1 What Happened Last Night? (30 pts)

Suppose there exists a freshman dorm with 100 rooms belonging to 100 students. During a crazy party, all of the students get rather drunk and each one randomly selects a room in which to sleep that night. Note that multiple students might end up in the same room.

(a) [10 pts] What is the expected number of students that end up returning to their own room?

(b) [10 pts] What is the expected number of students that end up in a room with exactly one other student? 

(c) [10 pts] With \( n \) students and \( n \) rooms, show that in the limit as \( n \to \infty \), the expected fraction of empty rooms after the crazy night approaches \( 1/e \).

2 A Quicker Quicksort (50 pts)

CLRS 7.3 introduces a randomized version of quicksort that obtains good expected performance over all inputs. Let’s learn about a variation that’s even quicker than this sort, which we’ll call QuickerSort3!

QuickerSort3 works like generic QuickSort except it chooses the pivot as the median (middle element) of a set of 3 elements randomly selected from the subarray.

For this problem, let’s assume that the elements in the input array \( A[1..n] \) are distinct and that \( n \geq 3 \). We denote the sorted output array by \( A'[1..n] \). Using this approach to choose the pivot element \( x \), we define \( p_i = Pr\{x = A'[i]\} \).

(a) [5 pts] Give an exact formula for \( p_i \) as a function of \( n \) and \( i \) for \( i = 2, 3, ..., n-1 \).

(b) [5 pts] What is the probability that QuickSort chooses the median (i.e. \( A'[(n+1)/2] \)) as the pivot?

(c) [10 pts] What is the probability that QuickerSort3 chooses the median as the pivot? How does this compare to QuickSort?
(d) [5 pts] If we define a “good” split to mean choosing the pivot as \( x = A'[i] \), where \( n/3 \leq i \leq 2n/3 \), what is the probability that \textbf{QuickSort} produces a good split?

(e) [10 pts] What is the probability that \textbf{QuickerSort3} produces a good split? You may leave summations in your answer. How does this compare to \textbf{QuickSort}?

(f) [15 pts] Suppose we define another algorithm called \textbf{QuickerSort5} which works like \textbf{QuickerSort3} except it chooses the pivot as the median of a set of 5 elements randomly selected from the subarray. Answer parts (c) and (e) for this variation.

3 Problem 3: A Slower Selection? (40 pts)

CLRS 9.3 introduces a selection algorithm that determines the \( i \)th smallest element of an input array of \( n > 1 \) distinct elements in linear time. Since the generic algorithm divides the \( n \) elements into \( \lfloor n/5 \rfloor \) groups of 5 elements each (plus one extra group with the leftover elements, if there are any), we’ll call it \textbf{Select5}.

(a) [25 pts] Suppose we design an algorithm that divides the \( n \) elements into \( \lfloor n/3 \rfloor \) groups of 3 elements each (plus possibly one extra group with fewer than 3 elements), which we’ll call \textbf{Select3}.

(i) Following the logic from lecture, how many elements are guaranteed to be discarded on each recursive call?

(ii) What recurrence relation does this give rise to?

(iii) Use the substitution method to prove that the runtime of this algorithm is \( O(n \log n) \).

(iv) Provide an English explanation for why \textbf{Select3} is slower than \textbf{Select5}.

(b) [15 pts] Now suppose we design an algorithm that divides the \( n \) elements into \( \lfloor n/7 \rfloor \) groups of 7 elements each (plus possibly one extra group with fewer than 7 elements), which we’ll call \textbf{Select7}.

(i) Following the logic from lecture, how many elements are guaranteed to be discarded on each recursive call?

(ii) What recurrence relation does this give rise to?

(iii) Use the substitution method to prove that the runtime of this algorithm is \( O(n) \).

4 Feedback form (2 pts extra credit)

Don’t forget to fill out the feedback form, at http://goo.gl/forms/Jegg96TXntn2sXFe2

Thank you for helping to make CS 161 a better class!