CS 5633 Analysis of Algorithms – Spring 09

1/29/09

3. Homework

Due **2/5/09** before class

Always justify your answers. All algorithms should be as efficient as possible and all runtimes should be as tight as possible.

1. **Master theorem (10 points)**
   Use the master theorem to find tight asymptotic bounds for the following recurrences. Justify your results. If the master theorem cannot be applied use the recursion tree method to estimate the answer. Assume that $T(1) = 1$.
   (a) **(2 points)**
   $T(n) = T(\frac{n}{2}) + \sqrt{n}$
   (b) **(2 points)**
   $T(n) = 27T(\frac{n}{3}) + n^3 \log^2 n$
   (c) **(2 points)**
   $T(n) = 2T(\frac{n}{2}) + \sqrt[3]{n}$
   (d) **(2 points)**
   $T(n) = 2T(\frac{2n}{5}) + n$
   (e) **(2 points)**
   $T(n) = T(n-1) + \log n$

2. **Multiplying polynomials (10 points)**
   A polynomial of degree n is a function

$$p(x) = \sum_{i=0}^{n} a_i x^i = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$$

   where a_i are constants and $a_n \neq 0$. For simplicity you may assume that n is a power of 2.

   (a) (1 point) What is the runtime of the straight-forward algorithm for multiplying two polynomials of degree n?
   (b) (5 points) We can rewrite the polynomial $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ as
   $x^{n/2}(a_n x^{n/2} + a_{n-1} x^{n/2-1} + \cdots + a_{n/2+1} + a_{n/2}) + (a_{n/2-1} x^{n/2-1} + \cdots + a_1 x + a_0)$.
   Use this as a starting point to design a divide-and-conquer algorithm for multiplying two degree-n polynomials (recurse on polynomials of degree $n/2$). Give a runtime analysis of your algorithm by setting up and solving a recurrence. The runtime of your algorithm should be the same as the runtime of part a).
   (c) (1 point) Show how to multiply two degree-1 polynomials $ax + b$ and $cx + d$ using only three multiplications. *Hint: One of the multiplications is* $(a + b) \cdot (c + d)$.

(d) (3 points) Design a divide-and-conquer algorithm for multiplying two polynomials of degree n in time $\Theta(n^{\log_2 3})$. *Hint: Reuse part* **b)** *and speed it up with the knowledge of part* **c)**

3. **Maximum in an array (3 points)**
   An array $A[1..n]$ contains n distinct numbers that are randomly ordered, with each permutation of the n numbers being equally likely. What is the expected value of the index of the maximum element in the array? Clearly describe the sample space and the random variable you use.

4. **Fives (4 points)**
   The game "Fives" is played as follows: In order to play the game the player has to pay \$1. Then the player rolls two fair six-sided dies. The bank pays \$5 for each rolled 5 (so the player can win either nothing, \$5, or \$10, but has to pay \$1 per game).

   What is the expected win/loss of "Fives"? Would you play it?

   Clearly describe the sample space and the random variables you use.

Related questions from previous PhD Exams

Just for your information. You **do not** need to solve them for homework credit.

P1 This problem is concerned with sorting. Suppose that an array A of n numbers comes from a uniform probability distribution over the interval $[0, 1)$. Let n_i be the number of values between i/n and $(i + 1)/n$ (greater or equal to i/n and less than $(i + 1)/n$).

(a) The idea of bucket sort is to divide the interval [0,1) into n equal-sized subintervals (buckets), distribute the n input numbers into the buckets, and sort each bucket using insertion sort. In pseudocode, write the bucket sorting algorithm to sort an array A. You do not need to write the pseudocode for insertion sort.

(b) What is $E(n_i)$, the expected value of n_i? (Hint: Use linearity of expectation.)

(c) What is $\mathrm{Var}(n_i)$, the variance of n_i? (Hint: n_i comes from a binomial distribution. The variance of a binomial distribution is $tp(1 - p)$ where there are t trials, and each trial has a probability of success p.)

(d) Using n_i as defined in a previous sub-question, what is $E(n_i^2)$, the expected value of n_i^2? (Hint: For a random variable X, it is the case that $E(X^2) = \mathrm{Var}(X) + (E[X])^2$.)

(e) Show that bucket sorting on array A sorts n numbers in $O(n)$ expected time. Be sure to address the issue of the quadratic complexity of insertion sort.