

CS 473 (Spring 2025)
Homework 6 (due Mar 27 Thu 10am)

Instructions: As in previous homeworks.

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).

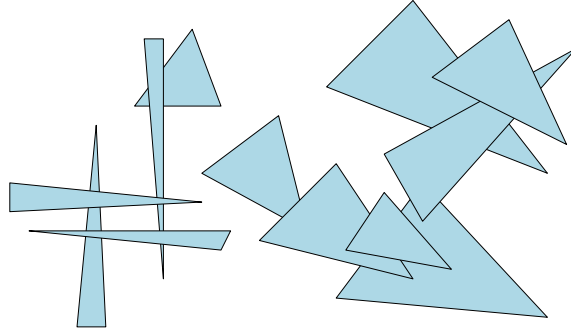
In this question, you will design an efficient Monte Carlo algorithm to solve this new problem. You may assume that there are no degeneracies, i.e., no 4 points lie on a common circle.

- (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, $\mathbb{E}[|R|] = n/k$.)
Prove 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 n triangles T_1, \dots, T_n in 2D. We want to compute the area of their union, i.e., $A = \text{area}(T_1 \cup \dots \cup T_n)$. (Note that the triangles may overlap, so A may not necessarily be equal to $\text{area}(T_1) + \dots + \text{area}(T_n)$.)

In this problem, we consider a randomized Monte Carlo algorithm to compute a good approximation of A . Let r be a parameter. Here is the pseudocode:

1. for $i = 1$ to n do
2. independently pick r random points $p_{i,1}, \dots, p_{i,r}$ from T_i
3. return $X = \sum_{i=1}^n |\{p_{i,1}, \dots, p_{i,r}\} \setminus (T_1 \cup \dots \cup T_{i-1})| \cdot \frac{\text{area}(T_i)}{r}$



- (a) (30 pts) Prove that $\mathbb{E}[X] = A$.
 [Hint: what is $\sum_{i=1}^n \text{area}(T_i \setminus (T_1 \cup \dots \cup T_{i-1}))$?]
- (b) (30 pts) Prove that $\text{Var}[X] \leq A^2/r$. ($\text{Var}[X]$ denotes the variance of X .)
- (c) (40 pts) By choosing r appropriately, show that the algorithm runs in $O(n^2)$ time and outputs a value X between $0.99A$ and $1.01A$ with probability at least 0.99 .
 [Note: you may assume that we can generate a random point in a triangle in $O(1)$ time, we can compute the area of a triangle in $O(1)$ time, and we can test whether a point is inside a triangle in $O(1)$ time. The above algorithm actually works for other types of objects besides triangles, and in dimension higher than 2.]

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.)