CS244 Theory of Computation
Homework 3 Solution

Problem 1

The ***Kolmogorov complexity*** of a bit string b, $K_L(b)$, is the length of the shortest program in language L that outputs b and only b. Is $K_L(b)$ computable? Prove your answer.

Solution

$K_L(b)$ is uncomputable.

Lemma. *For each $n \in \mathbb{N}$, there exists a bit string b such that $K_L(b) > n$.*

Proof. Assume for contradiction that for some $n \in \mathbb{N}$, all the bit strings b have $K_L(b) \leq n$. Then $1 + 2 + 4 + \cdots + 2^n = 2^{n+1} - 1$ programs can generate infinitely many bit strings. However, each program can generate at most one bit string. □

Proof. We prove it for the Turing Machine language. It follows the same procedure for other languages. Assume for contradiction that $K_{\mathsf{TM}}(b)$ is computable. Then there is a TM M on input b, outputs $K_{\mathsf{TM}}(b)$. The following TM S outputs the bit string b with $K_{\mathsf{TM}}(b)$:

S:

1. Obtain via the recursion theorem, $\langle S \rangle$.
2. Go through each possible $b \in \{0,1\}^*$ in string order (ε, 0, 1, 00, 01, 10, 11, 000, ...) and run M on b to get $K_{\mathsf{TM}}(b)$. If $K_{\mathsf{TM}}(b) > |\langle S \rangle|$ (the length of S), output b and halt.

The TM S outputs b and only b, so $K_{\mathsf{TM}}(b) \leq |\langle S \rangle|$, which is a contradiction. Thus $K_{\mathsf{TM}}(b)$ is uncomputable. □

Problem 2

Let $SHUFFLE = \{\langle w, x, y \rangle \mid w = a_1 b_1 \dots a_k b_k \text{ for } k \geq 0 \text{ where } x = a_1 a_2 \dots a_k \text{ and } y = b_1 b_2 \dots b_k, \text{ each } a_i, b_i \in \Sigma^*\}$.

(a) Show that $SHUFFLE \in$ NP.

Proof. Construct a polynomial-time verifier V:

V on input $\langle \langle w, x, y \rangle, \langle a_1, a_2, \dots, a_k, b_1, b_2, \dots, b_k \rangle \rangle$:

1. Test whether each $a_i, b_i \in \Sigma^*$.
2. Test whether $w = a_1 b_1 \dots a_k b_k$.
3. Test whether $x = a_1 a_2 \dots a_k$.
4. Test whether $y = b_1 b_2 \dots b_k$.
5. If all tests pass, *accept*. Otherwise, *reject*.

□

Problem 2 continued on next page...

(b) Show that $SHUFFLE \in$ P.

Proof. The polynomial time algorithm M deciding *SHUFFLE* using dynamic programming is as follows:

M on input $\langle w, x, y\rangle$, where $w = w_1 w_2 \dots w_r$, $x = x_1 x_2 \dots x_s$, $y = y_1 y_2 \dots y_t$:

1. If $r \neq s + t$, *reject*.
2. Assign $table(0, 0) \leftarrow$ True.
3. For $i = 1$ to s:
4. Assign $table(i, 0) \leftarrow (table(i - 1, 0) \wedge (w_{i-1} = x_{i-1}))$.
5. For $j = 1$ to t:
6. Assign $table(0, j) \leftarrow (table(0, j - 1) \wedge (w_{j-1} = y_{j-1}))$.
7. For $i = 1$ to s:
8. For $j = 1$ to t:
9. Assign $table(i, j) \leftarrow ((w_{i+j-1} = x_{i-1}) \wedge table(i-1, j)) \vee ((w_{i+j-1} = y_{i-1}) \wedge table(i, j-1))$.
10. If $table(s - 1, t - 1) = True$, *accept*; otherwise, *reject*.

The total running time of the algorithm is $O(st)$, which is polynomial in the size of the input. Thus $SHUFFLE \in$ P. □

Problem 3

Let *SET-SPLITTING* $= \{\langle S, C\rangle \mid S$ is a finite set and $C = \{C_1, \dots, C_k\}$ is a collection of subsets of S, where the elements of S can be colored red or blue so every C_i has at least one red element and at least one blue element$\}$. Show that *SET-SPLITTING* is NP-complete.

Proof. First, we show that *SET-SPLITTING* $\in$ NP. Construct a polynomial-time verifier V:

V on input $\langle\langle S, C\rangle, P\rangle$, where P is the coloring scheme:

1. Test whether each element of S is colored either red or blue by P.
2. Test whether each C_i in C has at least one red element and at least one blue element.
3. If both pass, *accept*; otherwise, *reject*."

Next, we show that *3SAT* $\leq_{\text{P}}$ *SET-SPLITTING*. Given a 3cnf-formula ϕ, where there are m variables $x_1, x_2, \dots, x_m$, the reduction is as follows:

1. $S = \{x_1, x_2, \dots, x_m, \overline{x_1}, \overline{x_2}, \dots, \overline{x_m}, a\}$ (a is a new variable).
2. $C = \{\{x_1, \overline{x_1}\}, \{x_2, \overline{x_2}\}, \dots, \{x_m, \overline{x_m}\}\} \cup \{\{y_1, y_2, y_3, a\} \mid (y_1 \vee y_2 \vee y_3)$ is a clause in $\phi\}$.

($\rightarrow$) If ϕ is satisfiable, then there is an assignment making ϕ True and each clause contains at least one literal that is True. Color the elements of S such that all True literals are colored red and all False literals are colored blue; specially, color a blue. A variable can either be assigned True or False, so in each $\{x_i, \overline{x_i}\}$ there is exactly one red element and one false element. Also, in each $\{y_1, y_2, y_3, a\}$, at least one of y_1, y_2, y_3 is colored red, and at least a is colored blue. Thus this is a valid coloring scheme.

($\leftarrow$) If there is a valid coloring scheme for $\langle S, C\rangle$, assign False to all literals in the same color as a and assign True to all literals in the other color. At least one literal in each clause is True, and $\{x_i, \overline{x_i}\}$s guarantee that the assignment is not contradictory. Thus ϕ is satisfiable.

The reduction can be computed in polynomial time. $3SAT$ is NP-complete, thus *SET-SPLITTING* is NP-complete. □