Homework 4. Due November 15, 9:59PM.

CS181: Fall 2021

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 twice the number of 0's.$\}$. So for instance $110, 1110, 011011 \in L$, 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]

The idea of the machine is similar to what we did in class with the main difference being that when you find a 0 you would skip two 1's.

Set $s = \mathsf{Check0}$.

(a) Scan to right until you find a 0.

(b) If No 0 found, cleanup tape and write 1 on tape (see below on how to do this).

(c) If Yes (a 0 is found):

i. Mark the 0 as "a" (seen).

ii. Go to start of input. (Up to the starting triangle).

iii. Scan input to right to find a 1.

A. If a 1 is found, mark it as "a". Scan the input to right to find a 1 again. If Yes (a second 1 is found), mark it as "a"; go to start of input and enter start state. If No, cleanup tape and write 0 on tape.

B. If No (a first 1 is not found), cleanup and write 0 on tape.

Cleanup, with write 1 is implemented as follows: First enter state 'Write1'.

(a) Keep going left while overwriting current symbol as $\emptyset$.

(b) Once start of tape is hit, move one symbol to the write and enter $Write1Halt$.

(c) From the state $Write1Halt$, write 1 and halt.

The cleanup, with write 0 is implemented in a similar way with 1 replaced with 0.

2. Exercise 7.4. You can assume that at the start of computation, the first tape is loaded with the input (as in class) and the second tape is empty. The output of the machine will be the output from the first tape only. Do not use HOC here but describe the TM in pseudocode as we did in class for Maj. [1 point]

[Hint: Try to use a special symbol (say, #) to concatenate the 'two tapes' and use special marked symbols to note the position of the two heads.]

Let the two-tape machine M_2. Suppose M_2 had k states and alphabet Σ. We will design a new machine M as follows. The alphabet of M will be the aplphabet of M_2 along with a special 'overline' version of each symbol in their alphabet (so like $0, 1, \emptyset, \overline{0}, \overline{1}, \overline{(\emptyset)}, \ldots,$ etc.,) and special symbols #.

(a) The machine M first puts its starting tape in the following format: $\triangle\overline{(x[0])}x[1]\cdots x[|x|-1]\#\overline{\emptyset}\#$.

The idea is that the part of the tape before the special symbol, # will be used as the first tape, the part between the two #'s will be used as the second tape, and the 'overline' symbols will be used as the location of the virtual heads.

(b) To simulate a move of the two-tape machine, M scans its tape from the start to identify the two symbols with overlines on them (the two 'virtual heads'). Then M makes a second scan to update the tapes according to the way that M_2's transition would. Concretely, M could have a special state for every possible $[k] \times \Sigma^2 \times L, R, S, H^2$ and upon reading the two symbols with overlines on them (that it remembers by entering

corresponding states), enters a state to remember the specific output of M_2. It then scans the tape from the start and updates the overline symbols according to this remembered state and updates the overline symbols according to the action.

(c) One problem with the above is if the first virtual head moves onto #, this action signifies that M_2 moved one of its heads onto a previously empty cell. So we write a blank symbol in place of # and shift the tape contents from this cell one unit to the right and continue as before.

Shifting the the contents to the right is accomplished by remembering the current symbol in the state, overwriting with previously remembered symbol, and moving to the right until you hit the last # (or an empty symbol after the #).

(d) Finally, to get the final output right, once M_2 halts, we move the head to the first overline symbol (to be equal to the virtual head), replace the overline symbol with a non-overline version and halt.

3. 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]

   (a) Scan right until you see the empty symbol.

   (b) You then scan left until you see a 1. If none is found, you enter ReachedZero to stop.

   (c) If a 1 is found, you enter a state that'll flip bits to the right until you see the empty symbol.

4. 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?]

   The basic idea is to keep decrementing the counter i until you reach 0, and for each decrement we move a 'virtual head' on the x part to indicate which bit we are reading. The implementation would be as follows: The alphabet of the machine would be $\{\triangle, 0, 1, \emptyset, o, i\}$. Here o will be used as a special version of 0, and i as a special version of 1.

(a) First setup the tape so that the first bit of $x[0]$ is replaced with o, i appropriately.

(b) Decrement i by 1. If you reach 0 (from decrementing) enter a FinalizeOutput state. FinalzeOutput would then scan the tape to the right until it finds a o or i and enter a Write0 or Write1 state respectively. Once in Write0 you cleanup the tape and write 0; from Write1 you cleanup the tape and write 1.

(c) If decrement doesn't reach 0, then we have to move our virtual counter to the right. Scan the tape to the right until you find a o or i. If a o is found, you replace it with a 0 and replace the next symbol to the right with a special version. If a i is found, you replace it with a 1 and replace the next symbol to the right with its special version. You then scan to the start of the tape and call decrement again.

Practice problem. Do not submit.

1. Show that computable functions are closed under the "concatenation" operation: If $F, G : \{0,1\}^* \to \{0,1\}$ are two functions, define a new function $H : \{0,1\}^* \to \{0,1\}$ with $H(x) = 1$ if x can be written as $x = x_1 \circ x_2$, $F(x_1) = G(x_2) = 1$; and $H(x) = 0$ if there is no way to break up x as such. Show that if F, G are computable, then so is H. [1 point]

   [Hint: You can give a multi-tape TM for H. That is, you can use two-tapes or three or even four tapes. You don't have to describe how these multi-tape TMs are equivalent to one-tape. Make sure, you give sufficient details to describe how to implement this high-level idea.]

   Let us use four tapes for convenience. Let M_F, M_G be the TMs for the two functions. We initialize the tapes as follows:

   (a) The first tape will just hold the input x.

   (b) The second tape will be initialized to have 1's upto the length of x. So you have $1^{|x|}$ sitting here.

   (c) The third and the fourth tape are empty.

   You repeat the following until the second tape is empty (i.e., the second head overwrites even the starting $\triangle$):

   (a) For each 1 on the second tape, copy a symbol from the first tape onto the third tape while moving both heads. Once you reach the end of 1's on second tape, you start copying the remainder of the first tape to the fourth tape.

   (b) Run M_F on the third tape, if the output is 1, run M_G on the fourth tape. If both outputs are 1, then enter Write1 state and proceed as before. That is, you cleanup the first tape to write 1 and halt.

   (c) If not, you reset the tapes as follows:

      i. Move the first tape head to the start of tape.

      ii. Mark the current cell second tape as $\emptyset$ and move to left.

      iii. Make the second and third tapes empty.

   If you didn't halt above, then you enter Write0 state, cleanup the first tape to write 0 and halt.