11/24/15

Extra Credit Homework

Due **12/3/15** at the beginning of the lab

1. **Best case for quicksort (4 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 slide 12 of the randomized algorithms 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, a sequence of 7 distinct numbers, and a sequence of 15 distinct numbers that cause this best-case behavior.
   (Hint: For the sequence of 7 numbers the first two recursive calls should be on sequences of 3 numbers each.)

2. **Randomized code snippet (5 points)**
   Consider the following code snippet, where `RandomInteger(i)` takes $O(1)$ time and returns an integer between 1 and i, each with probability $1/i$.

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

   (a) (1 point) What is the best case runtime, in terms of n? Describe what triggers a best-case scenario.
   (b) (1 point) What is the worst case runtime, in terms of n? Describe what triggers a worst-case scenario.
   (c) (3 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.)*

3. **Sorting runtimes (6 points)**
   Consider the input array $[1^1, 2^2, 3^3, 4^4, \ldots, (n-1)^{n-1}, n^n]$ of n numbers. For this particular input array, what are the runtimes of the following algorithms? Justify your answers shortly.

   (a) Deterministic insertion sort.
   (b) Randomized insertion sort.
   (c) Deterministic quicksort where the pivot is always chosen as the first element.
   (d) Randomized quicksort.
   (e) Mergesort.
   (f) Counting sort.

Flip over to next page ⟹

4. **Randomized quicksort (3 points)**
   What is the runtime of randomized quicksort on an array of n equal numbers?

5. **To be or not to be ... in NP (6 points)**
   Which of the problems below are in NP, and which are not? Either justify why the problem is not in NP, or show that it is in NP by sketching an appropriate algorithm and its runtime.

   (a) Given a positive integer a, is a composite, i.e., the product of two integers greater than 1?

   (b) Given an undirected graph G with edge-weights. Compute a minimum spanning tree of G.

   (c) Given an array $A[1..n]$ of numbers, and a number K. Is there a subset of numbers in A that sum up to exactly K?