Problem 1

Show that Theorem 7.3 implies that the average number of vertices of a Voronoi cell is less than six.

> ***Proof.*** Consider a voronoi diagram (V, E, F) with n points and therefore n cells. By theorem 7.3, there are at most $3n - 6$ edges in the diagram. Note that the number of vertices of a cell is bounded above by the number of edges is has (since an unbounded cell has 1 more edge than vertices). Therefore the average number of vertices v_{avg} is
>
> $$\begin{aligned} v_{\text{avg}} &= \frac{1}{n} \cdot \sum_{f \in F} (\#\text{ of vertices adjacent to } f) \\ &\leq \frac{1}{n} \cdot \sum_{f \in F} (\#\text{ of edges adjacent to } f) \\ &= \frac{2}{n} \cdot \#(E) \\ &\leq \frac{2}{n}(3n - 6) = 6 - \frac{12}{n} < 6 \end{aligned}$$
>
> By reading the extreme LHS and RHS we have $v_{\text{avg}} < 6$. ■

Problem 2

Show that $\Omega(n \log n)$ is a lower bound for computing Voronoi diagrams by reducing the sorting problem to the problem of computing Voronoi diagrams. You can assume that the Voronoi diagram algorithm should be able to compute for every vertex of the Voronoi diagram its incident edges in cyclic order around the vertex.

> ***Proof.*** Let $S \subseteq \mathbb{R}$ be the finite set of numbers to be sorted. We can associate a voronoi diagram to S based on the set of points $\{(s, 0) : s \in S\} \cup \{(0, \infty)\}$. Note that traversing the boundary formed between $(0, \infty)$ and the other points would give the points in order of their x-coordinates which are the numbers of s. Therefore since the sorting problem has the lower bound of $\Omega(n \log n)$, constructing a voronoi diagram must also have a lower bound of $\Omega(n \log n)$ otherwise we could solve the sorting problem in sub log linear time. ■

Problem 3

Suppose you are given a set of n red points and n blue points in the plane. Describe an $O(n \log n)$-time algorithm for finding, for each blue point, p, the red point that is closest to p.

The key idea is that we can make a voronoi diagram out of the red points and then do point location queries with all the blue points to get the nearest red point to each blue point. The algorithm then is

---

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

---

1. Construct a DCEL of the voronoi diagram of R
2. Construct a point location query structure over this DCEL (ex. path copied slab method)
3. For each point $p \in B$, run a point location query and associate p with the red point in the voronoi cell
4. Return these associations between points in B and R

Breaking down the running time complexity of the algorithm

1. Constructing the DCEL of the voronoi diagram can be done in $O(n \log n)$ time
2. Making the point location query structure can be done in $O(n \log n)$ time as well
3. Each point location query takes $O(\log n)$ time, therefore doing one for each of the n blue points takes $O(n \log n)$

Therefore in total the time complexity of the algorithm is $O(3n \log n) = O(n \log n)$.

Problem 4

Suppose you are given a set, S, of $n > 3$ points in the plane such that no three lie on the same line. Show that a point, p, in S, is on the convex hull of S if and only if p has an unbounded cell in the Voronoi diagram of S.

Proof. We consider both directions

$\Rightarrow$) Let p be on the convex hull of S. Since there are at least 3 points in S, we can take q_1, q_2 to be the neighboring points to p on the convex hull. Let Q_1 and Q_2 be the perpendincular bisectors between pq_1 and pq_2 respectively. Since p is on the convex hull, there are no other points in S that lie outwards of the convex hull and between Q_1 and Q_2. Therefore the cell for p must be unbounded.

$\Leftarrow$) Suppose towards contradiction that the cell for p is unbounded, but p does not lay on the convex hull. Then p must be inside the convex hull. Since the

cell for p is unbounded, there must be a half line eminating from p that is contained in its cell. However, this half line will eventually exit the convex hull, and so a point on the boundary of the convex hull will eventually be closer to a point of the half line. This is a contradiction, hence p itself must be on the convex hull ■

Problem 5

Let P be a set of n points in the plane. Give an $O(n \log n)$ time algorithm to find for each point p in P another point in P that is closest to it.

From problem 1, we know that the number of vertices in a given voronoi cell is on average constant since the number of neighboring cells is proportional to a given cells vertex count. Therefore if we find the voronoi diagram of P, we can then for each point in P look at the neighboring cells and associate with it the closest point in those cells. Since the expected number of neighboring cells is constant, this work takes constant time. Making the voronoi diagram takes $O(n \log n)$ time and finding the nearest point for each point takes $O(1) \cdot O(n) = O(n)$ time. In total then the expected running time of this algorithm is $O(n \log n)$.