15-451 Algorithms, Fall 2011

Homework # 1 **Due: September 13, 2011**

Please hand in each problem on a separate sheet and put your **name** and **recitation** (time or letter) at the top of each page. You will be handing each problem into a separate box in lecture, and we will then give homeworks back in recitation.

Remember: written homeworks are to be done *individually*. Group work is only for the oral-presentation assignments.

Problems:

(25 pts) 1. **Recurrences.** Solve the following recurrences, giving your answer in Θ notation. For each of them, assume the base case $T(x) = 1$ for $x \leq 10$. Show your work.

(a) $T(n) = T(n-2) + n^4$.

(b) $T(n) = 3T(n/4) + n$.

(c) $T(n) = 5T(n-2)$.

(d) $T(n) = \sqrt{n}\, T(\sqrt{n}) + n$. (E.g., we might get this from a divide-and-conquer procedure that uses linear time to break the problem into $\sqrt{n}$ pieces of size $\sqrt{n}$ each. Hint: write out the recursion tree.)

(25 pts) 2. **Recurrences and proofs by induction.** Consider the following recurrence:

$$T(n) \;=\; 2T(n/2) + n \lg n.$$

Let's use a base-case of $T(2) = 2$ and let's assume n is a power of 2. We would like you to solve this recurrence using the "guess and prove by induction" method.

(a) Try to prove by induction that $T(n) \leq cn \lg n$. In other words, assume inductively that $T(n') \leq cn' \lg n'$ for all $n' < n$ and try to show it holds for n. This guess is *incorrect* and so your proof should *fail.* (If your proof succeeds, then there is a problem!!) Point out where this proof fails.

(b) Use the way the above proof failed to suggest a better guess $g(n)$. Explain why you chose this guess and prove by induction that $T(n) \leq g(n)$ as desired.

(c) Now give a proof by induction to show that $T(n) \geq c' g(n)$ where $c' > 0$ is some constant and $g(n)$ is your guess from (b). Combining this with (b), this implies that $T(n) = \Theta(g(n))$.

(10 pts) 3. **Probability and expectation.** An *inversion* in an array $A = [a_1, a_2, \ldots, a_n]$ is a pair (a_i, a_j) such that $i < j$ but $a_i > a_j$. For example, in the array $[4, 2, 5, 3]$ there are three inversions. A sorted array has no inversions, and more generally, the number of inversions is a measure of how "well-sorted" an array is.

(a) What is the *expected* number of inversions in a random array of n elements? By "random array" we mean a random permutation of n distinct elements $a_1, \ldots, a_n$. Show your work. Hint: use linearity of expectation.

(b) It turns out that the number of comparisons made by the Insertion-Sort sorting algorithm is between I and $n + I - 1$, where I is the number of inversions in the array. Given this fact, what does your answer to part (a) say about the average-case running time of Insertion Sort (in Θ notation)?

(40 pts) 4. **Sorting and matrix games.** [Note: parts (c) and (d) are the harder ones]

Al Gorithm wants to optimize some code for sorting arrays of size 3, since it is in the inner loop of a climate modeling system. Using quicksort, given an array $A = [a, b, c]$, the code will choose one element as the pivot and compare it to the other two. If it was lucky and the pivot was between the other two (e.g., it chose a and we had $b < a < c$) then it is done. Otherwise, it needs to make one more comparison (between the non-pivots) to finish sorting.

In order to precisely describe what is going on, Al writes down a matrix with the rows corresponding to different pivot choices, columns corresponding to different possible locations for the middle (median) element, and filling in the matrix with the total number of comparisons used by that choice of pivot in that situation. Specifically, the matrix is as follows:

$$\begin{array}{ll|ccc} & & \multicolumn{3}{l}{\text{Median element is}} \\ & & a & b & c \\ \hline & a & 2 & 3 & 3 \\ \text{Pivot choice} & b & 3 & 2 & 3 \\ & c & 3 & 3 & 2 \\ \hline \end{array} \qquad \leftarrow \text{ total cost for sorting (\# comparisons)}$$

(a) Suppose that based on how these arrays are generated, Al believes there is a 25% chance element a will be the median, a 30% chance element b will be the median, and a 45% chance element c will be the median. In that case, what is the best pivot to use and what is the expected cost for sorting using that pivot?

(b) Suppose that Al suspects that these arrays are being generated by his nemesis Doc Chiney. Because of this, Al decides to randomize.

i. With what probabilities should Al choose pivots a, b, and c so that no matter what ordering Doc selects, Al's expected sorting cost is as small as possible? (I.e., Al wants

$$\max_{\text{Doc's choices}} \mathbf{E}_{\text{Al's choices}}[\text{sorting cost}]$$

to be as small as possible).

ii. What is the expected cost under this distribution?

(c) Suppose that Al's processor has special features that result in the following improved matrix (the only change is the upper-left-corner):

$$\begin{array}{ll|ccc} & & \multicolumn{3}{l}{\text{Median element is}} \\ & & a & b & c \\ \hline \text{Pivot choice} & a & 1 & 3 & 3 \\ & b & 3 & 2 & 3 \\ & c & 3 & 3 & 2 \\ \hline \end{array} \qquad \leftarrow \text{ total cost with Al's special processor}$$

i. *Now* what probabilities should Al use for pivots a, b, and c so that the worst-case expected cost ($\max_{\text{Doc's choices}} \mathbf{E}_{\text{Al's choices}}[\text{sorting cost}]$) is as small as possible?
Hint: by symmetry, you can assume Al's probabilities on b and c are equal. So, for some value p, Al has probability p on b, p on c, and $1 - 2p$ on a.

ii. And what *is* Al's expected cost when using those probabilities?

iii. Was there anything unexpected about the probabilities you came up with in part (c)i? [There is no right or wrong answer here - but was your intuition that a should have higher or lower probability than b and c?]

(d) Considering the matrix in part (c) above, suppose now *Doc* is going to probabilistically decide which element to make the median.

i. What probabilities should Doc use so that no matter which pivot Al picks, Al's expected cost is as *high* as possible? (Formally,

$$\min_{\text{Al's choices}} \mathbf{E}_{\text{Doc's choices}}[\text{sorting cost}]$$

is as high as possible?)

ii. And what is Al's expected cost in this case?

The above is an example of something called a "matrix game" or a "2-player zero-sum game" between players Al and Doc. If you did this correctly, the values you computed in parts c(ii) and d(ii) should be identical. The fact that they are identical is a case of von Neumann's minimax theorem (which we will discuss later in the course), and this value is called the *value* of the game. In class, when we talk about giving "upper bounds" and "lower bounds" for a problem like sorting, we are really talking about upper and lower bounds on the value of the associated game. An upper bound of $O(n \log n)$ means we have an algorithm that guarantees expected cost $O(n \log n)$ no matter what input it is given. A lower bound of $\Omega(n \log n)$ means that no algorithm can do better, which we typically will show by giving an adversarial probability distribution on inputs (an "algorithm for Doc") that is bad for all algorithms.

Also note that while in the case of $n = 3$ we have been examining, we are talking very small differences in performance between the various options, as we saw in class the gap becomes quite substantial for large values of n. E.g., using the first element as pivot has a worst-case of $\Omega(n^2)$, whereas randomizing over pivots guarantees any input has expected cost $O(n \log n)$.