CS 577: Introduction to Algorithms (Section 1) — Homework 1

Out: 09/10/14 — **Due: 09/24/14**

Ground Rules

- This homework contains two sections. Problems in the first section will not be graded. These are for your practice only and do not need to be turned in. Problems in the second section are to be turned in and will be graded. We will provide solutions to all of the questions.

  This homework also contains an extra credit question. Extra credit questions will not directly contribute to your score. However, they will be taken into account in the final grading. Furthermore they are fun to solve and improve your understanding of the course.

- You will be graded on the correctness as well as clarity of your solutions. You are required to prove any claims that you make. In particular, when you are asked to design an algorithm, you must argue its correctness as well as running time. You may use without proof any theorems proven in class or in the textbook, as long as you state and cite them properly.

- You may work and submit solutions in pairs. Please refrain from consulting sources other than the text and lecture notes.

- Homework is due promptly at the start of class. Late homework will receive zero credit. Please submit each problem on a separate sheet of paper (or multiple sheets stapled properly) with your name written clearly on each sheet.

- Start working on your homework early. Plan your work in such a way that you have the opportunity to put some problems on the back burner for a while and revisit them later. Good luck!

Ungraded problems

1. Problem 5.2 in the textbook (Pg. 246).

2. Given a sorted array of distinct integers $A[1, \cdots, n]$, you want to find out whether there is an index i for which $A[i] = i$. Give a divide and conquer algorithm that runs in time $O(\log n)$.

3. Consider the following sorting algorithm. The initial call is SuperSort$(A, 0, \text{length}(A) - 1)$ where A is an array of integers.

```
void SuperSort(int A[],int i,int j){  \\sorts the subarray A[i..j]
    if (j == i+1)                     \\when there are only 2 elements
      if (A[i] > A[j]) swap(A,i,j)    \\swaps A[i] and A[j]
    else {
      int k = (j-i+1)/3;
      SuperSort(A,i,j-k);             \\sort first two thirds
      SuperSort(A,i+k,j);             \\sort second two thirds
      SuperSort(A,i,j-k);             \\sort first two thirds again
      }
}
```

   (a) Prove using induction that this algorithm is correct, that is, it always produces a sorted array.

   (b) Determine the asymptotic number of comparisons this algorithm makes.

4. Problem 5.6 in the textbook (Pg. 248).

Graded problems

5. **(5 points.)** Problem 5.3 in the textbook (Pg. 246–247).

   (Extra credit.) Give an $O(n)$ time algorithm for this problem. (Hint: In $O(n)$ time can you throw away at least half the cards such that if there is a "majority element" in the original collection, that element continues to form a majority in the remaining collection?)

6. **(5 points.)** You are given n nuts and n bolts of different sizes. Each nut fits exactly one bolt. Your job is to pair up the nuts with their matching bolts. You are allowed to compare nuts to bolts by trying to fit one into the other. The result of such a comparison is that the nut fits exactly with the bolt, or the nut is bigger, or the bolt is bigger. You cannot compare nuts to nuts or bolts to bolts. Give a randomized algorithm for solving this problem that runs in time $O(n \log n)$ in expectation.

7. **(5 points.)** Problem 5.7 in the textbook (Pg. 248–249).