CS 473, Spring 2023
Homework 6 (due Mar 22 Wed 9pm)

Instructions: As in Homework 1.

Problem 6.1: 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., we allow at most k points to be outside the circle).

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

(a) (10 pts) Describe a naive algorithm that runs in $O(n^4)$ time.

(b) (35 pts) 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 outside the circle. Form a random subset $R \subseteq S$ as follows: for each point $p \in S$, put p in R independently with probability $1/k$. (Consequently, $E[|R|] = n/k$.)
Show that the probability that $B^* \subseteq R \subseteq S - Q^*$ is at least $\Omega(1/k^3)$.

(c) (40 pts) 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; obviously, (b) is meant to help...]

(d) (15 pts) Now, describe an efficient Monte Carlo algorithm that has error probability at most 0.01. Analyze the worst-case running time as a function of n and k. When k is small, it should be much faster than the algorithm in (a).

Problem 6.2: We are given an $n \times n$ matrix A where every row is in increasing order and every column is in increasing order (i.e., $A[i,j] < A[i,j+1]$ and $A[i,j] < A[i+1,j]$ for every i, j).

(a) (25 pts) Given a value t, describe a deterministic algorithm for listing all elements in A that are at most t. The output does *not* have to be in sorted order. The running time should be $O(n + K)$, where K denotes the output size, i.e., the number of elements at most t.
[Note: this is similar to Problem 0.2.]

(b) (75 pts) Given k, describe a randomized Las Vegas algorithm for finding the first k smallest elements in A. The output does *not* have to be in sorted order. The expected running time should be $O(n + k)$.
[Hint: use part (a) and random sampling.]

Problem 6.3: We are given a (unweighted) bipartite graph G with n vertices and m edges. Suppose we have already computed a maximum matching M in G. Now suppose we delete a vertex v from G and all its incident edges. Let G' be the new graph (with $n-1$ vertices). Describe how to efficiently compute a new maximum matching M' in G'. (Your algorithm should be faster than re-running a matching algorithm from scratch. Remember to prove correctness of your algorithm.)