Assignment 1

Do the following problems and exercises from the book. Note that the ordering reflects the order in which the relevant material is being covered by the course. Exercises are spread throughout a chapter, and problems can be found at the end of each chapter.

Exercise 1.7

(a) If the 2D TM runs for time $T(n)$, how large can the grid portion it accesses be?

(b) Show how you can map a $N \times N$ grid on a linear array, and calculate at most how much time it takes to move between grid positions, if they are mapped on the linear array.

Exercise 1.10

Hint: If we think of A as the infinite tape of the TM we build, then i is the current head position. Also note that the number of lines of code in the program is *fixed* (we are not required to design a TM that executes *all* programs, just a given) one). So, can we encode the program into the TM's states? Maybe something like $q_{label} := (label, \sigma ...)$ for the *label*'th code line

$$label : \texttt{If } A[i] \texttt{ equals } \sigma \texttt{ then } cmds$$

Exercise 1.15

(a) *Hint:* How many bits are needed to write an ASCII text of n characters in binary (given that ASCII has 256=2^8 characters)? What would be the answer if ASCII had $b \geq 2$ characters? Then what is the shrinking/blowup factor of the input size when you write it using an alphabet of size $b \geq 2$ instead of 2 (binary)? Consider b to be a constant that is a feature of the TM and has nothing to do with the input.

(b) *Hint:* First note that writing the input in unary (e.g., to write the number 5 you write 11111 in unary, instead of 101 in binary) doesn't fall in the previous case (here $b = 1$).

This question is actually the difference between *polynomial* and *pseudo-polynomial* running times. Think of the running time for the multiplication of two numbers $N \times M$. If we write each the input in binary, then its size is $\log N + \log M$, and a TM can run the elementary-school multiplication algorithm (let's call it Algorithm A) in, say, $O(\log N \log M)$ time, which is polynomial (on the input size). But if there is another algorithm (call it Algorithm B), that does multiplication in, say, $O(NM) = O(2^{\log N + \log M})$ time, then this algorithm runs in *exponential* time (on the size of the input).

But if the input is represented in *unary*, i.e., the input size is $N + M$, then Algorithm B *is also polynomial on the size of the input* (Algorithm A actually is now poly-logarithmic, much faster than polynomial). So, now that the `UNARYFACTORING` input is in unary, can you come up with a polynomial algorithm to compute it (which would be considered exponential if the input were not in unary)?

Exercise 2.9

Hint: If you have the solution to exercise 2.8, you're almost there...

Exercise 2.10

Hint: Write down the definitions of $L_1, L_2 \in NP$, and derive a TM that certifies $L_1 \cup L_2$. Same for $L_1 \cap L_2$.

Exercise 2.15

Hint: Use the fact that if $I \subseteq V$ is an independent set of G, then $V \setminus I$ is a vertex cover for G, and I is a clique for $\bar{G}$, the complement of G.

Exercise 2.32

(a) If input x to a TM is represented in binary, what is its size, and what is its size if it is represented in *unary*?

(b) If $L \in NEXP$ when its strings are represented in binary, show that $L_{unit} \in NEXP$, where

$$L_{unit} = \{1^x : x \in L\}.$$

(c) (Padding) If $L \in NEXP$, i.e., there is a NTM M that runs in time $2^{p(\log x)}$ for a *known* polynomial p and recognizes L, then in which class does language

$$L_{padded} = \{[w]_{unary} 1^{2^{p(\log w)}} : w \in L\}$$

belong to? Here $[w]_{unary}$ is the representation of binary number w in unary. (*Hint:* Note that L_{padded} is a unary language, and that you can use the NDTM for L on the binary representation of w, after you strip $[w]_{unary} 1^{2^{p(\log w)}}$ of the last $1^{2^{p(\log w)}}$ 1's.)

(d) Complete the proof of exercise 2.32.