CS 5633 Analysis of Algorithms – Spring 10

1/28/10

3. Homework

Due **2/4/10** before class

Always justify your answers.

1. **Rolling dice (8 points)**
    - **a) (2 points)**
      Compute the expected value of rolling a fair six-sided die.
    - **b) (2 points)**
      Compute the expected value of rolling a fair k-sided die, for any $k \geq 1$.
    - **c) (2 points)**
      Use linearity of expectation to compute the expected value of the sum of m fair 6-sided dice, for any $m \geq 1$. *(Hint: Use part a). You may want to show it for $m = 2$ first if a general value of m is too confusing.)*
    - **d) (2 points)**
      Use linearity of expectation to compute the expected value of the sum of m fair k-sided dice. *(Hint: Use part b)).*

    Clearly describe the sample space and the random variables you use. 4 points will be given for correct notation. (The point of this exercise is to learn the notation, not just to get the intuition right.)

2. SimpleRoulette **(4 points)**

    The game SimpleRoulette is played as follows: The roulette wheel has a slot for each number from 0 to 36. You can bet on any number between 1 and 36, but not on the number 0. A bet costs you \$10. If the ball drops on the slot with your number, you get paid \$360, otherwise you don't get paid anything.

    Assuming that the wheel is fair (i.e., all numbers are equally likely), what is your expected win/loss in this game?

    Clearly describe the sample space and the random variables you use. 2 points will be given for correct notation.

3. **Best case example (2 points)**

    Consider (deterministic) quicksort which takes the first array-element as the pivot. Design an example input consisting of the 63 numbers $1, \ldots, 63$ which causes quicksort to always split $\frac{1}{2} : \frac{1}{2}$ in each recursive call. (You should be able to design it such that you don't need to worry about rounding fractions, but if you need to please do.)

Flip over to back page $\Longrightarrow$

4. **Randomized code snippets (8 points)**

Consider the following code snippets, where `RandomInteger(i)` takes $O(1)$ time and returns an integer between 1 and i, each with probability $1/i$.

```
 (I) for(i=2; i<=n; i++){
       if(RandomInteger(i)==1){
         for(j=1; j<=n; j++){
           for(k=1; k<=n; k++){
             print(‘‘hello’’);
           }
         }
       }
(II) for(i=2; i<=n; i++){
       if(RandomInteger(n)==1){
         for(j=1; j<=n; j++){
           for(k=1; k<=n; k++){
             print(‘‘hello’’);
           }
         }
       }
```

Answer the following questions for each of the code snippets above.

(a) (2 points) What is the best case runtime, in terms of n, of this code snippet? Describe what triggers a best-case scenario.

(b) (2 points) What is the worst case runtime, in terms of n, of this code snippet? Describe what triggers a worst-case scenario.

(c) (4 points) Now analyze the **expected** runtime. Clearly define your random variable. *Hint: Break your random variable into multiple random variables, one per outer loop iteration.*