E0234: Assignment 3

Due: Monday, 1st Feb 2016.

It is highly recommended you do not google for the answers to the questions below. You can discuss with your friends, but then mention that in your submission. The writing should solely be your own.

0. **(Not to be submitted but recommended.)** Implement the randomized median finding algorithm where the sorting algorithm is also implemented by you. Your sorting algorithm should also keep account of number of array comparisons it makes. Run the algorithm on an array of a million entries where each entry is a random real between 0 and 100. Compare the time **and** number of comparisons that the randomized median finding algorithm makes with the those of (a) the naive algorithm which just sorts (using your sorting implementation) and returns the middle entry, and (b) The Find algorithm done in class. Do you see a big difference?

1. (**From high probability to expectation.**) In class, we proved that the randomized median finding algorithm found the median with $2n + o(n)$ comparisons with probability $1 - o(1)$. Prove that the expected running time of the algorithm is also $2n + o(n)$.

2. Prove that the expected running time of the Find subroutine done in class (see MR, Chap 1.4) is $O(n)$.

3. (**Non-uniform Birthday Paradox.**) Consider m balls being thrown randomly into n bins, however, the distribution is not uniformly at random. In particular, each ball lands in bin 1 with probability p_1, bin 2 with probability p_2, and so on, where $\mathbf{p} := (p_1, \ldots, p_n)$ is a probability vector with $\sum_{i=1}^{n} p_i = 1$ and $p_i \geq 0$ for all i. What is the smallest m for which you can say that there would be at least one bin with at least 2 balls with probability $> 1/2$? **Hint:** Observe that if all $p_i = 1/n$, this is the birthday paradox question, and so $m \approx \sqrt{2n}$. Also note that if some p_i is close to 1, then the required $m = O(1)$.

4. (**Variance of QuickSort**.) Compute the variance of the number of comparisons made by the QuickSort algorithm. We are looking for the correct order of magnitude and not the exact constants. **Hint:** Recall the computation of the expectation from the 1st lecture. Apply first principles.