CMU 15-251 **Fall 2018**

MIDTERM 2 PRACTICE PROBLEMS

---

1. We recursively define a set $S \subseteq \mathbb{N} \times \mathbb{N}$ as follows.

   - $(0, 0) \in S$;
   - if $(a, b) \in S$, then $(a, b + 1) \in S$;
   - if $(a, b) \in S$, then $(a + 1, b + 1) \in S$;
   - if $(a, b) \in S$, then $(a + 2, b + 1) \in S$.

   Prove that $S = \{(x, y) \in \mathbb{N} \times \mathbb{N} : x \leq 2y\}$.

2. For this problem we will need to talk about polynomials. It is ok if you are not very familiar with them; we will tell you all you need to know. A *real polynomial* is something that looks like this:

   $$P(x) = 6.1x^3 - 2.8x^2 + 5x + 1.3.$$

   (Here it would be a bit more proper if we had written $6.1x^3 + (-2.8)x^2 + 5x^1 + 1.3x^0$.) In $P(x)$ above, x is the *variable*, the numbers 6.1, -2.8, 5, 1.3 standing next to the powers of x are the *coefficients* (which can be any real number), and the *degree* of $P(x)$ is the highest power of x appearing in the polynomial. In the above example, the degree is 3. The degree is always a natural number. (*Special case: The polynomial* 0 *does not have a well-defined degree.*) We say that $r \in \mathbb{R}$ is a *root* of polynomial $P(x)$ if you get 0 when you plug in $x = r$ and compute the result. Finally, the most glorious fact about real polynomials, which you may freely use, is:

   > If $P(x)$ is a (nonzero) real polynomial with degree $d \in \mathbb{N}$,
   > then it has at most d distinct real roots.

   If you have any questions about polynomials and their definitions, please do not hesitate to ask on Diderot, since they're not exactly the main point of this problem.

   One way mathematicians like to define *rational numbers* is as follows: "A real number is called *rational* if it is the root of some polynomial with integer coefficients and degree 1." (For example, the rational 22/7 is a root of $7x - 22$, which is a polynomial with integer coefficients and degree 1.) As you probably know, there exist reals that are *irrational* (i.e., not rational). The most famous one is $\sqrt{2}$; you probably have seen a proof that this is irrational that requires a little number theory.

   Mathematicians also made the following definition: "A real number is called *algebraic* if it is the root of some polynomial with integer coefficients (and any degree $d \in \mathbb{N}$)." (For example, $\sqrt{2}$ is algebraic because it is a root of $x^2 - 2$. This polynomial also shows that $-\sqrt{2}$ is algebraic!) A real number that is not algebraic is called *transcendental*.

   From the definition, it is not clear at all if transcendental numbers exist. In fact, in 1844, Liouville used some hard-core number theory methods to show that transcendental numbers indeed exist. This was a pretty big deal at the time!

   Flex your CS muscles and prove that there exists a transcendental number. You do not need to know any number theory or algebra for this problem!

3. In Turkish, the word *dededen* means "from grandfather". Suppose we wish to search for this word within some text. We will assume the text alphabet is $\Sigma = \{n, e, r, d\}$. Let

$$T = \{x \in \Sigma^* : x \text{ contains } dededen \text{ as a substring}\}$$

Draw a DFA M that decides T.

[For practice, you might test out your ideas on the following text:

$$nerederenderendedededededededededi$$

Well, actually that practice text ends in an i; just pretend it's an r. By the way, this text is a flattening of the following dramatic one-act play:

"Nerede rende?"

"Rende de dedede," dede dedi.

Which in Turkish means:

"Where's the grater?"

"The grater is also with grandfather," grandfather said.]

4. Consider the following languages over the alphabet $\{a, b\}$:

$$L = \{a^n w : w \in \Sigma^*, n \in \mathbb{N}^+, w \text{ contains at most } n \text{ } a\text{'s}\},$$
$$K = \{a^n w : w \in \Sigma^*, n \in \mathbb{N}^+, w \text{ contains at least } n \text{ } a\text{'s}\}.$$

Show that one of these languages is regular and the other is not. For the proof of regularity, all you need to do is draw or describe a DFA; a proof of correctness is not required. For the proof of irregularity, give a complete proof.

5. Fix $\Sigma = \{a, b\}$. Given a word w, we let filter(w) denote w with its even-indexed characters removed. For example, filter$(a) = a$, filter$(ab) = a$, filter$(abba) = ab$, and filter$(\epsilon) = \epsilon$. For a language $L \subseteq \Sigma^*$, define

$$\text{DOUBLE}(L) = \{x \in \Sigma^* : \text{filter}(x) \in L\}.$$

Show that if L is regular, then so is DOUBLE(L), by giving an explicit construction of a DFA that recognizes DOUBLE(L). Give a few sentences of explanation of why your construction recognizes DOUBLE(L) correctly.

6. (a) We say that a language L is *acceptable* if there exists a Turing Machine M such that $M(x)$ accepts for all $x \in L$ and $M(x)$ either rejects or loops for all $x \notin L$. Show that L is decidable if and only if L and $\overline{L} = \Sigma^* \backslash L$ are both acceptable.

   (b) Recall that the language HALTS $= \{\langle M, x\rangle : M \text{ is a TM and } M(x) \text{ halts}\}$ is undecidable. Show that HALTS is acceptable (and that therefore, the set of decidable languages is a strict subset of the set of acceptable languages).

   (c) Show that $\overline{\text{HALTS}}$ is not acceptable.

   (d) Show that every acceptable language L reduces to HALTS.

7. (a) Fix a Turing Machine M with input alphabet Σ and consider the language $L_M = \{x \in \Sigma^* : M(x) \text{ halts}\}$. Is there an M such that L_M is decidable?

(b) Fix a string $x \in \Sigma^*$ and consider the language $L_x = \{\langle M \rangle : M(x) \text{ halts}\}$. Is there an x such that L_x is decidable?

8. Recall that for $w \in \Sigma^*$, w^R denotes the reversal of w. For example $10111^R = 11101$. Show that

$$K = \{\langle M \rangle : M \text{ is a TM which accepts } \langle M \rangle^R\}$$

is undecidable.

9. Prove that ACCEPTS $\leq$ EQ.

10. Prove that $\log n = O(n^\epsilon)$ for any $\epsilon > 0$.

11. (a) Draw a Turing Machine M which decides the language $\{0^n 1^m : n, m \in \mathbb{N}\}$.
    (b) Determine the running time function $T_M(n)$ completely exactly.
    (c) Prove that $T_M(n) = \Theta(n)$.
    (d) Although we rarely consider "best-case running time", let's do it in this problem. Define

$$U_M(n) = \min_{\substack{\text{instances } x \\ \text{of length } n}} \{\#\text{ of steps } M \text{ takes on } x\}.$$

    Determine the function $U_M(n)$ exactly. (Probably it will involve "cases".)
    (e) Prove that $U_M(n) = \Theta(1)$.

12. Let $T(n)$ satisfy the following recurrence relation:

$$T(1) = c, \qquad T(n) \leq 3 \cdot T(n/5) + k \cdot n^4 \quad \text{for } n > 1,$$

where c and k are some constants that don't depend on n. You can assume n is power of 5, i.e. $n = 5^t$ for some $t \in \mathbb{N}$.

    (a) Consider the recursion tree corresponding to the above recursive relation. Determine the total number of nodes in the tree, in terms of n, using the $\Theta(\cdot)$ notation. Prove your claim.
    (b) Prove a tight upper bound on $T(n)$ using the big-O notation.

13. Describe a "linear-time reduction from multiplication to squaring". That is, suppose you are given access to black-box that, given a number B, returns B^2 to you. Show how to multiply two n-bit numbers using time $O(n)$ plus at most a constant (like, one or two or three) number of calls to the squaring black-box.

    (The point of this problem is to illustrate that if you didn't know Karatsuba's algorithm for doing faster-than-quadratic-time multiplication, and you were trying to discover such an algorithm, you could WLOG focus just on doing the special case of faster-than-quadratic-squaring. In fact, both Kolmogorov and Karatsuba knew this fact, and it helped Karatsuba discover his algorithm.)

14. Consider the following computational problem: given as input positive integers A and B with $B < A$, output True if A has a factor between B and $A-1$ (both inclusive), and output False otherwise. Suppose someone gives you an algorithm that solves this problem in polynomial time i.e., $O(n^k)$ time for some constant k, where n denotes the number of bits in the binary representation of A. You can also assume that someone gives you a polynomial-time algorithm for division (which outputs the quotient and the remainder).

(a) Consider the following problem: given as input a positive integer C, output the smallest factor of C that is not equal to 1. Given the above polynomial-time algorithms, show how to solve this problem in polynomial time (with respect to the input length). You do not have to give a proof of correctness for your algorithm. And you can describe it at a high level without even writing pseudocode. However, you should argue **carefully** why the running time is at most a polynomial in the input length.

(b) Consider the following problem: given as input a positive integer C, output its prime factorization (e.g., if $C = 12$, the output would be $(2, 2, 3)$). Show how to solve this problem in polynomial time. As before, you do not have to give a proof of correctness for your algorithm (which you can describe at a high level), but you should argue **carefully** why the running time is at most a polynomial in the input length.

15. Can we put a universal upper bound on the time complexity of all decidable languages? For instance, is it possible that all decidable languages can be decided by a TM with time complexity $O(n^k)$ for some constant k? If that sounds too ridiculous to be true, you are right. It is not true. But how would you prove it? Here is another variant of this type of question. Is it possible that there exists a constant c (maybe the constant is 15251) such that every language that can be decided in polynomial time can be decided by a TM with time complexity n^c? In this problem, we are interested in this question.

    For any $k > 0$, describe a language L_k which can be decided in polynomial time such that no Turing machine with time complexity at most n^k can decide L.

    Note: This question is hard without any hints. If you want a real challenge, you can try to solve it without any hints. Otherwise, take a look at the hint on the next page.

Hint: Review the proof that HALTS is undecidable and come up with a similar diagonalization argument. Obviously you cannot choose HALTS as your language L_k, so consider

$$L_k = \{\langle M \rangle : M(\langle M \rangle) \text{ halts in at most } n^{2k+1} \text{ steps}\}.$$

You may assume the following in your proof: There exists a universal TM U such that for every $\langle M, x \rangle$ where M is a TM and x is an input to M, $U(\langle M, x \rangle) = M(x)$. Furthermore, if $M(x)$ takes t steps, then $U(\langle M, x \rangle)$ takes at most t^2 steps.