

CS 473 (Fall 2026)

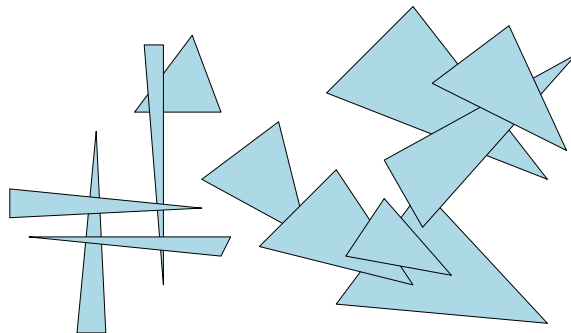
Homework 6 (due Oct 22 Thu 10am)

Instructions: Carefully read <http://courses.grainger.illinois.edu/cs473/fa2026/policies.html#hw> and <http://courses.grainger.illinois.edu/cs473/fa2026/integrity.html> (in particular, remember to include the 3 disclosure statements, on collaborators, outside sources, and AI use, for each problem).

Problem 6.1: Given a set S of n points in 2D and an integer k , we want to find the smallest-area rectangle that encloses at least $n - k$ points (i.e., we allow at most k points to be outside the rectangle). The sides of the rectangle must be axis-parallel (i.e., the rectangle cannot be rotated).

- (a) (5 pts) Describe an algorithm for the $k = 0$ special case that runs in $O(n)$ time.
- (b) (35 pts) Now consider the general k case. Let r^* denote the optimal rectangle, let B^* denote the set of the 4 points on the boundary of r^* , and let Q^* denote the set of at most k points outside r^* . Take 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 $\Omega(1/k^4)$.
- (c) (40 pts) Describe a Monte Carlo algorithm that runs in $O(n)$ worst-case time and is correct with probability $\Omega(1/k^4)$. (Yes, this probability converges to 0, i.e., your algorithm is supposed to be wrong most of the time!) [Hint: use (b)...]
- (d) (20 pts) Use (c) to obtain a Monte Carlo algorithm that is correct with probability at least 0.99. Analyze the worst-case running time as a function of n and k .

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.]

Optional Problem 6.3 (not graded): 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.)