Computational Complexity

Reading guide and Homework for **Week 01**

1 Reading Guide

In class we have learned:

- How to encode objects using binary strings, and how this allows us to have a general definition of a computational problem. You should review this in §0.1 and §0.2 of the book.
- What is a Turing machine, and what is a Universal Turing machine. You should review this in §1.1, §1.2, §1.4, and §1.7 of the book.
- We have seen the first example of a lower-bound: the halting problem is uncomputable! You should review this in §1.5 of the book.
- How to measure the amount of time taken by a Turing machine, and what is how this gives us a notion of *time complexity* of a function $f : \{0,1\}^* \to \{0,1\}^*$. You should review this in §0.3 (Big-O Notation), §1.3 and §1.6 of the book.

There were several details that we covered very quickly, and which **must** be looked at carefully, in order to properly understand what is going on. This kind of familiarity will be necessary to understand what's being said in class.

2 Exercise guide

Before doing the homework, you should train yourself with the following exercises from the book. Solutions will be provided for the exercises marked with an asterisk (see the webpage).

- 0.1
- ∗ 1.1 for addition (to make sure you understand the formal definition of Turing machine)
- 1.1 for multiplication (at least argue at a high level that it can be done).
- ∗ 1.9 (random access memory vs tape memory)
- 1.10.
- ∗ 1.14 (a), (b) (to make sure you understand how to argue at a high level about basic algorithms)
- 1.14 (c), (d)
- ∗ The 3SUM problem is as follows: you are given as input a list $(x_1, \ldots, x_n)$ of natural numbers, with each $-n^3 \leq x_i \leq n^3$. The goal is to decide whether there exist three indices $i, j, k \in [n]$ such that $x_i + x_j + x_k = 0$.

  Show that 3SUM is in P.
- ∗ Show that 3SUM can be solved in time $O(n^2 \log n)$ by a RAM Turing machine (defined in Exercise 1.9).

3 Homework

You should turn in solutions for the following exercises before the beginning of the next class. You can turn in solutions electronically (to `bruno.loff+homework@gmail.com`), or in paper, in which case we will scan them ourselves. If you given them in paper, please respect the following rule, which is meant to make scanning easy:

Solutions should be given in separate (not stapled) a4 sheets of paper, and your name and number should appear clearly on every page.

You should feel free to discuss the exercises with your colleagues, but when you write down the answer, you must do it alone, without any help.

In all exercises, don't build the Turing machine explicitly, just provide a high-level (yet careful) argument of how it can be done.

1. An n-vertex graph is called d-regular if every vertex has degree exactly d (i.e. has d incident edges).

   Show that, for any fixed natural number $d \geq 1$, the problem of deciding whether a given n-vertex graph is d regular is in P.

   Then show that the problem of deciding whether G is d-regular, when given an n-vertex graph G and a number $d \in \{1, \ldots, n-1\}$ as input (so d is not fixed), is also in P.

2. A k-clique in an n-vertex undirected graph $G = ([n], E)$ is a set $S \subseteq [n]$ of $|S| = k$ vertices of G such that any pair of vertices in S has an edge between them ($\forall_{a,b \in S, a \neq b} \{a, b\} \in E$).

   Show that, for any fixed natural number $k \geq 2$, the problem of deciding whether a given n-vertex graph G has a k-clique is in P.

   How about the problem of deciding whether G has a k-clique, when given both G and k as input (so k is not fixed)?

3. Consider the problem where we are given a list $X = (x_1, \ldots, x_n)$ of natural numbers, where each x_i is bounded as $0 \leq x_i \leq n^3$, and we wish to *sort* the list, i.e., output a list Y with the same numbers,[1] but sorted in ascending order. Show that this problem can be solved in time $O(n \log n)$ on a multitape Turing machine (hint: lookup *merge sort*).

[1] Including multiplicities, i.e., if a number appears repeated k times in X, it should also appear k times in Y.