CMSC 754: Fall 2021 Dave Mount

Homework 3: Arrangements, Search, and Approximations

Handed out Tue, Nov 16. Due at the start of class on **Tue, Nov 30**. Late homeworks are not accepted (unless an extension has been prearranged) so please turn in whatever you have completed by the due date.

Unless otherwise specified, you may assume that all inputs are in *general position*. Whenever asked to give an algorithm running in $O(f(n))$ time, you may give a *randomized algorithm* whose expected running time is $O(f(n))$. If your algorithm involves a plane sweep of a line arrangement and runs in time $O(n^2 \log n)$, you may assume that there exists a topological plane sweep variant that runs in time $O(n^2)$.

Problem 1. Present $O(n^2)$ time algorithms for the following two problems. (The two solutions are closely related, so you can give one solution in detail and explain the modifications needed for the second one.)

(a) Given a set $S = \{s_1, \ldots, s_n\}$ of non-intersecting line segments in the plane, does there exist a (nonvertical) line ℓ that intersects none of the segments of S such that there is at least one segment of S lying above ℓ and at least one lying below? (See Fig. 1(a).)

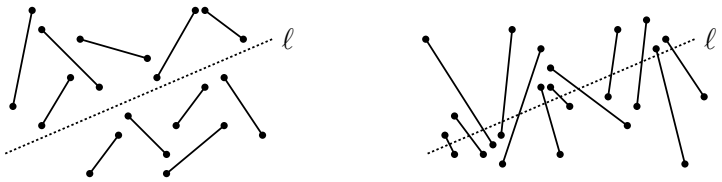


Figure 1: Stabbing segments.

(b) Given a set $S = \{s_1, \ldots, s_n\}$ of non-intersecting line segments in the plane, does there exist a line ℓ that intersects all of the segments of S? (See Fig. 1(b).)

(c) Part (b) looks very similar to a problem we have seen before of stabbing vertical segments. Provide an intuitive explanation as to what goes wrong if you were to try to adapt the LP-based solution to that problem to solve this problem.

Problem 2. *It was pointed out on Piazza that this problem as originally stated does not make sense since there always exist balanced slabs of unbounded height. Here is a version that (I hope) fixes these issues. If you believe that there are nonsensical or trivial solutions, please post a note to Piazza.*

You are given two sets of points R and B (red and blue) in $\mathbb{R}^2$. Let $n = |R| + |B|$ denote the total number of points. A *slab* $S(a, b^-, b^+)$ is the region between two pair of parallel lines, $S(a, b^-, b^+) = \{p : ap_x + b^- \leq p_y \leq ap_x + b^+ p_y\}$ (see Fig. 2(a)). Points lying on the slab's boundary may either be counted or excluded as lying within the slab. (This is needed, since otherwise it is possible to have solutions that are arbitrarily close to optimal, but not optimal.) The *height* of the slab is the vertical distance between the lines, $b^+ - b^-$.

A slab is *balanced* if it contains an equal number of red and blue points. In this problem, we will consider computing the balanced slab of maximum height, which we'll call $\text{maxBS}(R, B)$. Note that the maximum height slab may be of infinite height. To avoid some trivial special cases involving vertical (or nearly vertical) slabs, we add the additional constraint that the slab must contain at least one point from each set, and the slope a of the slab's bounding lines must satisfy $-1 \leq a \leq +1$.

(a) Assuming the standard dual transformation given in class $(a, b) \leftrightarrow y = ax - b$, describe the dual-equivalent formulation of this problem. That is, what is the slab? what is its height? what

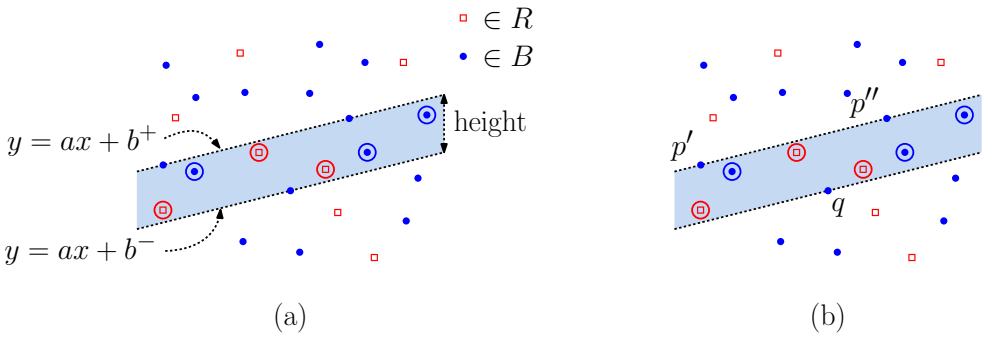


Figure 2: Balanced slab. (Note that the slab shown is *not* the maxBS(R, B).)

condition is satisfied for the slab to have maximum height? (Please read parts (b) and (c) before writing down your answer.)

(b) Assuming that the point set $R \cup B$ is in general position, prove that if maxBS(R, B) is of finite height and its slope is strictly between -1 and $+1$ then it has three points on its boundary, two on one line and one on the other (see Fig. 2(b)).

(c) Present an efficient algorithm, which given R and B, computes the maxBS(R, B). Derive your algorithm's running time and justify its correctness.

Hint: This can be done in time $O(n^2 \log n)$, but it even processing is not obvious. You will get half credit if your algorithm run in $O(n^3)$ time.)

Problem 3. As mentioned in class, a WSPD is an efficient (approximate) representation for the complete graph on a set of points P. Another important structure is the complete *bipartite graph* on a pair of point sets. In this problem, we will explore this topic.

You are given two sets of points in $\mathbb{R}^d$, called R (for red) and B (for blue). Throughout, let $n = |R|+|B|$. A *bichromatic pair* is any pair of points (p, q), where $p \in R$ and $q \in B$. Given a parameter $s > 0$, define a *bichromatic s-WSPD* is a collection of pairs of subsets $\{(R_1, B_1), (R_2, B_2), \ldots\}$ such that

(i) $R_i \subseteq R$ and $B_i \subseteq B$,

(ii) R_i and B_i are s-well separated, and

(iii) for every bichromatic pair (p, q) there is exactly one pair (R_i, B_i) such that $p \in R_i$ and $q \in B_i$.

Given this definition, answer the following questions:

(a) Explain how to modify the standard WSPD algorithm given in class to produce a bichromatic s-WSPD for the sets R and B. (I do not need a complete algorithm description. You can explain what changes to make to the algorithm given in class.)

(b) Show that the asymptotic running time and total size of your bichromatic WSPD construction are the same as for the standard WSPD construction.

(c) Given R and B, define the *average bichromatic distance* to be

$$\Delta(R, B) \;=\; \frac{1}{|R| \cdot |B|} \sum_{p \in R} \sum_{q \in B} \|pq\|.$$

Present an algorithm that, given R, B, and $0 < \varepsilon < 1$, computes an ε-approximation to the average bichromatic distance. That is, your algorithm should return value Δ^* such that

$$\frac{\Delta(R, B)}{1 + \epsilon} \;\leq\; \Delta^* \;\leq\; (1 + \varepsilon) \cdot \Delta(R, B).$$

Your algorithm should run in time $O(n \log n + n/\varepsilon^d)$ time.

Problem 4. In this problem, we will consider how to use/modify range trees to answer a number of queries. In each case, the input is an n-element point set P in $\mathbb{R}^2$. In each case, explain what points are stored in the range tree, what the various levels of the range tree are, and how queries are answered. Finally, justify your algorithm's correctness and derive its storage and running time as a function of n.

(a) A *skewed rectangle* is defined by two points $q^- = (x^-, y^-)$ and $q^+ = (x^+, y^+)$. The range shape is a parallelogram that has two vertical sides and two sides with a slope of $+1$. The lower left corner is q^- and the upper right corner is q^+. The answer to the query is the number of points of P that lie within the parallelogram. (In Fig. 3(a), the answer to the given query is 3.)

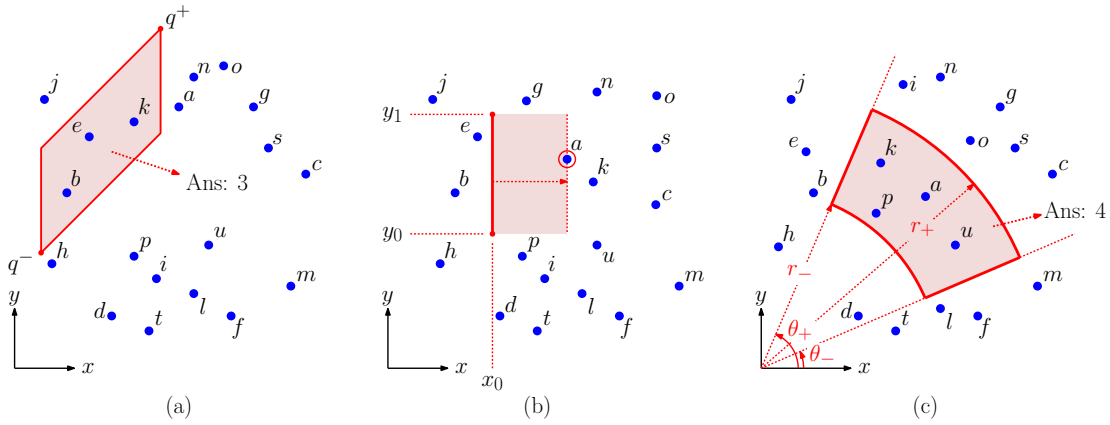


Figure 3: Using range trees to answer various queries.

(b) In a *vertical segment-sliding query* (VSS), you are given a vertical line segment with x-coordinate x_0 and endpoints at y-coordinates y_0 and y_1, where $y_0 < y_1$. The answer to the query is the point that is first hit if the segment is translated to the right. More formally, among the points of P whose y-coordinates lie within the interval $[y_0, y_1]$, the answer is the point with the smallest x coordinate greater than or equal to x_0. If there are no points that satisfy these conditions, the query returns the special value `null`. (For example, in Fig. 3(b), the query returns point a.)

(c) A *sector query* is defined by two nonnegative lengths $r_- < r_+$ and two nonnegative angles $\theta_- < \theta_+ \leq 2\pi$. This defines a shape bounded between two rays emanating from the origin at angles θ_- and θ_+ and two circular arcs between these rays at distances r_- and r_+ from the origin (see Fig. 3(c)). The answer to the query is the number of points of P lying within this region.

Hint: In all cases, it is possible to answer queries in $O(\log^2 n)$ time without the need to resort to cascading.

Challenge problems count for extra credit points. These additional points are factored in only after the final cutoffs have been set, and can only increase your final grade.

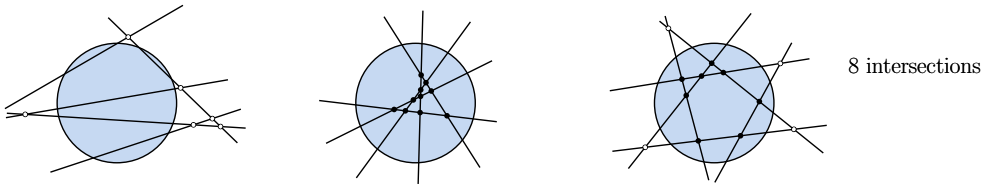
Challenge Problem. You are given a ball B in $\mathbb{R}^2$ and a collection of n lines $L = \{\ell_1, \ldots, \ell_n\}$ in the plane. For each of the following tasks, provide an $O(n \log n)$ time solution. (If you answer (c), you immediately have an answer to both (a) and (b).)

(a) Determine whether none of the lines L intersect within B (see Fig. 4(a)).

(b) Determine whether all of the lines of L intersect each other within B (see Fig. 4(b)).

(c) Count the number of pairs of lines of L that intersect within B (see Fig. 4(c)).

Hint: You may assume that all the lines intersect the ball. Start by labeling the points where the lines intersect the boundary of the ball with the line's index.



(a) (b) (c)

Figure 4: Challenge Problem.