Introduction to Algorithms	Final Exam Suggested Problems
CS 4820, Spring 2011	May 16, 2011

Thoughts

The problems suggested are enclosed to give you some practice with good exam-level questions. Exam-level questions are questions that you should need to do some thinking about, but not a couple of hours. The first things that comes to mind should be pretty close to the intended solution.

Besides these suggested problems, the first 10 or so problems of each chapter are generally good practice problems. Two numbers concatenated with an & are related to each other. Do both, one or neither depending on how clear what is going on is to you.

Greedy

K&T CH 4: 2,3,6

Divide and Conquer

K&T CH5: 1,2,4

Dynamic Programming

K&T CH6: 2,4,8

Network Flow

K&T CH7: 4&5,9,15*

*For 15 (a) you may want to write out a full solution (reduction, runtime, proof) explaining any relevant details from the bipartite matching problem.

NP-Completeness

K&T CH8: 2&3,6,9

Decidability

Prove whether each of these languages is decidable, undecidable but recursively enumerable, or not even recursively enumerable. It may be good to go through an iteration of trying to think what each should be without proof, checking your intuition with us, then fleshing out full proofs.

1 The language of Turing Machines which output "yes" on some input.
2 The language of pairs of Turing Machines that produce the same output on all inputs.
3 The language of Turing Machines, M, such that $L(M)$ contains at least 4820 elements.
4 The language of Turing Machines, M, such that $L(M)$ contains at most 4820 elements.
5 The language of Turing Machines, M, such that $L(M)$ is an infinite language.
6 The language of Turing Machines, M, such that $1011 \in L(M)$.
7 The language of Turing Machines M, such that $L(M) = \Sigma^*$.

Rice's Theorem is useful in solving most, but not all, of the foregoing problems. Note, however, then when applying Rice's Theorem to show that a language L is undecidable, the theorem does not reveal whether or not the language is recursively enumerable (r.e.). To do that, you generally need to do one of the following things:
(a) prove directly that L is r.e. by constructing a Turing machine that accepts L;
(b) prove directly that L is not r.e. by reducing the complement of the halting problem to L;
(c) prove that the complement of L is r.e. and conclude that L is not r.e., since an undecidable language whose complement is r.e. cannot be r.e..

Approximation

K&T CH7:11
K&T CH11:1,2,5