CMPS 101: Winter 2016: HW 1

Due: 15th January 2016

• The assignment is to be attempted in groups of two.
• Each group needs to submit only one set of solutions.
• LATEX is preferred, but neatly handwritten solutions will also be accepted.
• The names of the group members, and their UCSC ID (@ucsc.edu email address) should prominently be written on the upper left corner of the first page.
• Multiple sheets should be stapled together in the upper right corner.
• Solutions to the problems should be clearly labeled with the problem number.
• Although no points are given for neatness, illegible and/or poorly organized solutions can be penalized at the graders option.
• Clearly acknowledge sources, and mention if you discussed the problems with other students or groups. In all cases, the course policy on collaboration applies, and you should refrain from getting direct answers from anybody or any source. If in doubt, please ask the instructors or TAs.

Question 1 (1 point): In the class we talked about different possible ways of speeding up insertion sort. One proposal was to use binary search.

• Write a pseudo-code for binary search which takes as input a sorted array $A$ of size $n$ and a query element $q$. Your procedure should return the position of $q$ in $A$, if $q$ occurs in $A$. Otherwise it should return $-1$.

• Given a sorted array of size $n$, and an arbitrary query $q$, what is the maximum number of comparisons that your binary search procedure performs? Prove your result formally.

Question 2 (2 point): Your friend claims to have invented a new sorting algorithm. The basic idea is as follows: Find the smallest number in $A[1 \ldots n]$ and exchange it with $A[1]$. Then find the smallest number in $A[2 \ldots n]$ and exchange it with $A[2]$ and so on. She needs your help to formalize this idea, and to prove its correctness.
• Write a pseudo-code for your friends algorithm.
• State the loop invariant that the algorithm maintains.
• Show using an induction argument that the algorithm indeed maintains the loop invariant.
• What is the time complexity of the algorithm in \( \Theta \) notation?

**Question 3 (2 points):** This problem will help you get some practice with induction proofs. In the class, we used induction to prove the loop invariant of insertion sort. Here, we want to use induction to prove the following statement:

The number of subsets of \( \{1, 2, ..., n\} \) having an odd number of elements is \( 2^{n-1} \).

**Question 4 (1 points)** Let \( f(n) = a_0 + a_1 n + a_2 n^2 + \ldots + a_k n^k \) be a degree-k polynomial where every \( a_i > 0 \). Show that \( f(n) \in \Theta(n^k) \).