CS 466/666, Spring 2015
Timothy Chan

Assignment 3 (due July 8 Wednesday 5pm)

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

1. [*20 marks*] Consider the following problem: given a binary string $u = a_1 a_2 \ldots a_n \in \{0,1\}^*$, decide whether u contains 00 as a substring (i.e., whether u contains two consecutive 0'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) to solve this problem.

   (a) [*10 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 might still receive some partial marks. Hint: use an adversary argument. Fix $a_1 = a_5 = \cdots = 1$ and try to force all positions $a_2, a_3, a_4, a_6, a_7, a_8 \ldots$ to be evaluated.]

   (b) [*10 marks*] Give a Las Vegas algorithm that 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 $a_i = a_{i+2} = 1$, then we do not need to evaluate a_{i+1} (why?); (ii) if $a_1 a_2 a_3 a_4 a_5$ does not contain 00, then $a_1 = a_3 = 1$ or $a_2 = a_4 = 1$ or $a_3 = a_5 = 1$ (why?).]

2. [*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 to solve 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.

3. [*11 marks*] Consider a variant of the random walk scenario: An ant is placed on an interval $[0, n]$. At position i, the ant moves to position $i - 1$ with probability 1/2 and position $i + 2$ with probability 1/2. The exceptional cases are when $i = n$ and $i = n - 1$: here, the ant moves to position $i - 1$ with probability 1. At position $i = 0$, the ant stops.

   Prove that the expected number of steps before the ant stops starting at any position $i \neq 0$ is at least exponential: more precisely, $\Omega(\phi^n)$, where $\phi = (1 + \sqrt{5})/2$.

   [Hint: define t_i as in class and $d_i = t_i - t_{i-1}$. As you may have guessed, Fibonacci numbers come up somehow (you may use the fact that the k-th Fibonacci number has growth rate $F_k = \Theta(\phi^k)$). You do not need to compute t_i exactly; just lower-bound the value of t_i.]

4. [*14 marks*] Consider the following very simple algorithm for the minimum spanning tree problem, which uses the "inclusion rule": randomly pick a vertex u, find its nearest neighbor v, contract v to u, and repeat.

(a) [*4 marks*] What is the expected degree of a randomly chosen vertex from a graph with n vertices and m edges?

(b) [*6 marks*] Show that with an appropriate implementation, the above algorithm requires an expected running time that satisfies a recurrence of the following form

$$T(m, n) = T(m, n - 1) + O(\text{ the expression from part (a) }).$$

(c) [*4 marks*] Solve the recurrence to determine the expected running time. How does this algorithm compare with the classical MST algorithms (Borůvka, Prim, and Kruskal), in terms of performance and ease of implementation?