Turing Machines and Computability Homework

WU CS 465—Fritz Ruehr—Spring 2019

This homework assignment covers Turing Machines, but without too much emphasis on the technical details, rather more on the theoretical insights and philosophical issues they elicit. This style is representative of what will appear on the final exam; i.e., not much technical there, either, as it would be too time-consuming and error-prone.

---

1. Because a DFA simply reads across its input, a string say of length or size n, one symbol at a time, and then halts with its judgement (based on final states), we can say that it runs in time O(n). Can you put a similar O-notation bound (presumably a larger one) on the running time of a Turing Machine, when given some arbitrary input of length n? Why or why not?

2. Your roommate claims to have an interesting result in theoretical computing. He's found an interesting problem in graph theory that he says is related to the Halting Problem for Turing Machines, in the sense that one of the two problems is *reducible* to the other. Recall that "P is reducible to Q" means that *if* we could solve Q, *then* we could solve P—i.e., solving P is reduced to "just" solving Q.

   Which of the following claims by your roommate means they have discovered something interesting, and which one is no big news? Explain your answer.

   - "My graph theory problem is reducible to the Halting Problem!"

   - "The Halting Problem is reducible to my graph theory problem!"

3. Your friend James is a big Java buff and thinks it's the best programming language, but your other friend Simone thinks Haskell is far better. One day they come to you to resolve a dispute: Simone thinks she has a problem that can only be solved in Haskell. James disagrees, but doesn't have time during finals to prove he's right. Suggest a "thought experiment" involving compilers that will show James and Simone that neither one of them is right … so you can get back to your *own* finals!

4. "I've solved the Halting Problem for Java!" shouts your classmate. "How so?" you ask. "Well, I can just look and see if there is a while loop in the code; if there is, I'll just evaluate the condition and see if it's True. If it is, the loop will go forever, and then I can tell that it doesn't halt!" Without dissing them too badly, explain why *this strategy* will not solve the Halting Problem for Java programs. (I.e., it won't allow them to tell, for an arbitrary Java program and it's input, whether the program will halt on that input.)