CS 466/666, Spring 2006
Timothy Chan

Assignment 3 (due June 29 Thursday noon)

Please read `http://www.student.cs.uwaterloo.ca/~cs466/policies.html` first for general instructions.

1. [*12 marks*] Two $n \times n$ matrices A and B are said to *commute* if $AB = BA$. (Matrix multiplication in general is not commutative, but occasionally commuting pairs do arise, e.g., try $A = I$ and $B =$ anything.)

   We can test whether A and B commute in $O(n^{2.81})$ time by just running Strassen's algorithm. In this exercise, we will discover a faster Monte Carlo algorithm:

   (a) [*6 marks*] Let p be a prime, let x be a random number between $\{0, \ldots, p-1\}$, and let

   $$y = \begin{pmatrix} 1 \\ x \\ x^2 \\ \vdots \\ x^{n-1} \end{pmatrix}.$$

   Given two $n \times n$ matrices C and D where $C \neq D$ and entries are integers in $\{0, \ldots, p-1\}$, upper-bound the probability that $Cy = Dy$ with entries taken modulo p.

   (b) [*6 marks*] Now, give a Monte-Carlo algorithm that decides whether A and B commute in only $O(n^2)$ time with error probability at most $1/n$. Your algorithm should use arithmetic on $O(\log M)$-bit numbers only, assuming that the entries of A and B are integers in $\{0, \ldots, M-1\}$ and $M \geq n$.

   [Hint: can ABy be computed in $O(n^2)$ time?]

2. [*18 marks*] Consider the following problem: given a binary string $u = a_1 a_2 \ldots a_n \in \{0,1\}^*$, decide whether u contains 11 as a substring (i.e., whether u contains two consecutive 1's). Obviously, there is an $O(n)$-time algorithm. We will investigate the precise number of bits that need to be examined (i.e., number of a_i's that need to be evaluated) for this problem.

   (a) [*9 marks*] Prove that every deterministic algorithm needs to evaluate at least $cn - o(n)$ bits in the worst case for some constant $c \geq 0.75$.

   [Note: A proof for a weaker constant, e.g., $c = 1/2$ or $c = 2/3$, will receive some partial marks. Hint: use an adversary argument. Fix $a_1 = a_5 = \cdots = 0$ and try to force all positions $a_2, a_3, a_4, a_6, a_7, a_8 \ldots$ to be evaluated.]

(b) [*9 marks*] Give a Las Vegas algorithm that only evaluates an expected $c'n+o(n)$ number of bits for any given input string for some constant $c' < 1$.

[Note: A correct proof for any constant strictly smaller than 1 will receive full marks. Hint: for one possible approach, consider the following observations: (i) if $a_i = a_{i+2} = 0$, then we do not need to evaluate a_{i+1} (why?); if $a_1a_2a_3a_4a_5$ does not contain 11, then $a_1 = a_3 = 0$ or $a_2 = a_4 = 0$ or $a_3 = a_5 = 0$ (why?).]

3. [*20 marks*] The original version of the smallest enclosing circle problem is very sensitive to so-called "outliers": if a single point is corrupted, the optimal circle could change drastically. This motivates the following extension of the problem:

> given a set S of n points in 2D and an integer k, find the smallest circle that encloses at least $n - k$ points (i.e., at most k points can be strictly outside the circle).

In this question, you will design an efficient Monte Carlo algorithm for this new problem. Assume that there are no degeneracies, i.e., no 4 points lie on a common circle.

(a) [*2 marks*] Describe a naive algorithm that runs in $O(n^4)$ time.

(b) [*4 marks*] Let C^* denote the optimal circle, let B^* denote the set of the (at most) 3 points on the boundary of C^*, and let Q^* denote the set of at most k points strictly outside the circle. Form a random subset $R \subseteq S$ as follows: for each point $p \in S$, put p in R with probability $1/k$ (Consequently, $E[|R|] = n/k$.)
Show that the probability that $B^* \subseteq R \subseteq S - Q^*$ is at least $\Omega(1/k^3)$.
[Hint: the inequality $1 - x \leq e^{-x}$ might come in handy...]

(c) [*7 marks*] Describe a Monte Carlo algorithm that runs in $O(n)$ worst-case time and is correct with probability $\Omega(1/k^3)$. (Yes, this probability converges to 0, i.e., your algorithm is supposed to wrong most of the time!)
[Hint: you may use the smallest enclosing circle algorithm from class as a subroutine, which runs in linear expected time; obviously part (a) is meant to help...]

(d) [*7 marks*] Now, describe a Monte Carlo algorithm that runs in $O(nk^3)$ worst-case time and is correct with probability greater than 0.99.