CSC364 Summer 2002 – Homework 1

The following questions are assigned each week. You need only hand in those questions marked with a star * as part of your assignment, and only these questions will be marked. Make sure you include a completed and signed cover page as your first page.

Week 1

1. To illustrate how asymptotic notation can be used to rank the efficiency of algorithms, use the relations $\subset$ (strict subset) and $=$ (set equality) to put the *orders* of the following functions into a sequence:

   $$2n^5 \qquad 8^{\log_2 n} \qquad 12n^{2n} \qquad 3^{(n^2)} \qquad 2^{100^{50}} \qquad n^2 \log_2 n \qquad 3\sqrt{2n}$$

   Do not use the relation $\subseteq$. Prove your answers. You need not prove relationships implied by transitivity.

*2. (hand in) Prove or disprove each of the following statements.

   (a) $5n^2 - 6n \in \Theta(n^2)$
   (b) $n! \in O(n^n)$
   (c) $33n^3 + 4n^2 \in \Omega(n^2)$
   (d) $33n^3 + 4n^2 \in \Omega(n^3)$
   (e) $n^2 \log n \in \Theta(n^2)$
   (f) $n^2 \log n \in \Theta(n^3)$
   (g) $n^{1.001} + n \log n \in \Theta(n^{1.001})$
   (h) $n^3 2^n + 6n^2 3^n \in O(n^3 2^n)$
   (i) $n\sqrt{n} + n \log n \in O(n \log n)$
   (j) $n^{2^{100}} \in O(1.001^n)$

3. Let a^n denote a string of length n characters, each character being an a. Write a Turing Machine to recognize strings of the form $\mathtt{a}^n\mathtt{b}^{2n}$, where $\mathtt{a}$ and $\mathtt{b}$ are elements of the input alphabet Σ and n is some nonnegative integer. Explicitly give your machine's alphabet, set of states, and transition function. Prove that your TM is correct.

Week 2

*4. (hand in) Write a Turing Machine which determines whether p divides q for arbitrary positive integers p and q. Explicitly give the structure of your TM, your input and tape alphabets, input encoding, set of states (commenting on the meaning of each state as necessary), and transition function. Argue the correctness of your TM (you should convince the grader you are correct, though a full formal proof is not required).

5. For a language $\mathcal{L}$, we define an enumerator to be a Turing Machine which writes each string in $\mathcal{L}$ to its tape exactly once, strings separated by some symbol `#`. For finite languages the TM will halt, but for infinite languages the TM will continue generating new strings forever. Note that an enumerator does not take input, but may use part of the tape as work space.

   Consider $\mathcal{L}$ being the infinite set of all strings consisting of 0's and 1's. Order these strings lexicographically: order by length (all strings of length 2 come before any strings of length 3), then alphabetically (011 before 110). Write an enumerator TM to output (write to its tape) the sequence of strings formed by this ordering. Separate outputed strings by the `#` symbol. Your machine should output:

   `#0#1#00#01#10#11#000#001#` ... `#111#0000#0001#` ...

6. A *write-once Turing Machine* is a single tape TM that can alter each tape square at most once (including the input portion of the tape). Show that the write-once TM model is equivalent to the ordinary TM model. (Hint: consider the write-twice TM as a first step, and use lots of tape!)

Week 3

7. Show that if $\mathcal{L}$ is the language accepted by a k-tape, l-dimensional, nondeterministic Turing Machine with m heads per tape, then $\mathcal{L}$ is accepted by some ordinary deterministic Turing Machine with one single-dimensional tape and one head.

8. Suppose the tape alphabets of all Turing Machines are selected from some infinte set of symbols $a_1, a_2, \ldots$. Show how each Turing Machine may be encoded as a binary string.

*9. (hand in) Let C be a language. Prove that $C \subseteq \Sigma^*$ is Turing-recognizable (semi-decidable or recursively enumerable) iff a decidable (recursive) language D exists such that $C = \{x \mid \exists y \in \Sigma^* \text{ such that } x\#y \in D\}$, where $\# \notin \Sigma$.
That is, C is Turing-recognizable iff there is some decidable language D such that each string in C is the prefix of some string in D.

Week 4

*10. (hand in) Given a string $w \in \{0,1\}^*$, let $\bar{w}$ be the one's complement of w, i.e., $\bar{w}$ is the result of flipping each 0 to 1 and each 1 to 0 in w. Let

$$S = \{\langle M \rangle \mid M \text{ is a TM that accepts } \bar{w} \text{ whenever } M \text{ accepts } w\}.$$

Prove that S is undecidable.

11. Prove each of the following problems either decidable or undecidable.

    (a) Given a TM M, an input string w, and a number $k \geq 0$, does M use at most k tape squares on input w?

    (b) Given a TM M and an input string w, does there exist a $k \geq 0$ such that M uses at most k tape squares on input w? (That is, does M use a finite amount of tape on input w?)

12. Let us now consider the "real-word" problem of protecting our computers from viruses. We would like to build a filter (a virus checker) which will detect programs which are viruses before they are executed. Unfortunately you will show that no virus checker can detect all viruses without itself being a virus. Of course, we must formalize what we mean by these terms.

    Consider a modern computer which uses some fixed operating system, under which all programs run. A *program* can be thought of as a function from strings to strings: it takes one string as input and produces another as output. On the other hand, a program itself can be thought of as a string.

    By definition, a program P spreads a *virus* on input x if running P with input x causes the operating system to be altered, and it is *safe* on input x if this doesn't happen. A program P is said to be *safe* if it is safe on every input string.

    A *virus checker* is a program, perhaps called `IsSafe`, that when given input $\langle P, x \rangle$, where P is a program and x is a string, produces the output 'YES' if P is safe on input x and 'NO' otherwise.

    Prove that if the possibility of a virus exists – i.e., there is a program and an input that would cause the operating system to be altered – then there can be no virus checker that is both safe and correct.