**Honors Algorithms — Fall 2015 — Problem Set 2**

**Due: Oct. 12**

1. **QuickSort recursion depth.** In class, we showed that the expected recursion depth of QuickSort on inputs of size $n$ is at most $\log_{3/2}(n^2) + O(1)$. Note that $\log_{3/2}(n^2) = (2/\ln(3/2))\ln(n)$, and the constant $2/\ln(3/2)$ is approximately 4.933.

In this exercise, you are to prove a better bound on the expected recursion depth, namely, $\alpha \ln(n) + O(1)$, where $\alpha \approx 4.311$ satisfies $\alpha = 2e^{1−1/\alpha}$.

Hint: mimic the proof given in class, but replacing powers of 2 by powers of $t$ for a parameter $t$, and then choose $t$ to minimize the result.

Note: this upper bound is actually optimal.

2. **QuickSelect recursion depth.** In class, we showed that the expected recursion depth of QuickSelect on inputs of size $n$ is at most $\log_{4/3}(\ln(n)) + O(1)$. Note that $\log_{4/3}(\ln(n)) = (1/\ln(4/3))\ln(n)$, and the constant $1/\ln(4/3)$ is approximately 3.476.

In this exercise, you are to prove a better bound on the expected recursion depth, namely, $2\ln(n) + O(1)$.

You should follow the following proof outline. Suppose QuickSelect is run with input $(L, k)$, so we are looking for the $k$th smallest element in $L$. For simplicity, assume that the elements of $L$ are distinct, and that if we order the elements of $L$ in ascending order, the $i$th smallest element is $x_i$, and $i$ is called the rank of $x_i$.

For $i = 1, \ldots, n$, let $P_i$ be the indicator variable for the event that $x_i$ is chosen as a pivot at some point in the computation.

(a) If $D$ is the depth of the recursion (i.e., number of levels in the recursion tree), argue that $D = \sum_{i=1}^{n} P_i$, and conclude that $E[D] = \sum_{i=1}^{n} E[P_i] = \sum_{i=1}^{n} \Pr[P_i = 1]$.

(b) Argue that for $i = 1, \ldots, n$, we have $\Pr[P_i = 1] = 1/(\lvert i − k \rvert + 1)$.

This is kind of intuitive if you think about it, but try to be as rigorous as possible. One approach is to compute the conditional probability that $x_i$ gets chosen as the pivot at level 0 given that the rank of the pivot chosen at level 0 is in the interval $\{\min(i, k), \ldots, \max(i, k)\}$, and then combine this with the law of total probability and a proof by (strong) induction on $n$.

(c) Combine parts (a) and (b) to conclude that $E[D]$ is at most $2\ln(n) + O(1)$.

(d) Now go back through your proof and indicate how you can eliminate the assumption that the elements in $L$ are distinct.

(e) Finally, use the result of this exercise to give a proof that the expected number of comparisons performed by QuickSort is at most $2n \ln(n) + O(n)$.

3. **Yet another analysis of QuickSort.** Here is another way to analyze the expected running time of QuickSort. As usual, assume that the input is a list $L$ of $n$ items (and may contain duplicates). Let $T_L$ be a random variable representing the number of comparisons made on an input list $L$, and define $T(n)$ the be the maximum value of $E[T_L]$ over all lists $L$ of size at most $n$.

(a) Let $R$ be a random variable representing the relative position (in sorted order) of the randomly chosen pivot. So $R$ is uniformly distributed over $\{1, \ldots, n\}$. Let $S'$ and $S''$ be random variables representing the number of comparisons made in solving the two subproblems obtained after the partition step. We have

$$T_L \leq n − 1 + S' + S''.$$ 

Use the law of total expectation to argue that

$$E[T_L] \leq n − 1 + \frac{1}{n} \sum_{i=1}^{n} \left( E[S' \mid R = i] + E[S'' \mid R = i] \right).$$
(b) Using part (a), argue that

$$E[T_L] \leq n - 1 + \frac{2}{n} \sum_{i=1}^{n-1} \tilde{T}(i).$$

(c) Prove by (strong) induction on \( n \) that

$$\tilde{T}(n) \leq 2n \ln(n).$$

Hints: You may want to estimate the sum \( \sum_{i=1}^{n-1} i \ln(i) \) by the integral \( \int_1^n x \ln(x) dx \). Go ahead and use an online resource, such as Wolfram Alpha, to help calculate the indefinite integral \( \int x \ln(x) dx \).

4. **Nuts and bolts.** You have a mixed pile of \( n \) nuts and \( n \) bolts and need to quickly find the corresponding pairs of nuts and bolts. Each nut matches exactly one bolt, and each bolt matches exactly one nut. By fitting a nut and bolt together, you can see which is bigger. But it is not possible to directly compare two nuts or two bolts. Design and analyze an probabilistic algorithm for this problem with an \( O(n \log n) \) expected running time. Hint: customize QuickSort to the problem. Side note: only a very complicated deterministic \( O(n \log n) \) algorithm is known for this problem.

5. **Strange and random recursion.** Consider the following recursive, probabilistic algorithm \( A \), which takes as input a finite set \( S \) of items.

Algorithm \( A(S) \):

- if \( |S| \leq 1 \)
  - return \((0, 0, 0)\)
- else
  - let \( R \) be a randomly chosen subset of \( S \)
  - \((v_1, v_2, v_3) \leftarrow A(R)\)
  - \((w_1, w_2, w_3) \leftarrow A(S \setminus R)\)
  - return \((\max\{v_1, w_1\} + 1, v_2 + w_2 + |S|, v_3 + w_3 + |S|^2)\)

Let \((X_1, X_2, X_3)\) denote the output of \( A \) on input \( S \), and let \( n := |S| \). Show that \( E[X_1] = O(\log n) \), \( E[X_2] = O(n \log n) \), and \( E[X_3] = O(n^2) \).

Note: by a “randomly chosen subset of \( S \)”, we mean one that is chosen uniformly at random from among all \( 2^n \) subsets of \( S \).

Note: observe that \( X_1 \) represents the depth of the recursion tree, \( X_2 \) the sum of the recursive problem sizes, and \( X_3 \) the sum of the squares of the recursive problem sizes.

Note: this type of recursion naturally arises in some algorithms for polynomial factorization.

6. **Duplicate detection.** You are given \( n \) items, and you want to determine if there are any duplicates among them. One can obviously solve this problem by sorting, which takes \( O(n \log n) \) comparisons. Prove an \( \Omega(n \log n) \) time lower bound for this problem in the comparison model.

You may wish to proceed as follows.

- For a permutation \( \pi \) on \( \{1, \ldots, n\} \), let \( v(\pi) \) denote the leaf in the decision tree reached on any input \((a_1, \ldots, a_n)\) satisfying \( a_{\pi(1)} < a_{\pi(2)} < \cdots < a_{\pi(n)} \).
- Show that \( v(\pi) \neq v(\pi') \) if \( \pi \neq \pi' \).
  
  Hint: for any permutation \( \pi \), the comparisons along the path from the root to \( v(\pi) \) naturally define a partial ordering on \( \{1, \ldots, n\} \), and any partial ordering can always be extended to (at least) one total ordering.

7. **Sorting several lists.** Let \( L_1, \ldots, L_k \) be nonempty lists of integers in the range 1 to \( n \), and let \( m := |L_1| + \cdots + |L_k| \). Show how to sort all of the \( L_i \)'s (individually) in time \( O(m + n) \). That is, the output should be \( L_1', \ldots, L_k' \), where \( L_i' \) is the sorted version of \( L_i \), for \( i = 1, \ldots, k \).