Assignment 1

Introduction to Complexity Theory (CS 721)

Instructor: Vishwas Bhargava
IIT Bombay

Instructions. This assignment will be **tested in class on August 7**, following a discussion / doubt-clearing session. During the test you will be given **2 randomly chosen options** of a question, and you must attempt **1** of them.

- You may **re-pick** your questions once, for a **25% penalty**.
- **Solving on the board** earns **double points** (at most once per student).

All problem numbers below refer to Arora–Barak, *Computational Complexity: A Modern Approach* (https://theory.cs.princeton.edu/complexity/). In the 2007 web draft, exercises are numbered by section (§) within each chapter's Exercises; the corresponding §-numbers are given in brackets after each problem.

---

1. **Design a Turing machine** (*any one* of the following).
   Write out a Turing machine deciding *one* of the two languages below. Although a single tape suffices, you may use 2 or more tapes (finitely many) to make your construction cleaner. Be as descriptive as possible: give the states names that make sense, and clearly explain the role of each. Your machine must run in polynomial time, but it need not be especially efficient beyond that.

   (a) (Elementary arithmetic)

   $$L_1 = \{\, a^i b^j c^k \mid i \cdot j = k \text{ and } i, j, k \geq 1 \,\}.$$

   (b) (Element distinctness) The machine is given a list of strings over $\{0,1\}$ separated by #s, and must accept iff all the strings are distinct:

   $$L_2 = \{\, \#x_1\#x_2\#\cdots\#x_l \mid \text{each } x_i \in \{0,1\}^* \text{ and } x_i \neq x_j \text{ for each } i \neq j \,\}.$$

2. **Problem 1.5** (oblivious Turing machines). [web draft: Chapter 1, §8]
   A TM M is called *oblivious* if its head movements depend only on the input length and not on the input itself. That is, M is oblivious if for every input $x \in \{0,1\}^*$ and every $i \in \mathbb{N}$, the location of each of M's heads at the i-th step of execution on input x depends only on $|x|$ and i. Show that for every time-constructible $T : \mathbb{N} \to \mathbb{N}$, if $L \in \mathbf{DTIME}(T(n))$ then there is an oblivious TM that decides L in time $O(T(n)^2)$. Furthermore, show that there is such a TM that uses only two tapes: one input tape and one work/output tape.

3. **Problem 1.15** (representations and the class **P**). [web draft: Chapter 1, §17]
   Recall that normally we assume numbers are represented as strings using the *binary* basis: a number n is represented by the sequence $x_0, x_1, \ldots, x_{\lfloor \log n \rfloor}$ such that $n = \sum_{i=0}^{\lfloor \log n \rfloor} x_i 2^i$. More generally, for $b \geq 2$ the base-b representation $\langle n \rangle_b$ writes n as a sequence of digits in $\{0, \ldots, b-1\}$, each digit then encoded as a string of 0s and 1s. The *unary* representation $\langle n \rangle_{\text{unary}}$ is the string 1^n.

(a) Show that choosing a different base of representation makes no difference to the class **P**. That is, show that for every subset S of the natural numbers, if we define $L_S^b = \{\langle n \rangle_b : n \in S\}$, then for every $b \geq 2$, $L_S^b \in \mathbf{P}$ iff $L_S^2 \in \mathbf{P}$.

(b) Show that choosing the unary representation *can* make a difference, by proving that the following language is in **P**:

$$\textsc{UnaryFactoring} = \left\{ \langle \langle n \rangle_{\text{unary}}, \langle \ell \rangle_{\text{unary}}, \langle k \rangle_{\text{unary}} \rangle : \text{there is } j \in (\ell, k) \text{ dividing } n \right\}.$$

(This problem is not known to be in **P** under the binary representation.)

4. Cook reducibility (Chapter 2). [web draft: Chapter 2, §12]

The notion of polynomial-time reducibility used in Cook's paper was somewhat different: a language A is *polynomial-time Cook reducible* to a language B if there is a polynomial-time TM M that, given an *oracle* for deciding B, can decide A. (An oracle for B is a magical extra tape given to M, such that whenever M writes a string on this tape and goes into a special "invocation" state, then the string—in a single step!—gets overwritten by 1 or 0 depending upon whether the string is or is not in B.)

Show that the notion of Cook reducibility is transitive and that 3SAT is Cook-reducible to Tautology.

5. Exactly-One-3SAT and Subset Sum (Chapter 2). [web draft: Chapter 2, §16]

In the Exactly One 3SAT problem, we are given a 3CNF formula φ and need to decide if there exists a satisfying assignment u for φ such that every clause of φ has exactly one True literal. In the Subset Sum problem we are given a list of n numbers $A_1, \ldots, A_n$ and a number T, and need to decide whether there exists a subset $S \subseteq [n]$ such that $\sum_{i \in S} A_i = T$ (the problem size is the sum of all the bit representations of all numbers). Prove that both Exactly One 3SAT and Subset Sum are **NP**-complete.

6. Your favourite problem (open-ended, bonus).

Pick your favourite problem you have come across anywhere in computer science or mathematics. It may be *decisional* or *functional*; theoretical or applied; and its input can be anything at all—a graph, the pixels of an image, signals received from Mars, whatever you like. Then:

(a) Write it down as a language (or as a function), specifying precisely what the input is and what question is being asked / what is to be computed.

(b) Is it known to be in **P**? Is it **NP**-hard, or not? (Or is its status open?)

(c) Justify your answers. Give whatever argument, reduction, algorithm, or citation you can—even an informal but honest account of *why* is what we are after here.

Have fun with this one.