15-451 Algorithms, Fall 2000

Homework # 7 **due: Tuesday December 12, 2000**

Groundrules: You should try to do the assignment as much as possible by yourself. If you get stuck you may ask for help from friends or TAs or the instructors. In any event, you should do your writeup by yourself, in your own words. If you did get help from others, please list their names at the top of the first page of your assignment. Your assignment is due at the beginning of class on the due date.

Problems:

1. [Approximating Set-Cover] On the last assignment you proved that the *set-cover* problem is NP-complete. In class, we proved that the greedy algorithm achieved an approximation ratio of $O(\log n)$.

   (a) Suppose you are given a set cover instance in which no point is in more than t different sets. Show how to achieve a t-approximation in this case. I.e., give a polynomial-time algorithm that finds a set cover with at most t times as many sets as the optimal solution. Prove that your solution works.

   (b) What approximation ratio does your algorithm give for the *vertex cover* problem, when you translate it into set-cover?

   (c) Here is another approximation algorithm, based on solving the *fractional* version of the set-cover problem that you looked at on homework 6.

   Here's the algorithm:

   > First solve the fractional set-cover problem using linear programming. Let x_i be the value of the variable for set s_i, and let $k' = \sum_i x_i$. (We know that $k' \leq k$, where k is the number of sets in the optimal integral set cover.) Now normalize, by dividing all the variables by k'. Specifically, let $y_i = x_i/k'$. So, $\sum_i y_i = 1$. We now use y_i as a probability distribution as follows:
   >
   > - Randomly choose a set s_i according to the probability distribution defined by the y_i. Add this set to our current collection. Keep randomly picking sets this way until we have a set cover.

   This is the whole algorithm. Prove that after $2k'\ln(n)$ draws, all points are covered with probability at least $1 - 1/n$.

   (Hint: use the fact that $(1 - 1/d)^d \leq 1/e$ for all $d \geq 1$.)

2. Let $n = n_1 n_2 \cdots n_k$, where the n_i are all relatively prime. I.e., $\text{GCD}(n_i, n_j) = 1$ for all $i \neq j$.

   Give an efficient algorithm that takes as input a k-tuple $V = (r_1, r_2, \ldots, r_k)$ where $0 \leq r_i < n_i$ and computes a number m $(0 \leq m < n)$ such that $m \bmod n_i = r_i$ for all i.

3. Here's Bresenham's algorithm for drawing a line from $(0,0)$ to (a,b), where $0 \leq b \leq a$:

```
r = 0;
y = 0;
for (x=0; x<=a; x++) {
   plot(x,y);
   r = r+b;
   if (r > floor(a/2)) {
      r = r-a;
      y++;
   }
}
```

(a) Write a formula for the value of r at the beginning of the loop as a function of x and y.

(b) Prove that if $r = 0$ (at the beginning of the loop) for some value of x such that $0 < x < a$ then $\mathrm{GCD}(a,b) > 1$.

(c) Let E be the number of times this event happens (i.e. $r = 0$ at the beginning of the loop when $0 < x < a$). Express E as a function of a and b.