiple tapes.

[Hint: Try to have two tapes, and every time you move the head of the first tape (that contains the input), you do something on the second tape to double the length.]

4. Consider the function $EMPTY : \{0,1\}^* \rightarrow \{0,1\}$ that takes a DFA as input and outputs 1 if the language of the DFA is empty. That is, $EMPTY(D) = 1$ if D describes a DFA (under some encoding - the representation is not important) that does not accept any string. Define $EQUIVALENT : \{0,1\}^* \rightarrow \{0,1\}$ as the function that takes two DFAs D, D' and checks their equivalence: that is $EQUIVALENT(D, D') = 1$ if $D(x) = D'(x)$, $\forall x$. Give a reduction from EQUIVALENT to EMPTY. [1 point]

[You can use high-level programming languages or pseudocode to describe your reduction. By reduction, your goal is to give an algorithm R that takes an input for EQUIVALENT and outputs an input for EMPTY such that the condition for reduction holds: $R((D, D'))$ is a DFA such that $EQUIVALENT(D, D') = EMPTY(R(D, D'))$. As a hint, use the closure properties of DFAs to show that there is a DFA D'' such that D'' is empty if and only if D, D' are equivalent. Then, you can argue that the DFA D'' can be produced by an algorithm; you only have to provide high-level explanation for the latter.]