CS 634 Homework 3

Qilin Ye

October 30, 2025

This homework is typed in a rush as I have been extremely busy lately. I apologize for typos and ambiguities in advance.

Solution to problem 1. (1) Give points p, q, set $k = w(q)/w(p)$. Then the bisector is $\{x \in \mathbb{R}^2 : \|x - p\| = k\|x - q\|\}$ which is an Apollonius circle, with degenerate case $k = 1$ being the perpendicular bisector line.

(2) We claim the complexity is $\mathcal{O}(n^2)$. We relate to the lower envelope perspective used in standard Voronoi diagrams. For each p, let

$$C_p = \{(x, y, z) : z = w(p)\|(x, y) - p\|\},$$

so that the MW Voronoi diagram is the vertical projection of the lower envelope $\min_p C_p$, and a point (x, y) belongs to cell p iff C_p attains the minimum $\min_p C_p$ at (x, y). The lower envelope of n constant-degree algebraic surfaces in $\mathbb{R}^3$ has a combinatorial complexity of $\mathcal{O}(n^2)$, and so is the planar projection.

Solution to problem 2.

- For convenience write $\Delta = \min_{i \neq j} d(P_i, P_j)$. Suppose Δ is attained by $p \in P_i, q \in P_j$. Let m be the midpoint of pq and let D be the open disk centered at m with radius $\|p - q\|/2$. We claim that D contains no point of P. Suppose, for contradiction, that there exists $r \in P \backslash \{p, q\}$ with $\|r - m\| \leqslant \Delta/2$.

  - If $r \in P_i$, by triangle inequality
  $$\|r - q\| \leqslant \|r - m\| + \|m - q\| \leqslant \Delta$$
  with strict inequality unless r is collinear with $\{p, m, q\}$ and $r = p$ (the latter since if $r \neq p$, then $\|r - q\| < \Delta$).
  - The case $r \in P_j$ is symmetric.
  - If $r \notin P_i \cup P_j$, then from the definition of Δ,
  $$\|r - p\| \geqslant \Delta, \qquad \|r - q\| \geqslant \Delta$$
  so by triangle inequality $\|r - m\| \geqslant \|r - p\| - \|m - p\| \geqslant \Delta - \Delta/2 = \Delta/2$, which leads to contradiction like in the first case: equality would force either $r = p$ or $r = q$.

  Therefore, D is empty. Since the circle ∂D passes through p, q and contains no other points in p in its interior, pq is an Delaunay edge.

- We first compute $DT(P)$ using $\mathcal{O}(n \log n)$ time, and sort the edges in $DT(P)$ by edge, another $\mathcal{O}(n \log n)$ time. Following the hint, we initialize a Union-Find structure with each point in its own set. Scan the sorted edge list from shortest to longest, and whenever an edge uv has its endpoints in different sets, union those two sets. Stop as soon as exactly k sets remain, and output them as the clusters.

We briefly discuss correctness. At any moment of the scan, let the current partition be $C_1, \cdots, C_\ell$. Let $\delta = \min_{a \neq b} d(C_a, C_b)$ and pick x, y witnessing δ. By (1), xy is a Delaunay edge, and by our algorithm, it is the shortest whose endpoints are yet in different sets; hence it is precisely the next edge of the algorithm that we will use to merge two sets. Consequently, the algorithm always merges two clusters within minimum distance, implying that this is a greedy optimal algorithm.

Solution to problem 3. (1) Let $v \in V_j(L)$ be in the intersection of two lines ℓ, ℓ'. For v to be a vertex of $VA_0(R)$, we need (i) $\ell, \ell' \in R$, and (ii) none of the j lines of L strict below v be chosen by R. The choices are independent and the result is simply

$$\mathbb{P}(v \in V_0(R)) = \left(\frac{1}{k}\right)^2 \left(1 - \frac{1}{k}\right)^j.$$

(2) Let $X = |V_0(R)|$. By linearity of expectation,

$$\mathbb{E}X = \sum_{j \geqslant 0} \sum_{v \in V_j(L)} \left(\frac{1}{k}\right)^2 \left(1 - \frac{1}{k}\right)^j \geqslant \left(\frac{1}{k}\right)^2 \left(1 - \frac{1}{k}\right)^k |V_{\leqslant k}(L)|.$$

For any fixed R with $m = |R|$ lines, the lower envelope of R uses each line in at most one interval (recall the property of the 0-level), hence it has at most $m - 1$ vertices. Therefore $X \leqslant m - 1$ implies $\mathbb{E}X \leqslant \mathbb{E}m \leqslant n/k$. Combining this with the big inequality above yields

$$|V_{\leqslant k}(L)| \leqslant \frac{n/k}{(1/k)^2(1-1/k)^k} = \frac{nk}{(1-1/k)^k} = \mathcal{O}(nk)$$

since $(1-1/k)^k \geqslant 1/4$ for $k \geqslant 2$, and for case $k = 1$, we trivially have $|V_{\leqslant 1}(L)| \leqslant |V_{\leqslant 2}(L)| = \mathcal{O}(n)$.

Solution to problem 4. We use the following results from lecture 17. For a parameter $r \geqslant 1$, a $(1/r)$ cutting for L, along with the conflict lists $L_\Delta = \{\ell \in L : \ell \text{ crosses triangle } \Delta\}$, can be built in $\mathcal{O}(nr)$ time. We also use the location query time $\mathcal{O}((m + r^2)\log r)$. Both are from Hopcrofts theorem.

We design two routines and take the one that runs faster depending on the problem.

Algorithm A. Cut on L, yielding the $n\sqrt{m}$ term.

- Build a $(1/r)$ cutting of L with $r = \lceil \sqrt{m} \rceil$ and store the conflict lists L_Δ. This is done in $O(nr) = O(n\sqrt{m})$ time.
- Build a point-location structure on the cutting; for each $p \in P$, locate the triangle $\Delta(p)$ inside the cutting that contains p. The cost will be $\mathcal{O}((m + r^2)\log n) = \mathcal{O}(m \log n)$.
- Finally, for each p, scan the list $L_{\Delta(p)}$ and evaluate, in $\mathcal{O}(1)$ time per line, whether $p \in \ell$ for any $\ell \in L_{\Delta(p)}$. Return Yes if a positive is found; else return No.

To see that this algorithm is correct: if $p \in \ell$, then ℓ crosses every cell meeting p so it must be in $\ell \in L_{\Delta(p)}$. And the algorithm does not report Yes otherwise. To upper bound the runtime of the last step, by cutting property

$$\sum_{p \in P} |L_{\Delta(p)}| \leqslant m \cdot (n/r) = \mathcal{O}(mn/r) = \mathcal{O}(n\sqrt{m})$$

so overall Algorithm A runs in $\mathcal{O}((n\sqrt{m} + m)\log n)$ time.

ALGORITHM B. DUAL TRANSFORM, YIELDING THE $m\sqrt{n}$ TERM. We perform dual transform, so that $p \in \ell$ iff $\ell^* \in p^*$. (Edge cases like vertical lines can be handled directly, clearly within our complexity bound.) Now apply ALGORITHM A to the dual problem with the m lines P^* and the n points L^*, where $r = \lceil \sqrt{n} \rceil$. The same analysis yields that ALGORITHM B runs in $\mathcal{O}((m\sqrt{n} + n)\log m + m)$ time.
COMBINING. Run whichever algorithm is faster: for example, we run ALGORITHM A iff $m \geqslant n^2$. In either case, the overall complexity is $\mathcal{O}((\min\{m\sqrt{n}, n\sqrt{m}\} + m + n)\log n)$, as stated.

Solution to problem 5. We use duality transform. For two points (p,q)m the supporting line $\ell(p,q)$ dualizes to $\ell^*(p,q) = p^* \cap q^*$. For any third point r, the perpendicular distance from r to $\ell(p,q)$ is the vertical distance from $\ell^*(p,q)$ to the line r^* measured at the x-coordinate of $\ell^*(p,q)$.
In the original problem, for a fixed base (p,q), among all triangles pqr (where we allow r to vary), the one minimizing area is attained by the r with the closest distance to the line pq (i.e., smallest "height"). A better way to put this is, the line pq partitions the (primal) $\mathbb{R}^2$ into two parts, a "left" and a "right." On each side, there is a point r that is closest to the line pq, acting as the triangle's height. Thus, among all points, only two candidates can possibly minimize area of (p,q,r) with p,q fixed: namely, the closest point on the "left" or the closest point on the "right." So for each (p,q), if we can locate r_1, r_2 efficiently, the problem is solved.
In the dualized view, this is very clean: Under the arrangement of P^*, the base pq becomes the point $\ell^*(p,q)$, and finding r_1, r_2 amounts to finding the two (dualized) lines r_1^*, r_2^* that are immediately above/below the point $\ell^*(p,q)$.
So this reduces to a topological sweep. We compute the arrangement of the n dual lines P^* with $\Theta(n^2)$ intersections. Perform a left-to-right topological sweep, maintaining the vertical order of the lines. When the sweep processes the vertex $\ell^*(p,q)$, the dual to the line segment pq, read off the two immediate neighbors (one above, one below), and translate back to their primal points r_1, r_2. Thus, for each (p,q), we have two candidate triangles: (p,q,r_1), and (p,q,r_2).
To summarize the observations above, our algorithm therefore goes as follows.

- Dualize P and compute the arrangement of the lines p^*.
- Perform a topological sweep of the arrangement: visit every intersection $\ell^*(p,q)$ once, while maintaining the vertical order of the lines at the current x.
- At each visited vertex $v = \ell^*(p,q)$: (i) read the two lines r_1^*, r_2^* that are immediately above/below v, (ii) find their primal points r_1, r_2, and (iii) compute the areas of (p,q,r_1) and (p,q,r_2), updating the running minimum if needed.

We note that if a bundle of $\geqslant 3$ lines meets at v, then some triple is colinear and the minimum area is 0, so we may WLOG assume the points are in general position to begin with.
Finally, this algorithm meets the runtime requirement. An arrangement of n lines has $\mathcal{O}(n^2)$ vertices. A topological sweep enumerates them and updates vertical order with $\mathcal{O}(1)$ work perfex (only swaps). Our per-vertex computation is $\mathcal{O}(1)$. Hence the global runtime is $\mathcal{O}(n^2)$ as required.