Homework 5

Problem 1. Prove the following corollary of the time hierarchy theorem: for all constants $c_1 > c_0 \geq 1$, $\text{TIME}(n^{c_0}) \subsetneq \text{TIME}(n^{c_1})$.

Solution. A special note here: Not every constant c, the corresponding function n^c is time constructible. However, there exists some rational number q such that $q \in (c_0, c_1)$. Since q can be represented by finite bits, n^q is a time constructible function(More precisely, the rounded version of n^q). Since $\lim_{n\to\infty} \frac{\log n}{n^{q-c_0}} = 0$, we have that n^{c_0} is in $o(n^q/\log n)$, thus $\text{TIME}(n^{c_0}) \subsetneq \text{TIME}(n^q) \subseteq \text{TIME}(n^{c_1})$.

Problem 2.

(a) Show that P is closed under union, concatenation, and complement. That is, $A \cup B, A \circ B, A^{\text{c}} \in \text{P}$ if $A, B \in \text{P}$. Note that the concatenation $A \circ B$ of two languages A and B is defined as

$$A \circ B = \{xy \mid x \in A, y \in B\}.$$

(b) Show that P is closed under the star operation. That is, $A^* \in \text{P}$ if $A \in \text{P}$ where $A^* = \{x_1 x_2 \cdots x_k \mid k \geq 0, x_j \in A \text{ for } j = 1, 2, \ldots, k\}$. (Hint: You may need to use dynamic programming to maintain a table whose i, j-th entry indicates whether $x_i \cdots x_j \in A^*$)

Solution. (a) $A \cup B$: Given x, generate one copy, and simulate M_A and M_B, accept if one of them accept. Running time $O(T_A(n) + T_B(n))$.

$A \circ B$: Given x, it has $n + 1$ possible ways of division(Note that $\epsilon \circ x$ is also a valid division). For each division, run M_A on the first part, and M_B on the second part, accept if both accept. Running time $O(n(T_A(n) + T_B(n)))$.

A^{c}: Run A on x, accept if A rejects, and vice versa. Running time $O(T_A(n))$.

(b) Define f_i as the indicator for $x[1, \ldots, i] \in A^*$. We first set $f_0 = 1$, and we update f_i as follows:

$$f_i = \bigvee_{j=0}^{i-1} (f_j \wedge T_A(x[j+1, \ldots i]))$$

If in the end $f_n = 1$, we output 1, else output 0.

Thus the total running time is bounded by $O(n^2 T_A(n))$.

Problem 3. Karatsuba algorithm is an efficient algorithm for multiplying two natural numbers of $n = 2^k$ bits, outperforming the straightforward $O(n^2)$ primary-school method. The key idea is as follows. First, we write the numbers as $a2^\ell + b$ and $c2^\ell + d$ where $\ell = n/2$ and $a, b, c, d \in \{0, 1, \ldots, 2^\ell - 1\}$. So the product is

$$ac2^{2\ell} + (ad + bc)2^\ell + bd,$$

and this reduces the computation of the product to four multiplications (ac, ad, bc, bd) of shorter numbers. Second, Karatsuba's key idea is that three multiplications suffice for the computation as one can first compute ac, bd. Then the coefficient in front of 2^ℓ can be computed by one extra multiplication as $ad + bc = (a + b)(c + d) - ac - bd$. Show that Karatsuba's algorithm has time complexity $O(n^{\log_2 3}) \approx O(n^{1.585})$.

Solution. Note that at each step of recursion, each step of add would cost $O(n)$ time instead of $O(1)$ time. Easy to write out the recurrence relation $T(n) = 3T(n/2) + O(n)$. By Master theorem, $T(n) = O(n^{\log_2 3})$.