Undergraduate Complexity Theory **CMU 15-455, Spring 2017**

HOMEWORK 1
Due: 5:00pm, Thursday January 26

---

1. **(Know your course, know yourself.)**

   (a) (1 point.) What is the exact time and date that Gradescope "thinks" this homework is due?

   (b) (1 point.) What is the *actual* exact time and date at which this homework must be turned in to Gradescope to count as "not late"?

   (c) (1 point.) How many "late days" are you allotted in this course?

   (d) (1 point.) What are the day, time, and location of Ryan's weekly office hours?

   (e) (1 point.) In what semester did you take 15-251? Who were the instructors then?

   (f) (5 points; this problem will be read and graded by Ryan.) Write a short note about yourself. For full points, your note should include: at least one sentence describing what part(s) of computer science you're interested in; at least one sentence about what you hope to be doing after you graduate; and, at least one sentence stating a fact about yourself that qualifies as at least mildly interesting, such as your hometown or your favorite hobby/video game/pet/television show/food/neighborhood/subreddit.

2. **(Encodings.)** In this problem, all our encodings will use the binary alphabet $\Sigma = \{0, 1\}$.

   (a) (5 points.) The usual way to encode natural numbers $x \in \mathbb{N} = \{0, 1, 2, 3, \dots\}$ is by their binary representation, $\text{base}_2(x)$. For example,

   $$\langle 35 \rangle = \text{base}_2(35) = 100011, \qquad \langle 455 \rangle = \text{base}_2(455) = 111000111, \qquad \langle 0 \rangle = \text{base}_2(0) = 0.$$

   This has the slight disadvantage that not all strings are valid encodings; specifically, ε (the empty string) and any string starting with 0 are invalid. A typical way to handle this would be to allow leading 0's, and to let ε count as an encoding of 0. But this has the slight downside that every number has multiple encodings.

   Alternatively, there is a cute way to get a perfect bijective encoding from $\mathbb{N}$ to $\{0, 1\}^*$: given $x \in \mathbb{N}$, its encoding is formed by taking the string $\text{base}_2(x + 1)$ and deleting the first symbol (i.e., most signficiant bit, which is always a 1). Briefly prove that this indeed gives a bijection from $\mathbb{N}$ to $\{0, 1\}^*$; and, state an exact formula for the length of this encoding of x, as a mathematical function of x itself.

   (b) (5 points.) In class we discussed some ways to encode pairs of natural numbers (a, b) over the alphabet $\Sigma = \{0, 1\}$. For example, one could first encode them over $\{0, 1, \#\}$ as $\text{base}_2(a)\#\text{base}_2(b)$, and then one could encode each symbol in $\{0, 1, \#\}$ using two bits. If we informally say that $|\text{base}_2(n)| \approx \log_2 n$ then this scheme has $|\langle (a, b) \rangle| \approx 2 \log_2 a + 2 \log_2 b$ (plus a couple).

   Describe a different "reasonably simple" encoding scheme for pairs of natural numbers with the property that $|\langle (a, b) \rangle| \approx \log_2 a + \log_2 b + 2 \log_2 \log_2 a$ (or ever smaller, if you can!). You may be a bit informal.

3. **(Search-to-decision.)**

   (a) (5 points.) Suppose I give you an algorithm D that solves the following "decision problem": Given as input three positive integers $a \le b \le m$, the algorithm outputs yes/no depending on whether m has a prime factor p satisfying $a \le p \le b$. Now suppose you want to build a search algorithm F with the following property: Given three positive integers $a \le b \le m$, the algorithm outputs a prime factor p of m satisfying $a \le p \le b$ if one exists, else it outputs "no".

   Write high-level pseudocode for F using black-box calls to D. Your algorithm should not make more than about $\log m$ calls to D. You don't have to prove your algorithm is correct (hopefully it'll be pretty obvious), but at least say a couple words about the idea of the algorithm.

   (b) (5 points.) Let $f : \{0,1\}^* \to \{0,1\}^*$ which is "length-preserving", meaning that $|f(x)| = |x|$ for all $x \in \{0,1\}^*$. We can think of f as a computational function problem. On the other hand, let d be the following decision problem: given as input a string $x \in \{0,1\}^*$, an integer i satisfying $1 \le i \le |x|$, and a bit $b \in \{0,1\}$, decide yes/no if the ith bit of $f(x)$ is equal to b.

   Show that f and d are "easily interreducible" problems. That is, give simple high-level pseudocode for solving d given a black-box for solving f, and give simple high-level pseudocode for solving f given a black-box for solving d. (For the latter, on input x you should make at most $|x|$ calls to the algorithm for d.)

4. **(Programming in Turing Machine.)** Program a Turing Machine to reverse its input string. On input $x \in \{0,1\}^*$, it should halt with the tape containing the reverse of x and nothing else. Your tape alphabet may contain more symbols than 0, 1, $\sqcup$, if you want. More precisely...

   (a) (5 points.) Give an English-prose "implementation description" of your Turing Machine (cf. Chapter 3.3 of Sipser), at a similar level of detail to the several quotation-marked TM descriptions in Chapter 3.1 of Sipser.

   (b) (3 points.) Include a print-out of your actual TM code in the

   `http://morphett.info/turing/turing.html`

   format. (I assume you'll be using that website to test your code.)

   (c) (2 points.) Let $f : \{0,1,\ldots,9,a,b,c,\ldots,z\} \to \{0,1\}^2$ be defined by the following table:

0	00	6	10	c	00	i	10	o	00	u	10
1	01	7	11	d	01	j	11	p	01	v	11
2	10	8	00	e	10	k	00	q	10	w	00
3	11	9	01	f	11	l	01	r	11	x	01
4	00	a	10	g	00	m	10	s	00	y	10
5	01	b	11	h	01	n	11	t	01	z	11

   Let $x \in \{0,1\}^*$ be $f(\textit{your andrew id})$, a binary string between 6 and 16 symbols long. First, state what x is. Second, state the configuration your Turing Machine is in after it has completed 10 steps on input x. (If one of your states is named something like "goLeft", use the symbol q_{goLeft} in your configuration string.)