Homework 1 – Due Friday, August 29, 2014

Please refer to the general information handout for the full homework policy and options.

Reminders

- Your solutions are due before the lecture. Late homework will not be accepted.
- Collaboration is permitted, but you must write the solutions *by yourself without assistance*, and be ready to explain them orally to a member of the course staff if asked. You must also identify your collaborators. *Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.*
- To facilitate grading, please write down your solution to each problem on a separate sheet of paper. Make sure to include all identifying information and your collaborators on each sheet. Your solutions to different problems will be graded separately, possibly by different people, and returned to you independently of each other.
- For problems that require you to provide an algorithm, you must give *a precise description of the algorithm*, together with a *proof of correctness* and an *analysis of its running time*. You may use algorithms from class as subroutines. You may also use any facts that we proved in class.

Exercises These should not be handed in, but the material they cover may appear on exams.

- (**Basic proof techniques**)
  1. (**Induction**) Chapter 3, problem 5.
  2. (**Contradiction**) Chapter 3, problem 7.
- Point out the error in the following proof by induction.

  Claim 1 *In any set of h horses, all horses are the same color.*

  Proof: We proceed by induction on the number h of horses. (Base case) If $h = 1$, then there is only one horse in the set, and so all the horses in the set are clearly the same color.

  (Induction Step) For $k \geq 1$, we assume that the claim holds for $h = k$ and prove that it is true for $h = k+1$. Take any set H of $k+1$ horses. We show that all horses in this set are of the same color. Remove one horse from this set to obtain the set H_1 with just k horses. By the induction hypothesis, all the horses in H_1 are the same color. Now replace the removed horse and remove a different one to obtain a the set H_2. By the same argument, all the horses in H_2 are the same color. Therefore all the horses in H must be the same color, and the proof is complete.

Problems to be handed in

1. (**Review of Divide and Conquer**) Your niece has a number between 1 and n in her head, which she wants you to guess. After playing the game with her for a while, you notice the following pattern: when your current guess is closer to her number than your previous guess, she giggles; when your current guess is further away than previous guess (or the same distance away) she frowns. When you guess the right answer exactly, the jumps up and down and screams "you guessed it!".

   Give an algorithm that uses this extra information to find the secret number quickly. You can guess a number with the pseudocode command "guess(x)". After the first guess, this command will return either "closer", "further" or "correct." (On the first guess, the command returns only "correct" or "wrong".) Please give *both* an English description *and* pseudocode for your algorithm. Prove its correctness and analyze its time and space complexity.

   To get any credit at all, your algorithm must be correct. For full credit, your algorithm should use $\log_2(n) + O(1)$ guesses in the worst case (that is, an algorithm that makes $2\log_2 n$ guesses is not sufficient). You might find it helpful to recall how binary search works, but you'll need a slightly different pattern of queries to get an algorithm with the right number of queries .

   (Note: See the reminder above on problems that require an algorithm.)

2. (**Graph Algorithms**) An edge $e \in E$ in an undirected graph is *vulnerable* if removing e from the graph increases the number of connected components (that is, removing e breaks up one of the graph's connected components).

   Give as efficient an algorithm as you can for finding all the vulnerable edges of a graph $G = (V, E)$ given in adjacency list format. State the running time of your algorithm in terms of $m = |E|$ and $n = |V|$.

   [For full credit, your algorithm should run in time $O(m + n)$; a slower but correct algorithm will still receive significant credit. *Hint:* Use depth-first search (DFS) and look at the "non-tree" edges in the DFS tree, i.e. edges that point to nodes already explored. For each vertex, compute two quantities: its depth in the DFS tree, and the least depth reachable from one of its descendants by following non-tree edges.]

   (Note: See the reminder above on problems that require an algorithm.)