CSOR W4231.002 – Spring, 2016

Homework 2

Out: Monday, February 8, 2016
Due: 8pm, Monday, February 22, 2016

Please keep your answers clear and concise. For all algorithms you suggest, you must prove correctness and give the best upper bound that you can for the running time. You should always describe your algorithm clearly in English. You are encouraged to write pseudocode for all your algorithms.

You should write up the solutions entirely on your own. Collaboration is limited to discussion of ideas only. You should list your collaborators on your write-up. If you do not type your solutions, be sure that your hand-writing is legible, your scan is high-quality (if applicable) and your name is clearly written on your homework.

1. (20 points) Let $x_1, \ldots, x_n$ be a list of n distinct numbers. We say that a_i and a_j are *inverted* if $i < j$ but $a_i > a_j$. The *Bubblesort* sorting algorithm swaps pairwise adjacent inverted numbers in the list until there are no more inversions (hence the list is sorted). Suppose that the input to Bubblesort is a uniformly at random selected permutation of the set $\{x_1, \ldots, x_n\}$. Find the expected number of inversions that need to be corrected by Bubblesort.

2. (20 points) Suppose that a sequence of items passes by one at a time. We want to maintain a sample of one item with the property that it is uniformly distributed over all the items that we have seen so far. Moreover we do not know the total number of items in advance and we cannot store more than one item at any time.

   (a) Consider the following algorithm. When the first item appears, we store it. When the k-th item appears, we replace the stored item with probability $1/k$. Show that this algorithm solves the problem.

   (b) Now suppose that when the k-th item appears, we replace the stored item with probability $1/2$. What is the distribution of the stored item in this case?

3. (30 points) An array A with n entries is said to have a *majority* element if more than half of its entries are the same. Given A, we want to find such a majority element, if one exists. A question of the form *is* $A[i] = A[j]$*?* can be answered in constant time; however questions of the form *is* $A[i] > A[j]$*?* are **not** permitted (for example, the elements of the array could be images so there is no order among them).

   (a) (10 points) Show how to solve this problem in $O(n \log n)$ time.

   (b) (20 points) Now give an (exact) linear-time algorithm for this problem.

4. (20 points) Problem 7-2 in the textbook.

5. (20 points) Problem 7-4 in the textbook.
   (If necessary, read pages 232-233 to refresh your memory on the definition of a stack.)

6. (20 points) The *square* of an $n \times n$ matrix A is its product with itself, AA.

   (a) Show that five integer multiplications suffice to compute the square of a 2×2 matrix.

   (b) What is wrong with the following algorithm for computing the square of an $n \times n$ matrix? "Use a divide & conquer approach as in Strassen's algorithm, except that instead of getting 7 subproblems of size $n/2$, we now get 5 subproblems of size $n/2$, thanks to part (a). Using the same analysis as in Strassen's algorithm, we can conclude that the algorithm runs in time $O(n^{\log_2 5})$".

   (c) In fact, squaring matrices is no easier than matrix multiplication. Show that if $n \times n$ matrices can be squared in time $O(n^c)$, then any two $n \times n$ matrices can be multiplied in time $O(n^c)$.