CS 3343 Analysis of Algorithms – Fall 11

9/27/11

# 5. Homework

**Due:** Tuesday 10/4/10 before class

1. **Expected values of dice (8 points)**
   Consider the following game: You roll $k$ fair six-sided dice. For every 6 you roll you win \$6, for rolling any other number you lose \$1.

   (a) (2 points) First assume $k = 1$, so you only roll one six-sided die. Describe the sample space and the random variable for this experiment.

   (b) (1 point) Compute the expected value of the random variable for $k = 1$.

   (c) (2 points) Now assume $k = 2$, so you roll two six-sided dice. Describe the sample space and the random variable for this experiment.

   (d) (2 points) Use linearity of expectation to compute the expected value of the random variable for $k = 2$. *(Hint: Express your random variable as the sum of two random variables.)*

   (e) (1 point) Would you play this game?

2. **Almost-best case for quicksort (8 points)**
   Let “Deterministic Quicksort” be the non-randomized Quicksort which takes the first element as a pivot, using the partition routine that we covered in class on the quicksort slides.

   Consider another “almost-best case” for quicksort, in which the pivot always splits the array $\frac{1}{3} : \frac{2}{3}$, i.e., one third is on the left, and two thirds are on the right, for all recursive calls of Deterministic Quicksort.

   (a) (2 points) Give the runtime recurrence for this almost-best case.

   (b) (2 points) Use the recursion tree to argue why the runtime recurrence solves to $\Theta(n \log n)$. You do not need to do big-Oh induction.

   (c) (4 points) Give a sequence of 4 distinct numbers and a sequence of 13 distinct numbers that cause this almost-best case behavior. (Assume that for 4 numbers the array is split into 1 element on the left side, the pivot, and two elements on the right side. Similarly, for 13 numbers it is split with 4 elements on the left, the pivot, and 8 elements on the right side.)

3. **Randomized insertion sort (3 points)**
   Assume randomized insertion sort computes a random permutation of the input array, and then runs deterministic insertion sort on this permutation.

   (a) (1 point) What is the runtime of deterministic insertion sort on the input array $[n, n-1, n-2, \ldots, 3, 2, 1]$?

   (b) (2 points) What is the best-case runtime of randomized insertion sort on the input array $[n, n-1, n-2, \ldots, 3, 2, 1]$? Describe what causes this best-case behavior.

Flip over to back page $\Longrightarrow$

# Practice Problems

**(Not required for homework credit.)**

1. **Expected values of dice**
   Clearly describe the sample space and the random variables you use.
   (a) Compute the expected value of rolling a fair four-sided die.
   (b) Compute the expected value of the sum of two fair four-sided dies...
      i. ... using the definition of the expected value.
      ii. ... using linearity of expectation. *(Hint: Express your random variable as the sum of two random variables.)*

2. **Best case for quicksort**
   Let “Deterministic Quicksort” be the non-randomized Quicksort which takes the first element as a pivot, using the partition routine that we covered in class on the quicksort slides.

   In the best case the pivot always splits the array in half, for all recursive calls of Deterministic Quicksort. Give a sequence of 3 distinct numbers and a sequence of 7 distinct numbers that cause this best-case behavior.

3. **Computing a permutation**
   Give pseudo-code for computing a random permutation of an input array of $n$ numbers. What is its runtime?