CS 332 - Homework 3 - draft

Due: Wednesday, March 22

Reading:
Chapter 4, section 4.2 pages 173-182 and

Problems:

1. Assume M is a TM whose program only allows the tape head to move right or stay stationery, but that it never moves left.
   Prove that the language of M is decidable. In particular, give an algorithm which shows that for any input w to M we can decide if M(w) loop or halts. From this conclude we can decide L(M) = the set of strings w that M accepts.
   (Hint: To get started think about how many steps the TM can make while staying on the same tape square without repeating the same configuration (and hence being in a loop)).

2. A set S is countable if S is finite or there is a 1-1 function from N, the set of natural numbers onto S.
   i. Prove that for any finite alphabet Σ, Σ∗ is countable.
   ii. Show that any subset of a countable set is countable.
   iii. As a corollary show that any decidable set is countable.
   Notes: By decidable here we mean decidable by a TM. You can can use the fact (proved in your lab section) that an decidable set can be enumerated in increasing order.

3. Prove that \( R_{TM} = \{ < M, w > | M \text{ is a TM and } M(w) \text{ rejects} \} \) is undecidable.

4. Prove that \( L = \{ < M, w > | M \text{ is a TM and at some point of the computation of } M(w) \text{ the state of } M \text{ is } q_4 \} \) is undecidable.

5. In class we proved that if C is recognizable then there is a decidable set D such that \( C = \{ x | \exists y \text{ where } (x,y) \in D \} \).
   Now prove the converse of this statement. That is, prove that if there is a decidable set D such that C = \{ x | \exists y \text{ where } (x,y) \in D \}, then C is recognizable.

6. A k-Independent Set (IS) in a graph G is a set S of k vertices in G with the property that no 2 vertices in S are connected by an edge.
   i. Give an example of a graph with a 4 IS but which doesn’t have a 5 IS.
   ii. Describe an algorithm which takes as input a graph G and which decides if G has a 4 IS or not.
   What is the order of the (time) complexity of your algorithm?