Home Work 2: Theory of Computation
October 26, 1998
Bud Mishra
Due On: November 19 1998

1 Time Complexity

Problem. 1
Order the following functions by their growth rate (the most slowly-growing function appearing first); if two functions are same, group them together.

(1)	1	(2)	7	(3)	$7^{\lg n}$
(4)	$(\lg n)^{\lg n}$	(5)	$\sqrt{n}\lg^2 n$	(6)	n
(7)	$n\lg n$	(8)	$n^{\frac{1}{\lg n}}$	(9)	$n^{\lg 7}$
(10)	$n^{1+\frac{\lg\lg n}{\lg n}}$	(11)	$n^{\lg\lg n}$	(12)	$\left(1-\frac{1}{n}\right)^n$
(13)	$\left(1-\frac{1}{7}\right)^n$	(14)	$\left(1+\frac{1}{n}\right)^n$	(15)	$\left(1+\frac{1}{7}\right)^n$

Solution. 1

(1)	$\left(1-\frac{1}{7}\right)^n$	$= 2^{-n\lg\frac{7}{6}}$
(2)	$\left(1-\frac{1}{n}\right)^n$	$= 2^{n\lg(1-\frac{1}{n})} = 2^{\lg e[n\ln(1-\frac{1}{n})]} \approx 2^{-\lg e}$
(3)	1	$= 2^0$
(4)	$n^{\frac{1}{\lg n}}$	$= 2^{\lg n\frac{1}{\lg n}} = 2^1$
(5)	$\left(1+\frac{1}{n}\right)^n$	$= 2^{n\lg(1+\frac{1}{n})} = 2^{\lg e[n\ln(1+\frac{1}{n})]} \approx 2^{\lg e}$
(6)	7	$= 2^{\lg 7}$
(7)	$\sqrt{n}\lg^2 n$	$= 2^{\frac{1}{2}\lg n + 2\lg\lg n}$
(8)	n	$= 2^{\lg n}$
(9)	$n\lg n$	$= n^{1+\frac{\lg\lg n}{\lg n}} = 2^{\lg n + \lg\lg n}$
(10)	$7^{\lg n}$	$= n^{\lg 7} = 2^{\lg 7\lg n}$
(12)	$(\lg n)^{\lg n}$	$= n^{\lg\lg n} = 2^{\lg n\lg\lg n}$
(13)	$\left(1+\frac{1}{7}\right)^n$	$= 2^{n\lg\frac{8}{7}}$

2 Context Free Grammar

Problem. 2
Exercise 2.4: e, f, g. (Sipser, pp 120).

Solution. 2 Omitted.

3 Pumping Lemma

Problem. 3
Exercise 2.18: c, d. (Sipser, pp 121).

Solution. 3 Omitted.

4 CFL and Dyck Language [HARD]

Problem. 4 A *Dyck Language* is a language with with k types of balanced parentheses. Formally, each Dyck language is, for some k, $\mathcal{L}_D = L(G_k)$, where G_k is the grammar:

$$S \to SS \mid [_1 S]_1 \mid [_2 S]_2 \mid \cdots \mid [_k S]_k \mid \epsilon.$$

Prove that every CFL $\mathcal{L} = h(\mathcal{L}_D \cap R)$, where h is a homomorphism, R a regular set and $\mathcal{L}_D$ a Dyck language.

Solution. 4 Assume that (without loss of generality) that $\mathcal{L}$ has a grammar with the productions in Greibach Normal Form:

$$A \to a_j \mid a_j V_1 V_2 \cdots V_N.$$

We define a right linear grammar for a regular set as follows:

$$(\{S\}, \{[_j,]_j | a_j = \text{terminal}\} \cup \{[_V,]_V | V = \text{nonterminal}\}, P, S)$$

where P is

$$\begin{array}{lll} S & \to & [_S S \mid \epsilon \\ S & \to & [_j]_j]_A [_{V_N} [_{V_{N-1}} \cdots [_{V_2} [_{V_1} S \\ & & \text{for each original production of the form } A \to a_j V_1 \cdots V_N \\ S & \to & [_j]_j]_A S \\ & & \text{for each original production of the form } A \to a_j \end{array}$$

Define the homomorphism $h : \{[_1, \ldots, [_k,]_1, \ldots,]_k\} \to \{a_1, \ldots, a_k\}$ as $h([_j) = a_j$ and $h(]_j) = h([_V) = h(]_V) = \epsilon$. If our Dyck language is derived by the grammar

$$S \to SS \mid [_k S]_k \mid \epsilon \quad \text{for all } k \in V \cup T,$$

Then the regular grammar will be able to derive the strings in $h^{-1}(\mathcal{L})$ and the Dyck language will put the constraint that it is a leftmost derivation.

For instance, the language $\{0^n 1 0^n | n \geq 1\}$ can be generated by the grammar:

$$S \to 0ST;\ S \to 1;\ T \to 0.$$

A string that belongs to the regular language

$$S \rightarrow \epsilon \mid [_S S \mid [_1]_1]_S S \mid [_0]_0]_T S \mid [_0]_0]_S [_T [_S S$$

and the corresponding Dyck language is

$$[_S [_0]_0]_S [_T [_S [_0]_0]_S [_T [_S [_0]_0]_S [_T [_S [_1]_1]_S [_0]_0]_T [_0]_0]_T [_0]_0]_T .$$

Applying the homomorphism we get 0001000.