Homework 7

Discrete Structures
CS 173 [B] : Fall 2015

Released: Fri Apr 24
Due: Fri May 1, 5:00 PM

Submit on Moodle.

PART 1 (Machine-Graded Problems) on Moodle. [40 points]

PART 2 [60 points]

1. **Algorithm analysis** [20 points]

   Consider the following mystery function, which takes as input a list of real numbers and outputs a non-negative real number.

```
1: function Foo(A_1, A_2, ... A_n: real numbers)
2:     d := |A_1 − A_2|
3:     for i := 1 to n do
4:         for j := 1 to n do
5:             q := |A_i − A_j|
6:             if i ≠ j and q < d then
7:                 d := q
8:     return d
```

   (a) Give a brief English description of what the function foo computes.

   (b) How many times are lines 5 and 6 executed, as a function of n?

   (c) Express the running time of the algorithm expressed as $\Theta(f(n))$ for a simple function f? Briefly justify your answer.

   (d) This is a really bad algorithm for this task. Write pseudocode for a function with a better big-O running time.

   [Hint: First sort the input.]

   (e) Express the running time of your new algorithm from part d as a $\Theta(\cdot)$ expression. (You can use the fact that sorting can be done in $\Theta(n \log n)$ time. How much additional time do you take? What is the overall time?)

2. **Solving Search Problems using Decision Oracles** [20 points]

   In this problem, you need to devise a simple algorithm for efficiently solving a sudoku puzzle, but with the help of a "decision" oracle. In a sudoku puzzle you are given an $n \times n$ grid with some cells already filled in with numbers in the range 1 to n. Your goal is to fill in the rest of the cells with numbers in the range 1 to n, so that the grid satisfies some rule (for our purposes, the specific rule doesn't matter), or to discover that this is not possible (which is, in real-life puzzles, promised not to be the case).

You can query the decision oracle with a sudoku puzzle and it will only answer yes/no to indicate if there exists a solution for the sudoku puzzle or not.

Devise a strategy to solve any sudoku puzzle by making $O(n^3)$ queries to the decision oracle.

[Hint: Note that the oracle works no matter how many cells are already filled in. You can query it with partially filled grids to check if you are on the right track. Your algorithm will not need to rely on the exact rules of sudoku.]

Note: This is an instance of efficiently solving a search problem ("find a solution") by making use of a sub-routine that solves only the decision problem ("does there exist a solution"). When this is possible we say that the search problem can be reduced *to the decision problem. If the reduction can be carried out efficiently, the search problem "is not much more complex than" the decision problem, or alternately, the decision problem is "almost as complex as" the search problem.*

3. **A Finite State Machines** [20 points]

   (a) Consider a machine which has n bits of memory (initialized to all 0s). Its state is determined by the contents of its memory. This machine takes an input from the set $\Sigma = \{1, \ldots, n\}$: on input i, it toggles its i^{th} bit. (Toggling a bit changes its value from 0 to 1 or from 1 to 0.)

      i. Write down the transition function for this machine for $n = 2$, as table, with columns "current state," "input" and "next state." (Note that each state is labeled with 2 bits.)

      ii. What graph does the state-diagram for this machine (for a general value of n) resemble? You can treat pairs of directed edges between same two states, but pointing in opposite directions as a single undirected edge. Justify your answer.
      [Hint: For this part, it may be helpful to draw a diagram for the case $n = 2$ from the previous part. You don't need to submit your diagram.]

   (b) Construct a finite state acceptor that accepts exactly those strings which consist only of a's and b's such that the number of a's is even and the number of b's is odd. Represent this state machine by its transition function (in the form of a table). Indicate the start and final states.
   [Hint: How many states must this machine have?]