15-451 Algorithms, Fall 2011

Homework # 7 **due: Thurs December 8, 2011**

Please hand in each problem on a separate sheet and put your **name** and **recitation** (time or letter) at the top of each sheet. You will be handing each problem into a separate box, 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. [NP-completeness and approximation algorithms]

Let $\mathcal{A}$ be the set of pairs (G, k) such that G is a graph with a vertex cover of size k or less. Let $\mathcal{C}$ be the set of pairs (G, k) such that G has a vertex cover of size $k/2$ or less. Notice that $\mathcal{A} \supseteq \mathcal{C}$ because if $(G, k) \in \mathcal{C}$ then clearly $(G, k) \in \mathcal{A}$ also.

Determining whether a given input (G, k) belongs to $\mathcal{A}$ is NP-Complete (this is the Vertex-Cover problem), and also determining whether a given input (G, k) belongs to $\mathcal{C}$ is NP-complete (since this is really the same problem).

Describe a set $\mathcal{B}$ such that $\mathcal{A} \supseteq \mathcal{B} \supseteq \mathcal{C}$ but membership in $\mathcal{B}$ can be decided in polynomial time (and explain why membership in $\mathcal{B}$ can be decided in polynomial time). So this is just like the situation on Mini 5. Hint: think approximation algorithms.

(25 pts) 2. Randomized Rent-or-Buy

Consider the rent-or-buy problem in the simple case that the cost to buy skis is twice the rental cost. The optimal deterministic algorithm (as discussed in class) is: "rent once, then buy" for a competitive ratio of 3/2. In this problem we will consider *randomized* algorithms. For a randomized algorithm, the definition of competitive ratio is $\max_\sigma \mathbf{E}[\text{alg cost}(\sigma)]/OPT(\sigma)$, where "$\sigma$" is the number of times we go skiing.

We can think of competitive analysis as a 2-player game: we pick a strategy, the adversary picks how many times σ we go skiing, and then we are charged the above ratio. It turns out that if the purchase cost is twice the rental cost, there are only two choices for the adversary that we need to worry about: $\sigma = 1$ or $\sigma = \infty$. Because of this, there are only two deterministic strategies that make sense for us to randomize over: either (1) buy right away, or (2) rent once and then buy if we go skiing again.

(a) Draw out a matrix corresponding to this game. The matrix should have two rows (one for "buy right away" and one for "rent once then buy") and two columns (one for "$\sigma = 1$" and once for "$\sigma = \infty$"). The cells in the matrix should give the competitive ratio of that algorithm in that event.

(b) The minimax optimal strategy for the row player in this matrix game gives the randomized algorithm with the best competitive ratio. What is that strategy, and what is the minimax value of the game (the optimal randomized competitive ratio)?

(c) To complete the argument, prove the two assertions made above that allowed us to reduce the problem to the 2-by-2 case. Specifically,

   i. Show that we only need to consider $\sigma = 1$ and $\sigma = \infty$ by arguing why for any randomized algorithm $\mathcal{A}$, and any $\sigma > 1$, we have $\mathbf{E}[\mathcal{A}(\sigma)]/OPT(\sigma) \leq \mathbf{E}[\mathcal{A}(\sigma+1)]/OPT(\sigma+1)$.

   ii. Assuming now that either $\sigma = 1$ or $\sigma = \infty$, argue why the optimal randomized algorithm would put zero probability on the deterministic strategy "rent i times then buy" for any $i > 1$.

(25 pts) 3. [Random-access[1] long division].

Give a polynomial time algorithm to find the Nth digit of the fraction A/B, where A, B and N are all given in binary.

Input: integers (A, B, N) in binary notation, where $A < B$.

Let $0.d_1d_2d_3\cdots$ be the decimal expansion of the fraction $\frac{A}{B}$.

Output: d_N.

Note: the key thing here is that your algorithm's running time should be polynomial in $\log N$ (and $\log A$ and $\log B$). The standard way of doing long division would instead be polynomial in N. In particular, the standard long division would look like this:

```
for $i = 1$ to $N$ do:
    $d_i = 10A$ div $B$;
    $A = 10A \bmod B$;
```

where "div" is integer division.

(25 pts) 4. FFT

Consider the following problem. You are given a string P of 1's and $*$'s (the "pattern"), and a string T of 0's and 1's (the "text"). Your job is to find all places where the pattern P appears in text T, where a star can match either a 0 or a 1. For instance, if $P =$ `11*1` and $T =$ `10111101`, then P appears twice in T: once starting at the 3rd position in T and once starting at the 5th position in T.

Say P has length n and T has length m, where $m > n$. There is a simple $O(mn)$-time algorithm to solve this problem: try all $O(m)$ possible starting positions, and for each one, check in time $O(n)$ to see if P matches there.

Show how you can use the FFT to solve this problem in time only $O(m \log m)$.

[Hint: you may wish to reverse P. Note: you do not need to explain the inner workings of the FFT algorithm.]

[1] "Random access" as in random-access memory, i.e., as opposed to sequential-access. Not "random" as in probability.