Homework 1
Due 2017-Jan-16, 11:59pm

Prof. Amit Chakrabarti
Department of Computer Science
Dartmouth College

General Instructions. Each problem has a fairly short solution; if you find yourself writing more than a page of proof, you have probably not found the best solution; you might want to rethink your approach. The course website has a link to an "administrative details" page explaining the grading standards for all homework problems. **Each problem is worth 7 points.**

Honor Prinicple. Please read through the course's administrative details for a full explanation of the honor principle in this course. Some important points: you are allowed to discuss the problems and exchange solution *ideas*—but not full solutions—with your classmates. When you write up any solutions for submission, you must work alone. You may refer to any textbook you like, including online ones. However, you may not refer to published or online solutions to the specific problems on the homework. *If in doubt, ask the professor for clarification!*

---

1. We would like to sort an array of numbers using only in-place exchanges of the form `xch(i,j)` which swaps the array elements at indices i and j. Furthermore, we would like to avoid exchanging elements that are "far away": the two indices involved in an exchange must differ by no more than some constant C. For instance, "bubble sort" is an exchange-based sorting algorithm and it only swaps adjacent pairs, so it satisfies our requirement with $C = 1$.

   Prove that under these conditions sorting requires $\Omega(n^2)$ time in the worst case. Note that we did *not* insist that the algorithm use only comparisons.

2. Consider the problem of finding the second largest of n given elements (numbers, if you prefer) using comparisons. In class, we used a *leaf counting argument* to prove a lower bound of $n - 2 + \log n$ comparisons. This exercise will walk you through an alternative proof of this lower bound using an *adversarial argument*. The algorithm asks queries of the form "is $x_i < x_j$?" and the adversary answers them using an *adversarial strategy* which satisfies two properties:

   (a) There is always at least one input with which the adversary's answers are consistent.

   (b) If the algorithm has asked less than $n - 2 + \log n$ questions so far, then there are at least two consistent inputs with different answers (for the second largest element).

   Clearly the existence of such a strategy proves the lower bound.

   The adversary maintains n *tokens*, initially allocated one per element. When he is asked a comparison query between two elements, he declares the element that has more tokens the winner, and then he takes away all of the loser's tokens and gives them to the winner.

   2.1. Spell out this adversarial strategy in detail (make sure you handle all cases) and argue that it satisfies property (a) above.

   2.2. Recall that we argued in class that by the time the algorithm knows the second largest element, it must also know the largest. Prove that when the algorithm finds out the largest element, the adversary must have allocated all n tokens to that element.

   2.3. Prove that the largest element must therefore win at least $\log n$ comparisons.

   2.4. Finish the lower bound proof by showing that property (b) above is satisfied. Notice that you don't have to actually produce two different consistent inputs; just argue that having performed less than $n - 2 + \log n$ comparisons the algorithm cannot know the answer for sure.

3. Consider the problem of selecting both the minimum and the maximum of n given numbers, using comparisons. The naïve solution would be to first find the minimum and then the maximum, thereby using nearly $2n$ comparisons.

   3.1. Give an algorithm which solves this problem making at most $3\lceil n/2 \rceil$ comparisons.

   3.2. Prove the almost matching lower bound that any comparison-based algorithm for this problem *must* use at least $\lceil 3n/2 \rceil - 2$ comparisons.