CS 5633 Analysis of Algorithms – Spring 06

3. Homework
Due **2/9/06** before class

1. **Master theorem (10 points)**
   Use the master theorem to find tight asymptotic bounds for the following recurrences. Justify your results. In cases to which the master theorem cannot be applied, solve the recurrence in a different way.

   Assume that $T(n)$ is constant for $n \leq 2$.
   - **(2 points)**
     $T(n) = 2T(\frac{n}{2}) + \sqrt{n}$
   - **(2 points)**
     $T(n) = 2T(n-2) + 2$
   - **(2 points)**
     $T(n) = 9T(\frac{n}{3}) + n^2 \log n$
   - **(2 points)**
     $T(n) = 4T(\frac{n}{3}) + n^2$
   - **(2 points)**
     $T(n) = 2T(\frac{3n}{4}) + n$

2. **Multiplying polynomials (10 points)**
   This problem is concerned with algorithms for multiplying two polynomials of degree n (so, of the form $a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$). For simplicity assume that n is a power of 2.

   (a) **a) (1 points)** What is the runtime of the straight-forward algorithm for multiplying two polynomials of degree n?

   (b) **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 $x^{n/2}(a_n x^{n/2} + a_{n-1} x^{n/2-1} + \cdots + a_{n/2}) + (a_{n/2-1} x^{n/2-1} + \cdots + a_1 x + a_0)$. Use this as a starting point to design a divide-and-conquer algorithm for multiplying two degree-n polynomials. The runtime of your algorithm should be the same as the runtime of part a). Your divide-and-conquer algorithm should recurse on inputs of size $n/2$ (i.e., on polynomials of degree $n/2$).

   (c) **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) **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. **I Love Ones (5 points)**
   The game "I Love Ones" is played as follows: The player rolls two fair six-sided dies. If he rolls two 1's the bank pays him \$5, if he rolls exactly one 1 the bank pays him \$1, and otherwise the player has to pay 50 cents.

   What is the expected win/loss of "I love ones"? Would you play it?

   Clearly describe the sample space and the random variables you use. 3 points will be given for correct notation.

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 divide-and-conquer algorithms and recurrence relations. Note that in the following we will write $T(n/2)$ to denote $T(\lfloor n/2 \rfloor)$ or $T(\lceil n/2 \rceil)$, which simplifies arithmetic manipulation and does not change asymptotic bounds.

    (a) Explain the divide-and-conquer paradigm for algorithm design, including a generic recurrence relation for the runtime $T(n)$ for inputs of size n. You may introduce additional constants or functions for your description.

    (b) Consider the recurrence relation $T(n) = 2T(n/2) + n$, and assume $T(2) = 2$. Prove by induction that $T(n) = n \log_2 n$ when $n \geq 2$ is a power of 2.

    (c) Provide and briefly justify asymptotically tight bounds for the following recurrence relations:

$$T(n) = T(n/2) + 1$$
$$T(n) = 3T(n/3) + n$$
$$T(n) = 3T(n/2) + n$$
$$T(n) = T(n-1) + n$$

P2 This problem is concerned with binary heaps. Below are some key subroutines. The heap property is that $A[\text{PARENT}(i)] \geq A[i]$ whenever $1 \leq i$ and $\text{PARENT}(i) \leq \textit{heap-size}[A]$.

PARENT(i)
    return $\lfloor i/2 \rfloor$

LEFT(i)
    return $2i$

RIGHT(i)
    return $2i + 1$

    (a) Using the subroutines above, write pseudocode for an algorithm that ensures that the heap property is true for an array A with elements from indices 1 to $length[A]$.

    (b) Justify the correctness of your algorithm.

    (c) Provide asymptotically tight bounds on the running time of your algorithm.

    (d) Justify your bounds on the running time of your algorithm.