

York University
CSE 2001 – Unit 2 Models
Instructor: Jeff Edmonds

Read Jeff's notes. Read the book. Go to class. Ask lots of question. Study the slides. Work hard on solving these questions on your own. Talk to your friends about it. Talk to Jeff about it. Only after this should you read the posted solutions. Study the solutions. Understand the solutions. Memorize the solutions. The questions on the tests will be different. But the answers will be surprisingly close.

1. Loops: Do some computational problems require two loops or can all of them be accomplished by a single loop?
2. Suppose we want to design a TM to know whether a graph G is connected. What is the difference between G and $\langle G \rangle$? When do we use $\langle G \rangle$ and why is this done?
3. What is the difference between a computational problem that is a function vs one that is a language? If L is a language, what (if anything) do the following mean: $L(x)$, $x \in L$, $L = \emptyset$, $L = \{0, 1\}^*$, L is decidable.
4. (Answer in Slides) TM Transitions: In the slides we proved that a TM that is only able to write the characters $\{0, 1, b\}$ on the tape is as powerful as one that has a larger tape alphabet Σ . The simpler machine simulated the more complex one by grouping its cells into blocks of some fixed length so that each block contained a 0/1 code for the required character from Σ . Suppose this block size is four, input character 0 is encoded with 0000 and 1 with 0001. The simulation must start by inserting three zeros in front of every bit of the input. For example, on input $b1011bb\dots$, the output of this first phase should be $\dots bb0001000000010001bb\dots$. Note that we don't actually care if the the output is at the beginning of the tape as long as it is surrounded by blanks. Writing down TM descriptions is hard. Instead, I recommend starting by writing pseudo code using variables and loops. The only allowed actions are to read and write to the tape where the head is and to move the head to the left and right. I am also a big believer in *loop invariants*. This is a clear picture of what the tape looks like at the beginning of each iteration. To help, I will give you this code.

algorithm AddZerosBetweenEachDigit()

<pre-cond>: Assume head on the b before the input.

<post-cond>: Required output on tape after where the input had been

begin

 loop ***<loop-invariant>***: For some $i \in [0, n]$, the first i bits of the input have been blanked out. These bits have been copied into a block past the input separated by a single blank with three zeros inserted before each. The head is on the b before the $i+1^{st}$ input bit.

- 1: Move head right one
- 2: $a =$ bit at head (i.e. the $i+1^{st}$ input bit), Write a blank, Move head right
- 3: Move head right until it is at a blank (i.e. after remaining input)
- 4: Move head right one (i.e. into the output)
- 5: Move head right until it is at a blank (i.e. after produced output)
- 6-9: Write 0, head right; Write 0, head right; Write 0, head right; write a
- 10: Move head left until it is at a blank (i.e. after produced output)
- 11: Move head left one (i.e. into the remaining input)
- 12: if head is on a blank then exit loop (i.e. no more input)
- 13: Move head left until it is at a blank (i.e. after remaining input)

end loop
end algorithm

- (a) Your goal is to prove that you know how to translate this code into a TM. You do not need to give the full translation. That would be far too tedious. Suppose a program had 13 lines of code and two variables $x, y \in \{1..5\}$, then what Pooh bear should write on the wall is the current line number that is being executed and the value of x and of y . Hence for each $k \in \{1..13\}, x', y' \in \{1..5\}$, let $q_{\langle k, x', y' \rangle}$ denote the state in which the TM is on line k , x has the value x' , and y has the value y' . What are the states for the above program and how many of them are there? Give the general idea for how the transition function $\delta(q, c) = \langle q', c', direction \rangle$ is defined.
 - (b) Explain and give the parts of $\delta(q, c) = \langle q', c', direction \rangle$ needed for translating line 2 of the code, i.e. “ $a = \text{bit at head}$ (i.e. the $i+1^{\text{st}}$ input bit), Write a blank, Move head right”.
 - (c) Do the same for lines 3 and 4, i.e. “Move head right until it is at a blank” and “Move head right one (i.e. into the output)”.
 - (d) Do the same for line 9, i.e. “Write a ”.
 - (e) Bonus 10 marks if correct: The problem is to redo the previous problem, except the TM can only write 0 or 1. The tape starts with a two blanks before the input and an infinite number after it, but once a blank gets a 0/1 written in it, it will never be blank again. Hint: At first I was thinking that this could not be done, because the TM could not differentiate between whether what is written on the tape is the initial input or some markings to know that progress had been made. Then I saw how to do it. See if you can do it too.
5. **Turing Machine:** Write all the transition rules for a Turing Machine that solves the palindrome problem. A palindrome is a string that is the same if you reverse it. For example, 9235329 is an odd length palindrome and 923329 is an even length one. Your TM given a string of characters from $\{0, 1, \dots, 9\}$ will answer with either yes or no. The input will have a blank b at the beginning and end, i.e. $b9235329b$. The head will be at the first character. The loop invariant should be that either the TM has already halted because it has already discovered that the string is not to a palindrome or for some $i \geq 0$, the first i characters and the last i characters have been correctly matched and blanked out. What remains to check that the remaining string is a palindrome. For example, with $i = 2$ the tape would contain $bbb353bbb$. The head will be at the first remaining character. You are to first write the code as done in the slides and then translate this code into a TM.
6. (Answer in Slides) TM computing $J(I)$
- (a) The input consists two blocks each consisting of exactly a million 0/1 characters. The first is the binary description of a JAVA program J , the second of an input I . Describe, if possible, a simple TM for computing whether $J(I) = 1$, meaning that program J on this input halts and accepts or not, $J(I) = 0$. Focus on giving a meaningful name to each of the states of your machine. Describe the complete $\delta(q, c) = \langle q', c', direction \rangle$ function in all of its details.
 - (b) Redo the question when J is arbitrarily long, but I is still of this fixed length. Remember that lots of problems can be solved that have arbitrarily long inputs so if you want to say that this one is uncomputable, don't give this as your reason.
7. (Answer in Slides) Recall that a Universal TM is a TM $M_{universal}$, which when given input $\langle “M”, I \rangle$, simulates TM M on input I . You are to give a sketch of how such a $M_{universal}$ would be built. To make it easier, lets assume that M only has one tape and one head.
Hint: Because $M_{universal}$ is one fixed TM, it must have one fixed set of states $Q_{universal}$ and one fixed tape alphabet $\Sigma_{universal}$ (i.e. the set of characters c that it is allowed to write in each tape call.)
Hint: Fix $\Sigma_{universal} = \{ '0', '1', ' \langle', ' ', ' \rangle' \}$.
Hint: Let $M_{universal}$ have three tapes each with its own head.
When $M_{universal}$'s computation begins, the description of M will appear on $M_{universal}$'s first tape and I on its second.

- (a) We must assume that the TM M described in $M_{universal}$'s input has an arbitrarily large set of states and an arbitrarily large tape alphabet Σ_M .
 Explain how you would describe M on $M_{universal}$'s first tape. Remember this description must be encoded using characters from $\Sigma_{universal} = \{ '0', '1', '\langle', '\rangle', '\prime' \}$.
- (b) At the beginning of $M_{universal}$'s simulation of the t_M^{th} time step of M 's computation, $M_{universal}$'s first tape will still contain the description of M . What would be best to put on $M_{universal}$'s second and third tape? Remember what you have written there must be encoded using characters from $\Sigma_{universal} = \{ '0', '1', '\langle', '\rangle', '\prime' \}$.
 Also where will $M_{universal}$ three tape heads be.
- (c) We say that M can compute in one time step any function of “what it knows”, where “what it knows” is its current state and the character that its head currently on. More formally, if M is in state q and its head is seeing the character c , then $\delta_M(q, c) = \langle q', c', direction \rangle$ specifies M 's next state and next actions.
 Is it possible that $M_{universal}$ “knows” q , or c , or for that matter δ_M ?
- (d) Quickly sketch the actions taken by $M_{universal}$'s simulation of the t_M^{th} time step of M 's computation.
- (e) Let $T(|I|)$ denote the running time of M on I and let $|“M”|$ denote length of the description of M stored on $M_{universal}$'s first tape.
 What is the running time needed by $M_{universal}$ to simulate $M(I)$?
- (f) It is not fair that $M_{universal}$ is allowed three tapes but M only one. Suppose instead, $M_{universal}$ was only allowed one tape.
 Remember that a one tape TM $M_{universal}$ is able to simulate our three tape $M_{universal}$ by interweaving the cells of the three tapes on one tape or by having a larger tape alphabet $\Sigma_{universal}$ to include characters like $'0\langle 1'$ to signify that $'0'$ is on its first tape $'\langle'$ on its second and $'1'$ on its third.
 The challenge is that the one tape version of $M_{universal}$ has one not three heads.
 Show how the straight forward implementation requires $[O(|“M”|) + O(T(|I|))] \times T(|I|)$ time.
 Show how this can be improved to only $O(|“M”|) \times T(|I|)$ time with the smart trick of moving the “ M ” and the “ q ”.

8. (Solution in Slides)

Think carefully about the definitions of Turing Machines. For every state q that the machine might currently be in and character c that might be written in the cell of the tape pointed to by the head, we define $\delta(q, c) = \langle q', c', direction \rangle$ where q' is the next state, c' is the character to write, and $direction$ is the direction in which to move the head one cell.

For each of these three specified changes to the TM model, explain the effect it would have on the reasonableness and on the computational power of the model. Do one of:

- A: Either argue that the set of problems computable does not change because the new and the old versions of TM can simulate each other
- B: or show that the new model is weaker by giving a computable problem that no longer can be computed
- C: or show that the new model is stronger by giving an easy algorithm for an arbitrary uncomputable problem (for example the halting problem).

- (a) Instead of requiring the characters c that are written in each cell of the tape to be from a set of a fixed size (eg binary or ascii), allow the c and the c' in the above definition $\delta(q, c) = \langle q', c', direction \rangle$ to be arbitrarily large integers.

- (b) Change the model so that the TM “knows” the location of the head. More formally, $\delta(q, i, c) = \langle q', c', direction \rangle$ also depends on the index i of the cell that head is currently at.
- (c) Change the model so that the TM no longer has a head but can specify the index of a cell of the tape to read and to write to. The TM still has its input stored in the first n cells of the input and is only allowed a fixed/constant/finite number of states and alphabet size. Formally, $\delta(q, c) = \langle q', c', i, i' \rangle$ where c is the last value read, c' will be written to cell indexed by i and the next c will be read from that indexed by i' .