University of Virginia - cs3102: Theory of Computation Spring 2010

PS6

Due: 27 April 2010 (2:00pm)

This problem set focuses on material from Chapters 7 and classes since Exam 2. You are strongly encouraged to use LaTeX to produce your submission, and we have provided a LaTeX template to assist with this. You may, however, handwrite your answers *so long as your writing is legible and easily interpreted.* For full credit, answers must be concise, clear, and convincing, not just correct.

Collaboration Policy. For this assignment, we will follow the same collaboration and resource policy as on Problem Sets 1-5. See the Problem Set 1 handout for a full description of the policy. Use collaboration to help you learn, but not in ways that prevent you from learning to solve problems on your own.

Problem 1: Asymptotic Notation. The way we advocated using asymptotic notation to describe algorithm complexity in Classes 22 and 23 is different from how Sipser uses it in Section 7.1. Identify at least two ways in which Sisper's use of the asymptotic operators differs from the way we advocated using them in class, and illustrate each with a specific example from Sipser's text.

Problem 2: Asymptotic Propositions. For each sub-part below, state whether the statement is **true**, **false**, or **unknown** (where unknown means no one on Earth knows the answer, not just that you don't know it!). Include a brief but convincing arugment supporting your answer.

a. $\Theta(n \log n) \subset O(n^2)$

b. $\exists f \in \mathcal{N} \rightarrow \mathcal{R}^+$ such that $\Omega(f) \cap O(f) = \emptyset$

c. set of languages that can be recognized by a Janus machine (as defined in Exam 2) in $O(n^3)$ Steps $\subseteq$ set of languages that can be recognized by a TM in $O(n^3)$ steps

d. set of languages that can be recognized by a machine (as defined in Exam 2) in $O(n^3)$ Steps $\subset$ set of languages that can be recognized by a TM in $O(n^3)$ steps

e. $n2^n \in O(2^n)$

Problem 3: Problem Complexity. Describe the asymptotic time complexity of deciding the language, $MULT = \{1^x * 1^y = 1^z \mid x, y, z \in \mathcal{N}, z = x \times y\}$, as precisely as you can. A good answer would provide the tightest big-O and Ω bounds you can. A correct Θ bound for the stated problem (that is, an argument that there is no asymptotically faster algorithm exists) is worth bonus points, but be aware that an incorrect answer us worse than a correct but less precise answer.

Problem 4: Unary Subset Sum. (Sipser's 7.16) Let *UNARY_SSUM* be the subset sum problem in which all numbers are represented in unary.

a. Why does the NP-completeness proof for *SUBSET_SUM* fail to show *UNARY_SSUM* is NP-complete?

b. Show that $UNARY\_SSUM \in$ P.

Problem 5: Genome Assembly. In order to assemble a genome, it is necessary to combine snippets from many reads into a single sequence. The input is a set of n genome snippets, each of which is a string of up to k symbols. The output is the smallest single string that contains all of the input snippets as substrings. For example, if the input is

$$\{\texttt{ACCAGAATACC}, \texttt{TCCAGAATAA}, \texttt{TACCCGTGATCCA}\}$$

the output should be `ACCAGAATACCCGTGATCCAGAATAA`:

```
ACCAGAATACC
                TCCAGAATAA
       TACCCGTGATCCA
```

a. Prove that the genome assembly problem is in NP.

b. Prove that the genome assembly problem is NP-Complete.

c. Speculate on how the human genome was sequenced even though it involves solving an NP-Hard problem for a large input size. The human genome is about 3 Billion base pairs. Readers at the time were able to read about 700 bases per read fragment, and sequencing the genome involved aligning about 30 million fragments.