# 15-451 Algorithms, Fall 2009

**Homework # 1** **Due: September 8, 2009**

---

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 $\Theta$ notation. For each of them, assume the base case $T(x) = 1$ for $x \leq 5$. Show your work.

(a) $T(n) = 4T(n/5) + n$.

(b) $T(n) = T(n-3) + n^3$.

(c) $T(n) = 3T(n-5)$.

(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 $T(n) = \Theta(g(n))$.

(25 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 $\Theta$ notation)?

(25 pts) 4. **Matrix games.** In the game of evens-odds, two players $E$ (Eve) and $O$ (Odelia) simultaneously show one or two fingers. Eve wins if the sum is even and Odelia wins if the sum is odd. For example, they might do this to decide who has to do some chore. Let's view winning as getting a score of $+1$ and losing as getting a score of $-1$.

For a game of this sort, a *strategy* is a deterministic or randomized method for picking what to play. The *value* of a strategy is the score you get (or the expected score if the strategy is randomized) against an opponent who plays optimally knowing your strategy (she has spies). For instance, considering the case of Odelia, the deterministic strategy "play one" has value $-1$ since Eve knowing this is Odelia's strategy would just play one. Similarly, the deterministic strategy "play two" has value $-1$ since Eve would just play two. The strategy "flip a coin and with probability $1/2$ play one and with probability $1/2$ play two" has value 0 since whatever Eve chooses, the expected score will be 0.

Here is a variation on evens-odds: suppose that we change the game so that both players playing two is a draw (both get a score of 0). In other words, we can summarize the game from Odelia's point of view by the following "payoff matrix":

Odelia plays / Eve plays

| | one | two |
|---|---|---|
| one | $-1$ | 1 |
| two | 1 | 0 |

(a) What is the value to Odelia of the strategy "with probability $1/2$ play one and with probability $1/2$ play two"? Remember, to solve this, figure out what Eve would do knowing that this is her strategy.

(b) What is the value to Odelia of the strategy "always play two"?

(c) What is the strategy for Odelia that has the highest value, and what is its value?

(d) What strategy for Eve has the highest value to Eve, and what is its value? Remember, we are assuming now that *Odelia* has spies and will play *her* best response to this strategy. Also, a win for Odelia is a loss for Eve and vice-versa.

(e) Explain in a few sentences why your answer to part ($c$) proves that your answer in ($d$) was optimal (Eve can't hope to guarantee any better), and vice-versa.

**Comment:** Matrix games often provide a helpful perspective on algorithm design. Think of a matrix where each row represents some algorithm, each column represents some possible input, and entry $ij$ is the cost of running algorithm $i$ on input $j$. (So, matrix entries are like payoffs to the adversarial column-player.) A good randomized algorithm, like randomized-quicksort, can be thought of as a randomized strategy for the row-player that has a low expected cost no matter what input the adversary selects.