Solutions to Homework 6

Solution 1: To avoid the messy issue of points lying on lines, will consider how to compute the line of minimum separation defect that does not pass through any point of A or B. Given any set of points A in $\mathbb{R}^2$ and any line ℓ, let A^* and ℓ^* denote their associated dual objects, assuming the dual transformation given in class. By the order-reversing property of the transform, the subset of points of A that lie above/on/below ℓ corresponds to the subset of lines of A^* that pass below/on/above the dual point ℓ^*.

To solve the problem of finding the minimum defect line, we construct the line arrangement of the union of dual lines $\mathcal{A} = \mathcal{A}(A^* \cup B^*)$. We then perform a plane sweep through this arrangement. For each line along the sweep-line status, we maintain two values: its *A-depth* and its *B-depth*. We define the A-depth of a point in dual space to be the number of lines of A^* that lie on or above this point, and we define the B-depth analogously for B^*.

We first assert that we can maintain these two depth values for each edge as we sweep $\mathcal{A}$. The initial values at $x = -\infty$ can be computed in $O(n \log n)$ time by sorting the lines by slope. As we sweep each vertex of the arrangement, there are four cases depending on whether the lines are from A^* or B^*. (The details are not important, but for completeness they are shown in Fig. 1(a). The depth values change only for the upper edges, when two lines of different color intersect.) Observe that the depth values assigned to any edge of the arrangement apply to the face lying immediately below it.

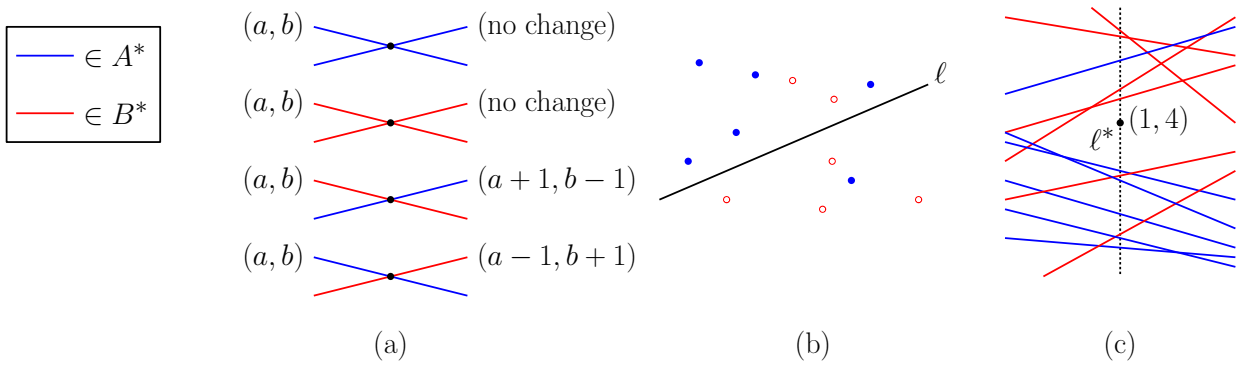


Figure 1: Computing the line minimizing the separation defect.

To compute the line of minimum separation defect, consider any face of the arrangement. By order reversal, the A- and B-depths gives the values of $|A \cap \ell^-|$ and $|B \cap \ell^-|$ (see Fig. 1(b) and (c)). Clearly, $|A \cap \ell^+|$ and $|B \cap \ell^+|$ are just n minus the depth values. Thus, in the traversal, we can compute the separation defects of the face lying directly below each vertex of the arrangement in $O(1)$ time per vertex. (The only face that is not considered is the upper face of the arrangement, but this face cannot minimize the defect.) On termination, we determine the face with the minimum separation defect, and output the dual of any point in this face. Assuming that this is implemented using topological plane sweep, the minimum defect line can be computed in $O(n^2)$ time and $O(n)$ space.

Solution 2: Consider a line segment with endpoints p and q. The dual images of these two points are two lines p^* and q^*. A line ℓ stabs the segment $\overline{pq}$ if and only if p and q lie on opposite sides of ℓ, which is equivalent to saying the dual point ℓ^* lies in the *double wedge* region between p^* and q^* (see Fig. 2(a)).

Our approach to both parts involves sweeping the arrangement of the $2n$ duals of the segment endpoints and computing the *wedge depth* of each face of the arrangement, by which we mean the number of double wedges that overlap this face (see Fig. 2(b) and (c)). This can be computed by performing a plane sweep through the arrangement. For each line ℓ in the sweep-line status, we store:

- Whether ℓ is the upper or lower edge of its double wedge. (This information swaps whenever we sweep the central vertex of the double wedge.)
- The wedge depth of the face immediately below ℓ in the arrangement.

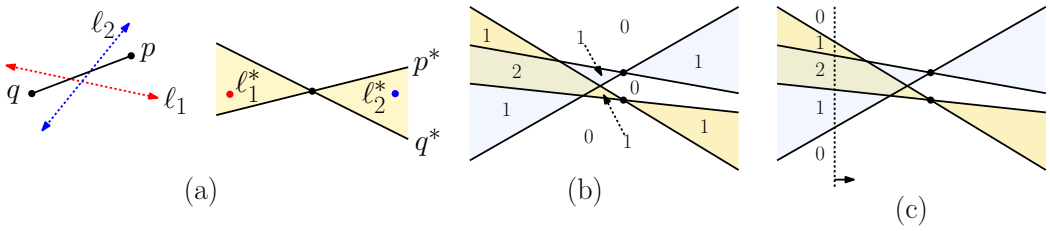


Figure 2: Computing the wedge-depth in a line arrangement.

We assert that we can maintain these two pieces of information in $O(1)$ time for every intersection event we process in the plane sweep. (We omit the straightforward details, but there is only one instance where the wedge depths change, namely when an upper edge of one wedge intersects the lower edge of another. The depth either increases or decreases by 2.)

(a) For this problem, we are seeking any face that has wedge depth zero, excluding the faces associated with the upper envelope and lower envelope. Any point in such a face corresponds to a line that stabs no segment and has at least one endpoint above and one endpoint below, which implies that at least one entire segment lies entirely above and one entirely below. Assuming that this is implemented using topological plane sweep, the minimum defect line can be computed in $O(n^2)$ time and $O(n)$ space.

(b) For this part, all we need to do is to determine whether there is a face in the line arrangement whose wedge depth is exactly n. Any point in such a face corresponds to a line that stabs all the segments. If no such face exists, then no such stabbing line exists. The running time is the same as in (a).

(c) Unlike the vertical segment case, where we knew that the upper endpoint lies above the stabbing line and the lower endpoint lies beneath it, it is not generally easy to determine which is the case with arbitrary segments. For example, consider the point set in Fig. 3. We show two lines that stab all the segments. We have colored the lines based on whether the left endpoint lies above or below the stabbing line.

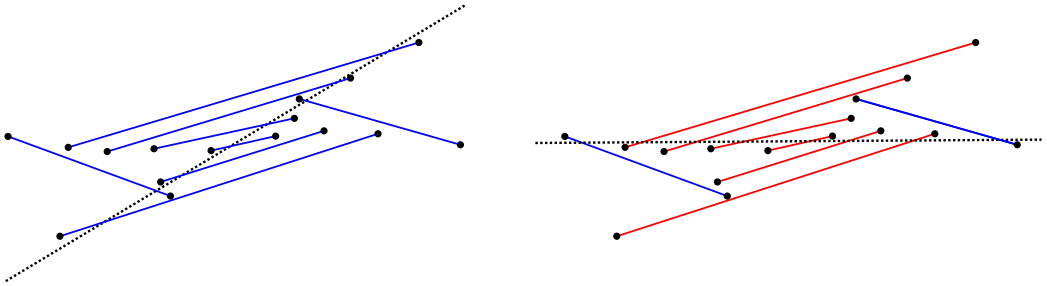


Figure 3: Segments can be stabbed in different ways.

There is a way to solve the problem in $O(n^2)$ time by $O(n)$ invocations of LP. The idea is to sort the line segments by their slopes, and for each consecutive pair of slopes θ_i and θ_{i+1} in sorted order, you can write an LP testing whether there is a stabbing line whose slope lies in the interval (θ_i, θ_{i+1}). Once the slope is constrained, it is possible to know which endpoints are above and which are below.

Solution 3:

(a) Here is a simple greedy algorithm for computing an (r, r)-Delone set for a point set P. Repeatedly, select an arbitrary point $q \in P$ and add it to Q. Then remove all the points of P that lie within distance r of q. When P is empty, return the resulting set Q. Clearly, every point that is removed is within distance r of some point of Q, so the covering condition is satisfied. Also, no two points of Q can be closer than r, since all such points are removed before considering the next candidate.

(b) Hash the points of P into the squares of a grid of side length r. To form the Delone set, let q be a point in any nonempty grid square S. Add q to Q, and remove all the points in the 3×3 set of squares surrounding S (see Fig. 4(a)). When P is empty, return the resulting set Q.

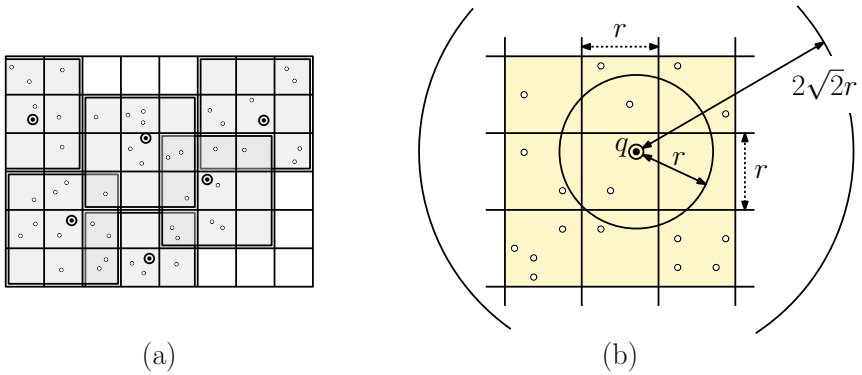


Figure 4: Delone set construction.

The surrounding 3×3 set of squares includes all the points of P that lie within distance r of q, and therefore the resulting set satisfies the packing condition. Also, any point that

is removed lies in one of the squares neighboring S. Since the diameter of each square is $r\sqrt{2}$, the distance between any two points in neighboring squares cannot exceed $(2\sqrt{2})r$ (see Fig. 4(b)). Therefore, the resulting set Q satisfies the covering condition for $t = (2\sqrt{2})r$.

The total running time is $O(n)$ to hash all the points into grid squares. Each nonempty grid square stores the points lying within it. The time needed to select an arbitrary point, visit the 9 surrounding grid squares, and delete all their points is proportional to the number of points removed, which can be at most $O(n)$ over the entire running time of the algorithm.

Solution 4:

(a) There are a number of possible approaches. I think the simplest involves building two separate quadtrees, one for each set. (Alternatively, we could build a single tree for the union of the point sets, but we would need to know for each node the number of blue and red points within its subtree.) We start by building quadtrees for each of R and B. We apply the standard recursive WSPD construction, where the initial bounding box is the same for both trees. The only change to the WSPD construction is that the initial call to the recursive function is ws-pairs(u_0, v_0, s), where u_0 is the root of the red tree and v_0 is the root of the blue tree. By correctness of the standard WSPD construction, each pair that is returned by the function will be well-separated, and each will involve one subtree from the red tree and one subtree from the blue tree, so all the pairs are bichromatic.

(b) The analysis is essentially the same as that for the standard WSPD construction. Observe that if we were to compute the WSPD for the union $R \cup B$, we would have $O(s^d n)$ pairs. We assert that in the bichromatic case, we generate at most twice as many pairs. To see this, recall that each well-separated pair is represented by a pair of nodes (u, v) (see Fig. 5). In our case, u comes from the red tree and v comes from the blue tree. Every time the standard algorithm outputs a well-separated pair $(P(u), P(v))$, we are generating the (at most) two pairs $(P(u) \cap R, P(v) \cap B)$ and $(P(u) \cap B, P(v) \cap R)$. It follows that if there are m pairs in P's s-WSPD, then there are at most $2m$ pairs in the bichromatic s-WSPD. Thus, the asymptotic number of pairs is unchanged.

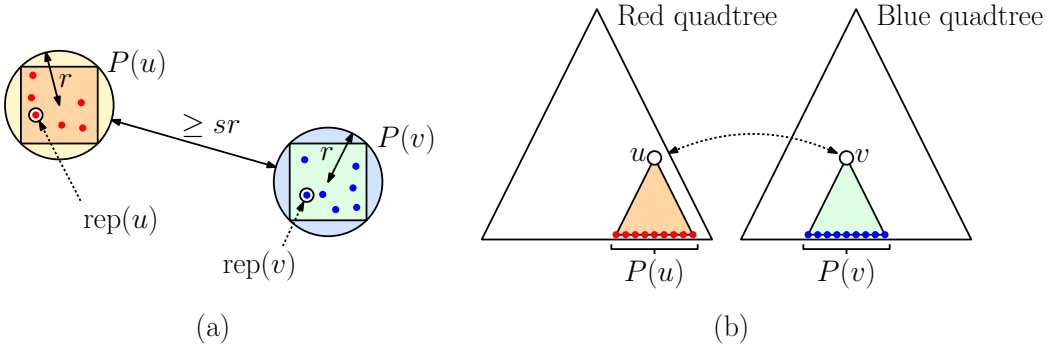


Figure 5: Bichromatic WSPD.

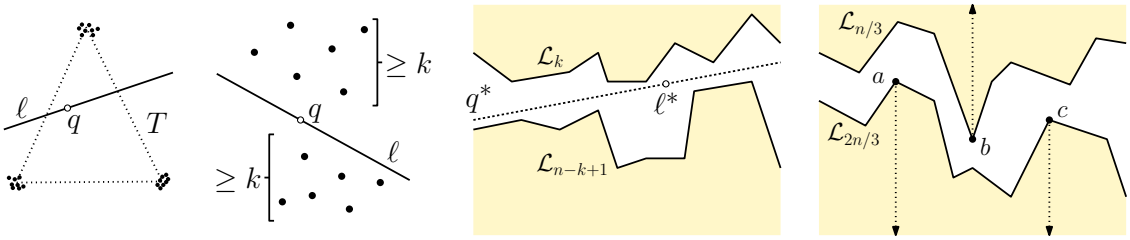
By the standard construction $O(s^d n)$ pairs are generated by the standard construction, and hence this bound applies to the bichromatic version as well. Because each set has at most n

elements, the quadtrees can be computed in $O(n \log n)$ time. Therefore, the overall running time is the same as that of the standard construction, which is $O(n \log n + s^d n)$.

(c) The representatives are computed in the same manner as in the standard case. Recall that each well-separated pair is represented by a pair of nodes, one from the red quadtree and one from the blue quadtree. For each node of each tree, we store the number of points descended from this node and the representative point. Once the tree is computed, these can be calculated by a bottom-up traversal, where the size is the sum of the sizes of the two children nodes and the representative is just the representative from either child. This traversal can be performed in time proportional to the size of the quadtree, which is $O(n)$.

Solution to the Challenge Problem:

(a) Assume that $n \geq 6$, so that $n/3 < \lceil n/2 \rceil$. For simplicity, let's assume n is a multiple of 3. Let T be an equilateral triangle, and define a set consisting of $n/3$ points placed in three tight clusters around each of the vertices of T (see Fig. 6(a)). For any point q in the plane, there exists a line through q such that at least one of the clusters lies entirely on one side of the line. This implies that the other side has at most $n/3$ points. Therefore, q has depth at most $n/3$, which is less than $\lceil n/2 \rceil$ by our assumption that $n \geq 6$. Indeed, this construction shows that for this set, the depth of any point in the plane is roughly $n/3$.



(a) (b) (c) (d)

Figure 6: Tukey depth.

(b) Suppose that levels $\mathcal{L}_k$ and $\mathcal{L}_{n-k+1}$ in the dual arrangement can be separated by a line q^*, where q is the associated point in the primal (see Fig. 6(b) and (c)). We will show that q has depth k. Consider any line ℓ passing through q. By incidence preservation, the associated dual point ℓ^* lies on q^*. Because q^* separates these levels, each point on q^*, including ℓ^*, lies below at least k dual lines and above at least k dual lines of the arrangement. Thus, the associated line ℓ passes through q and has at least k points of P lying below and above it.

Conversely, if there is a point q in the primal plane of depth k, then for each line ℓ passing through q there are at least k points above and below it. This implies that the dual point ℓ^* lies on the dual line q^* and has at least k lines of the arrangement passing above and below it. Thus, the line q^* separates levels $\mathcal{L}_k$ and $\mathcal{L}_{n-k+1}$ of the arrangement.

(c) By applying a topological plane sweep through the dual arrangement $\mathcal{A}(P^*)$, we can compute levels $\mathcal{L}_k$ and $\mathcal{L}_{n-k+1}$ in $O(n^2)$ time. Place the vertices of $\mathcal{L}_k$ in a set A and place the vertices of $\mathcal{L}_{n-k+1}$ in a set B. These two sets together have at most $\binom{n}{2} = O(n^2)$ points. Observe

that the two levels are linearly separable if and only if A and B are linearly separable. In our presentation of linear programming, we showed that determining whether two sets of size m could be solved by linear programming in $O(m)$ time. Thus, in $O(n^2)$ time, we can determine whether A and B are linearly separable, as desired.

(d) The proof is by contradiction. For simplicity, let's assume that n is a multiple of 3, and write $k = n/3$ and $n - k + 1 = 2n/3$. Suppose to the contrary that levels $\mathcal{L}_{n/3}$ and $\mathcal{L}_{2n/3}$ are not linearly separable. The failure to be linearly separable implies that the lower hull of $\mathcal{L}_k$ and the upper hull of $\mathcal{L}_{n-k+1}$ intersect. By the x-monotonicity of these hulls, a vertex of one of these levels lies within the convex hull of the other. (You might take a moment to convince yourself of this fact.) Let's assume (without loss of generality) that a vertex b of $\mathcal{L}_{n/3}$ lies within the convex hull of $\mathcal{L}_{n-k+1}$. It follows that there are two vertices a and c on $\mathcal{L}_{2n/3}$ such that $a_x < b_x < c_x$ and the segment $\overline{ac}$ passes strictly above b (see Fig. 6(c)).

Consider three vertical rays: two going downward from a and c and one going upward from b. By considering their levels, at most $n/3$ lines can intersect any of these rays. On the other hand, any line in the plane must intersect at least one of these rays, and this applies to the lines of P^*. In fact, there is one more, because any line through b (and there is such a line, since it is a vertex of the arrangement) must also hit either a's ray or c's ray. Therefore, the total number of ray-line intersections is at least $n + 1$. But since each ray intersects at most $n/3$ lines, the total number of ray-line intersections is at most $3(n/3) = n$, which yields the contradiction.