Problem 1

Prove that the duality transform introduced in this chapter is indeed incidence and order preserving, as claimed in Observation 8.3.

Proof. Let $(p_x, p_y) \in \mathbb{R}^2$ and $l : y = mx + b$. We then have $p^* \equiv p_x x - p_y$ and $l^* \equiv (m, -b)$.

Incidence Preservation

$\Rightarrow$) Suppose $p \in l$. Then there exists t such that $p = (t, mt + b)$. Therefore $p^* \equiv tx - mt - b$. Notice that l^* is a point on p^* since pluggin in $x = m$ for p^* gives $t(m) - mt - b = 0 - b = -b$. Hence $l^* \in p^*$.

$\Leftarrow$) Now suppose that $l^* \in p^*$. That is $l^* = (t, p_x t - p_y)$ for some t. Therefore $l \equiv tx - p_x t + p_y$. Plugging in $x = p_x$ gives $t(p_x) - p_x t + p_y = p_y$ which means that p is on l. Hence $p \in l$.

Order Preservation

We can describe aboveness of p compared to l with the inequality $p_y > mp_x + b$ and similarly for l^* above p^*. Therefore we have p lies above l iff $p_y > mp_x + b$ iff $-b > mp_x - p_y$ iff l^* lies above p^*. ■

Problem 2

Use Euler's formula to show that the maximum number of faces is $n^2/2 + n/2 + 1$ for an arrangement with $n(n-1)/2$ vertices and n^2 edges.

Proof. From Euler's formula we know that $n_v - n_e + n_f = 2$. We also consider there to be a vertex at infinity meaning

$$\begin{aligned} n_v - n_e + n_f &= 2 \\ \left(\frac{n(n-1)}{2} + 1\right) - n^2 + n_f &= 2 \\ n_f &= n^2 - \frac{n(n-1)}{2} - 1 + 2 \\ n_f &= n^2 - \frac{n^2}{2} + \frac{n}{2} + 1 \\ n_f &= \frac{n^2}{2} + \frac{n}{2} + 1 \end{aligned}$$ ■

Problem 3

Let L be a set of n lines in the plane. Give an $O(n \log n)$ time algorithm to compute an axis-parallel rectangle that contains all the vertices of $A(L)$ in its interior.

> Since we are looking for an axis aligned rectangle, we are interested only in the y and x coordinates of each respective bounding line. Consider the case of finding the x-coordinate for the right side bounding line. If we have a vertical line L positioned sufficiently along the positive x-axis, then the intersection of all the lines in the arrangement with L will have x-coordinates in order of their slopes from largest to smallest slope (where large/small is relative to a slope of 0). This is because the slope dominates the growth of the lines at the distance of L and so the line that is the steepest has the largest x-coordinate for its intersection with L. This same argument holds for the next largest slope and on and on. We can enforce L to also be far enough such that every vertex of $A(L)$ is to the left of it. If we imagine sliding L back towards 0, we can note that the rightmost intersection will occur between two adjacent lines in terms of slope when they were intersecting L originally. Therefore we can find the x-coordiant of the right bounding line as follows.
>
> 1. Sort each line in the arrangement based on slope
> 2. Compute the x-coordinate of the intersection between pairs of lines in this ordering
> 3. Pick the largest of these coordinates
>
> This same technique can be done for the other 3 bounding lines. Performing this algorithm for a given boundary line takes $O(n \log n)$ since sorting takes log linear time, computing the intersections takes linear time, and finding the max takes linear time. Doing this for each bounding line still gives a complexity of $O(n \log n)$.

Problem 4

Let R be a set of n red points in the plane, and let B be a set of n blue points in the plane. We call a line l separator for R and B if l has all points of R to one side and all points of B to the other side. Give a randomized algorithm that can decide in $O(n)$ expected time whether R and B have a separator.

> Consider the set of dual lines R^* and B^*. We can make half planes of opposing direction between R and B and find their intersection in $O(n)$ time with the incremental randomized algorithm for half plane intersection. Since the half planes are opposing directions between R and B, if there is a point in this region, then we have from order preservation that its dual will lie below all of one of R and B and lie above the other. Therefore we can do the incremental half plane algorithm for

the two possible choices of half planes of R going up and B down and vice versa and check if they are feasible.

Algorithm 1 SeparatorExistsQ(R: red points, B: blue points)

```
1. Create the set of half planes $R_u^*$ from $R^*$ oriented up
2. Create the set of half planes $B_d^*$ from $D^*$ oriented down
3. Run the randomized incremental half plane intersection on $R_u^* \cup B_d^*$
     • If feasible, return true otherwise false
4. Create the set of half planes $R_d^*$ from $R^*$ oriented down
5. Create the set of half planes $B_u^*$ from $D^*$ oriented up
6. Run the randomized incremental half plane intersection on $R_d^* \cup B_u^*$
     • If feasible, return true otherwise false
```

Creating the half planes from the dual lines all takes $O(n)$ time. The incremental half plane algorithm also takes $O(n)$ time so in total this algorithm takes $O(n)$ expected time.

Problem 5

Let S be a set of n segments in the plane. A line l intersects all segments of S is called a *transversal* or *stabber* for S. Give an $O(n^2)$ algorithm to decide if a stabber exists for S.

If we consider a single segment, we can take the dual of both of its endpoints. This will give a wedge shape. If we consider some point in the dual plane in this wedge region, then we know the point is above one dual line and below the other. Therefore the dual line of this point will be above one endpoint and below the other, hence it must intersect the original line segment. Therefore if we consider all such possible wedge regions from each segment, if their intersection is non empty then there must be a point and hence a dual line that intersects all the segments. We can determine if the region is non empty by performing the line arrangement algorithm but we modify it to keep track of faces of the arrangement that satisfy this condition. Specifically the algorithm would look like

Algorithm 2 StabberExistsQ(S: segments)

```
1. Create the set of dual lines $P^*$ from the endpoints of the segments in $S$
2. Calculate the arrangement $A(P^*)$ but do the following during it
    • Add each segment's dual lines in as pairs, and for the very first insertion
      mark the dual wedge face
    • With each new dual line pair check if the marked faces are still valid
      and if not unmark them (note that a marked face may be subdivided so
      it is necessary to also check the subdivided parts for validity)
3. Check if the arrangement has any marked faces. If so return true otherwise
   return false
```

As for complexity, we can create the set of dual lines in $O(n)$ time. The modified arrangement calculation still takes $O(n^2)$ time as the extra step of keeping track of marked faces only takes constant time since at most 2 marked regions can be divided, bounding the work done there. The number of faces in the output is $O(n^2)$ so the final step is $O(n^2)$. Hence the overall complexity is $O(n^2)$.