15-451 Algorithms, Fall 2000

Homework # 2 **due: Wed, Thurs Sept 20-21, 2000**

Groundrules:

- This is an oral presentation assignment. As stated in the course information handout, you should work in groups of three. At some point before **Sunday Sept 17 at midnight** you should sign up for a 1-hour time slot on the signup sheet on the course web page.
- You are not required to hand anything in at your presentation, but you may if you choose. If you do hand something in, it will be taken into consideration (in a non-negative way) in the grading.

Problems:

1. [median finding] Let A and B be two sorted arrays of n elements each. We can easily find the kth smallest element in A in $O(1)$ time by just outputting $A[k]$. Similarly, we can easily find the kth smallest element in B. Give an $O(\log k)$ time algorithm to find the kth smallest element overall — i.e., the kth smallest in the *union* of A and B. Give a proof that your algorithm works as advertised. For purposes of defining what we mean here, you may assume there are no duplicate elements.

2. [lower bounds for deterministic quicksort] We saw in class that if we always choose the leftmost element as the pivot for quicksort then it will take time $\Omega(n^2)$ if the input is already sorted.[1]

   Your goal in this problem is to show that for *any* deterministic pivot-selection procedure, so long as the pivot is chosen by examining only a constant number of elements, you can reverse-engineer an input that makes quicksort take $\Omega(n^2)$ time.

   (a) Show how to construct an array of n elements (for any given value of n) such that the strategy "pick the middle element $A[n/2]$ as pivot" will have $\Omega(n^2)$ running time.

   (b) Now show how we can do the same thing for any pivot-selection strategy S that examines only a constant number of elements. Specifically, given the length n of the array, S chooses an index i_1 to examine. Based on what it sees ($A[i_1]$), it then chooses a new index i_2 to examine. Based on what it sees there, it then chooses a new index i_3 and so on up to i_c where c is some constant (like 5). S then chooses one of $A[i_1], A[i_2], \ldots, A[i_c]$ as the pivot. Show how given access to S you can reverse-engineer an n-element array such that quicksort using S takes time $\Omega(n^2)$.

[1] Technical detail: assume that the partitioning (rearranging) procedure doesn't change the relative order of elements that go into the same bucket. E.g., if the original array was $[3, 5, 0, 4, 2, 6, 1]$, and we use the leftmost element as pivot, the we will get $[0, 2, 1, 3, 5, 4, 6]$.

3. [Matrix games] In the game of rock-paper-scissors, two players simultaneously choose rock, paper, or scissors, and the winner is determined according to the rule "rock beats scissors, scissors beats paper, and paper beats rock". If both players choose the same object, then it is a draw. Let's say the loser pays the winner \$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 amount of money you win (or the expected amount if the strategy is randomized) against an opponent who plays optimally knowing your strategy (he has spies). For instance, the deterministic strategy "always play rock" has value -1 since an optimal opponent could just always play paper. The strategy "flip a coin and with probability 1/2 play rock and with probability 1/2 play paper" has value $-1/2$ since an optimal opponent knowing this is your strategy could always play paper, giving you an expected gain of $\frac{1}{2} \cdot (-1) + \frac{1}{2} \cdot 0$. The strategy "pick each of rock, paper, or scissors with probability 1/3" has value 0.

   (a) Here is a new game. Player A (Alice) hides either a nickel or a quarter behind her back. Then, player B (Bob) guesses which it is. If Bob guesses correctly, he wins the coin. If Bob guesses incorrectly, he has to pay Alice the value of the coin that Alice was hiding. In other words, the amount that Alice wins can be summarized by the following "payoff matrix":

		Bob guesses N	Bob guesses Q
Alice hides	N	-5	$+5$
	Q	$+25$	-25

   i. What is the value to Alice of the strategy "with probability 1/2 hide a nickel and with probability 1/2 hide a quarter"?

   ii. What strategy for Alice has the highest value, and what is its value?

   iii. What strategy for Bob has the highest value to Bob (equivalently, the lowest value to Alice), and what is its value?

   Hint: solving part (*iii*) will help prove that your answer to (*ii*) is correct, and vice-versa.

   (b) Suppose we change the game to say that when Bob guesses incorrectly, he has to pay Alice a quarter, no matter which coin Alice was hiding. What are the new optimal strategies for each player?

   (c) Make an analogy to problem 2 on this assignment. Can you think of sorting as a matrix game between two players? What would the rows and columns correspond to?