Fall 2017 CIS 262

Automata, Computability and Complexity
Jean Gallier

Homework 9

November 28, 2017; Due December 11, 2017

Problem B1 (20 pts). (1) Prove that the set of composite natural numbers is listable (a natural number $n \in \mathbb{N}$ is composite if $n \geq 2$ and if n can be written as a product $n = n_1 n_2$ with $n_1, n_2 \geq 2$).

(2) Given that the set P of primes is known to be listable (say, by the result of Section 11.3), prove that the set P is actually computable (recursive).

Problem B2 (30 pts). Let $\Sigma = \{a_1, \ldots, a_k\}$ be some alphabet and suppose $g, h_1, \ldots, h_k$ are some total functions, with $g\colon (\Sigma^*)^{n-1} \to \Sigma^*$, and $h_i\colon (\Sigma^*)^{n+1} \to \Sigma^*$, for $i = 1, \ldots, k$. If we write $\overline{x}$ for $(x_2, \ldots, x_n)$, for any $y \in \Sigma^*$, where $y = a_{i_1} \cdots a_{i_m}$ (with $a_{i_j} \in \Sigma$), define the following sequences, u_j and v_j, for $j = 0, \ldots, m+1$:

$$
\begin{aligned}
u_0 &= \epsilon \\
u_1 &= u_0 a_{i_1} \\
&\ \vdots \\
u_j &= u_{j-1} a_{i_j} \\
&\ \vdots \\
u_m &= u_{m-1} a_{i_m} \\
u_{m+1} &= u_m a_i
\end{aligned}
$$

and

$$
\begin{aligned}
v_0 &= g(\overline{x}) \\
v_1 &= h_{i_1}(u_0, v_0, \overline{x}) \\
&\ \vdots \\
v_j &= h_{i_j}(u_{j-1}, v_{j-1}, \overline{x}) \\
&\ \vdots \\
v_m &= h_{i_m}(u_{m-1}, v_{m-1}, \overline{x}) \\
v_{m+1} &= h_i(y, v_m, \overline{x}).
\end{aligned}
$$

(i) Prove that

$$v_j = f(u_j, \overline{x}),$$

for $j = 0, \ldots, m+1$, where f is defined by primitive recursion from g and the h_i's, that is

$$\begin{aligned} f(\epsilon, \overline{x}) &= g(\overline{x}) \\ f(ya_1, \overline{x}) &= h_1(y, f(y, \overline{x}), \overline{x}) \\ &\vdots \\ f(ya_i, \overline{x}) &= h_i(y, f(y, \overline{x}), \overline{x}) \\ &\vdots \\ f(ya_k, \overline{x}) &= h_k(y, f(y, \overline{x}), \overline{x}), \end{aligned}$$

for all $y \in \Sigma^*$ and all $\overline{x} \in (\Sigma^*)^{n-1}$. Conclude that f is a total function.

(ii) Use (i) to prove that if g and the h_i's are RAM computable, then the function, f, defined by primitive recursion from g and the h_i's is also RAM computable.

Problem B3 (30 pts). *Ackermann's function* A is defined recursively as follows:

$$\begin{aligned} A(0,\, y) &= y+1, \\ A(x+1,\, 0) &= A(x,\, 1), \\ A(x+1,\, y+1) &= A(x,\, A(x+1, y)). \end{aligned}$$

Prove that

$$\begin{aligned} A(0,\, x) &= x+1, \\ A(1,\, x) &= x+2, \\ A(2,\, x) &= 2x+3, \\ A(3,\, x) &= 2^{x+3}-3, \end{aligned}$$

and

$$A(4, x) = \left.2^{2^{\cdot^{\cdot^{\cdot^{2^{16}}}}}}\right\}^{x} - 3,$$

with $A(4, 0) = 16 - 3 = 13$. Equivalently (and perhaps less confusing)

$$A(4, x) = \left.2^{2^{\cdot^{\cdot^{\cdot^{2^{2}}}}}}\right\}^{x+3} - 3.$$

Problem B4 (30 pts). Give a ram program computing the function, $f\colon \Sigma^* \to \Sigma^*$, given by

$$f(w) = w^R.$$

$(\Sigma = \{a, b\})$.

Problem B5 (20 pts). Prove that the following properties of partial recursive functions are undecidable:

(a) A partial recursive function is a constant function.

(b) Two partial recursive functions φ_x and φ_y are identical. More precisely, the set $\{\langle x, y\rangle \mid \varphi_x = \varphi_y\}$ is not computable (not recursive).

(c) A partial recursive function φ_x is equal to a given partial recursive function φ_a.

(d) A partial recursive function diverges for all input.

Problem B6 (30 pts). Given any set, X, for any subset, $A \subseteq X$, recall that the *characteristic function*, χ_A, of A is the function defined so that

$$\chi_A(x) = \begin{cases} 1 & \text{iff } x \in A \\ 0 & \text{iff } x \in X - A. \end{cases}$$

(i) Prove that, for any two subsets, $A, B \subseteq X$,

$$\begin{aligned} \chi_{A\cap B} &= \chi_A \cdot \chi_B \\ \chi_{A\cup B} &= \chi_A + \chi_B - \chi_A \cdot \chi_B. \end{aligned}$$

(ii) Prove that the union and the intersection of any two Diophantine sets $A, B \subseteq \mathbb{N}$, is also Diophantine.

(iii) Prove that the union and the intersection of any two listable sets $A, B \subseteq \mathbb{N}$, is also listable.

(iv) Prove that the union and the intersection of any two computable (recursive) sets, $A, B \subseteq \mathbb{N}$, is also a computable set (a recursive set).

Problem B7 (50 pts). Given an undirected graph $G = (V, E)$ and a set $C = \{c_1, \ldots, c_p\}$ of p colors, a *coloring* of G is an assignment of a color from C to each node in V such that no two adjacent nodes share the same color, or more precisely such that for evey edge $\{u, v\} \in E$, the nodes u and v are assigned different colors. A k-coloring of a graph G is a coloring using at most k-distinct colors. For example, the graph shown in Figure 1 has a 3-coloring (using green, blue, red).

The **graph coloring problem** is to decide whether a graph G is k-colorable for a given integer $k \geq 1$.

(1) Give a polynomial reduction from the **graph 3-coloring problem** to the **3-satisfiability problem for propositions in CNF**.

If $|V| = n$, create $n \times 3$ propositional variables x_{ij} with the intended meaning that x_{ij} is true iff node v_i is colored with color j. You need to write sets of clauses to assert the following facts:

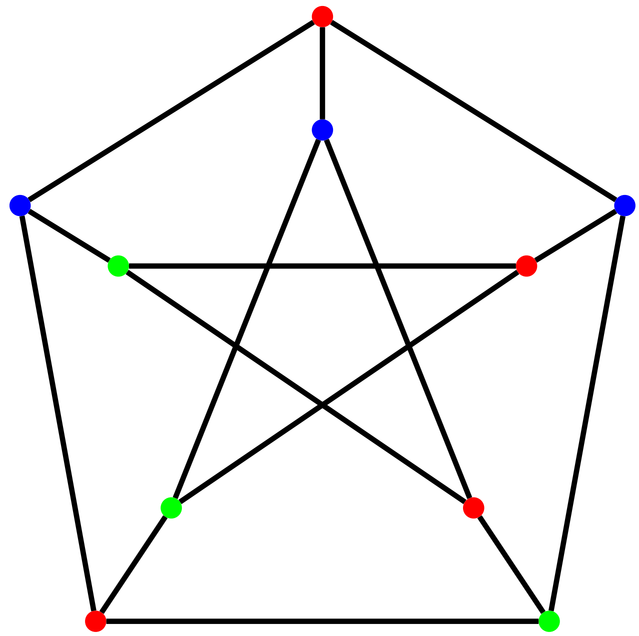

Figure 1: Petersen graph.

1. Every node is colored.
2. No two distinct colors are assigned to the same node.
3. For every edge $\{v_i, v_j\}$, nodes v_i and v_j cannot be assigned the same color.

Beware that it is possible to assert that every node is assigned one and only one color using a proposition in disjunctive normal form, but this is not a correct answer; we want a proposition in conjunctive normal form.

(2) Prove that 2-coloring can be solved deterministically in polynomial time.

Remark: It is known that a graph has a 2-coloring iff its is bipartite, but **do not** use this fact to solve B3(2). *Only use material covered in the notes for CIS262.*

The problem of 3-coloring is actually $\mathcal{NP}$-complete, but this is a bit tricky to prove.

Problem B8 (60 pts). Let A be any $p \times q$ matrix with integer coefficients and let $b \in \mathbb{Z}^p$ be any vector with integer coefficients. The 0-1 *integer programming problem* is to find whether a system of p linear equations in q variables

$$\begin{aligned} a_{11}x_1 + \cdots + a_{1q}x_q &= b_1 \\ \vdots \qquad & \quad \vdots \\ a_{i1}x_1 + \cdots + a_{iq}x_q &= b_i \\ \vdots \qquad & \quad \vdots \\ a_{p1}x_1 + \cdots + a_{pq}x_q &= b_p \end{aligned}$$

with $a_{ij}, b_i \in \mathbb{Z}$ has any solution $x \in \{0,1\}^q$, that is, with $x_i \in \{0,1\}$. In matrix form, if we let

$$A = \begin{pmatrix} a_{11} & \cdots & a_{1q} \\ \vdots & \ddots & \vdots \\ a_{p1} & \cdots & a_{pq} \end{pmatrix}, \quad b = \begin{pmatrix} b_1 \\ \vdots \\ b_p \end{pmatrix}, \quad x = \begin{pmatrix} x_1 \\ \vdots \\ x_q \end{pmatrix},$$

then we write the above system as

$$Ax = b.$$

(i) Prove that the 0-1 integer programming problem is in $\mathcal{NP}$.

(ii) Prove that the restricted 0-1 integer programming problem in which the coefficients of A are 0 or 1 and all entries in b are equal to 1 is $\mathcal{NP}$-complete by providing a polynomial-time reduction from the bounded-tiling problem. **Do not try to reduce any other problem to the 0-1 integer programming problem**.

Hint. Given a tiling problem, $((\mathcal{T}, V, H), \widehat{s}, \sigma_0)$, create a 0-1-valued variable, x_{mnt}, such that $x_{mnt} = 1$ iff tile t occurs in position (m,n) in some tiling. Write equations or inequalities expressing that a tiling exists and then use "slack variables" to convert inequalities to equations. For example, to express the fact that every position is tiled by a single tile, use the equation

$$\sum_{t \in \mathcal{T}} x_{mnt} = 1,$$

for all m, n with $1 \leq m \leq 2s$ and $1 \leq n \leq s$. Also, if you have an inequality such as

$$2x_1 + 3x_2 - x_3 \leq 5 \tag{$*$}$$

with $x_1, x_2, x_3 \in \mathbb{Z}$, then using a new variable y_1 *taking its values in* $\mathbb{N}$, that is, *nonnegative values*, we obtain the equation

$$2x_1 + 3x_2 - x_3 + y_1 = 5, \tag{$**$}$$

and the inequality $(*)$ has solutions with $x_1, x_2, x_3 \in \mathbb{Z}$ iff the equation $(**)$ has a solution with $x_1, x_2, x_3 \in \mathbb{Z}$ and $y_1 \in \mathbb{N}$. The variable y_1 is called a *slack variable* (this terminology comes from optimization theory, more specifically, linear programming). For the 0-1-integer programming problem, all variables, including the slack variables, take values in $\{0, 1\}$.

Conclude that the 0-1 integer programming problem is $\mathcal{NP}$-complete.

TOTAL: 270 points