Notes in Computational Geometry
Voronoi Diagrams

Prof. Sandeep Sen and Prof. Amit Kumar *

Indian Institute of Technology, Delhi

Voronoi Diagrams

In this lecture, we study Voronoi Diagrams, also known as Thiessen Diagrams or Dirichlet Diagrams. The concept of Voronoi diagrams is developed in section 2 followed by a divide-and-conquer algorithm to construct them in section 3. We start with the Post Office problem in the following section 1, which leads in to the development of Voronoi diagrams.

1 Post Office Problem

The Post Office problem is defined as

Problem 1. Given a set S of n points in a plane, we want to construct a data structure that supports queries of the kind - "For a query point q, which is the closest point in S from q?".

If the points in S are uniformly distributed, a reasonable approach would be to use a grid-based structure. Thus, with a $\sqrt{\mathrm{n}} \times \sqrt{\mathrm{n}}$ grid, we have, on an average, 1 point per square, hence, a constant query time complexity on expectation. This is similar to the grid-based method used for the closest pair problem.

For a general distribution of points, in 1 dimension, an effective approach would be to just sort all the points and performa binary search to answer a query. Here, the query time is $O(\log n)$. The space complexity would be $O(n)$ and preprocessing time $O(n \log n)$.

Moving to 2 dimensions, we start with a case where there are only two points in S. The division of the plane in to the regions of influence of each point is obtained by the perpendicular bisector of the line joining the two points. Thus, once we have the division of the plane in to the two regions of influence, we can determine which region a point lies in by answering a point location query, as was discussed in the previous topic. We can easily extend this idea to 3 points by constructing the perpendicular bisectors of each pair of points. We notice that the 3 bisectors intersect at a single point (since, the 3 points form a triangle) which is the circum-centre of the 3 points. These two cases are depicted in Fig. 1.

* Scribe: Siddharth Srinivasan

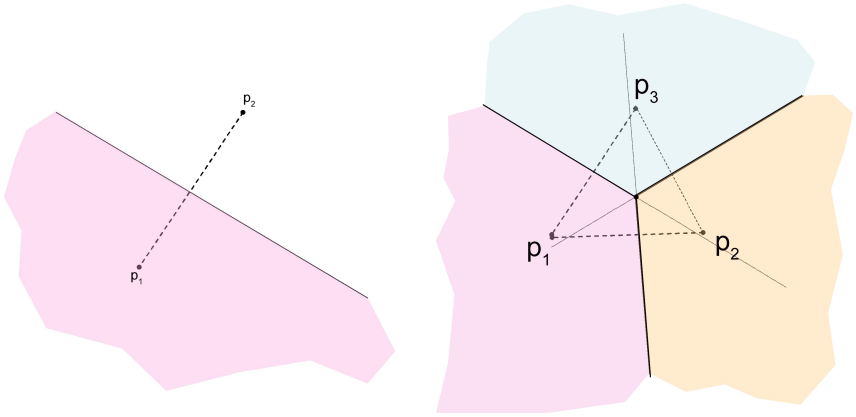

Fig. 1. Planar subdivision into regions of influence of individual points for the cases (a) n=2, and (b) n=3.

2 Voronoi Diagrams

We make a further assumption that no 4 points in the set S are co-circular. Now, proceeding in the manner devised above for the simple cases of 2 and 3 points, we can extend this technique of planar subdivision to any general n $\geq$ 2 by constructing perpendicular bisectors for each pair of neighbouring points. Two points are considered to be neighbouring if the segment joining them does not intersect any other segment smaller than itself. This definition of neighbourhood ensures that the subdivision generated is planar. Such a planar subdivision is called a *Voronoi Diagram* or *Dirichlet diagram* or *Thiessen diagram*. The segments of division are called the *Voronoi edges* and the points of intersection of these edges, the *Voronoi centres*. Each planar face of the subdivision is called a *Voronoi region* and each Voronoi region can be associated uniquely with a point of the input set S. Thus, a Voronoi region of a point signifies the region of influence of the point on the plane, given the other (n-1) points of S. Figure 2 shows an example Voronoi diagram.

Some important features of Voronoi diagrams are:

- Voronoi region of a point, $p_i \in S$, is $\cap_{i \neq j} H_{ij}$, where H_{ij} is the half-space containing p_i and defined by the perpendicular bisector of p_i and p_j. Thus, Voronoi regions are the intersections of half-spaces.
- Voronoi regions are convex regions, since, they are the intersection of convex regions.
- Since, it is a planar subdivision of the space, therefore the number of Voronoi vertices, edges, regions are all linearly related i.e. $O(n)$

Once a Voronoi diagram is created, a point location structure can be created to solve the post office problem.

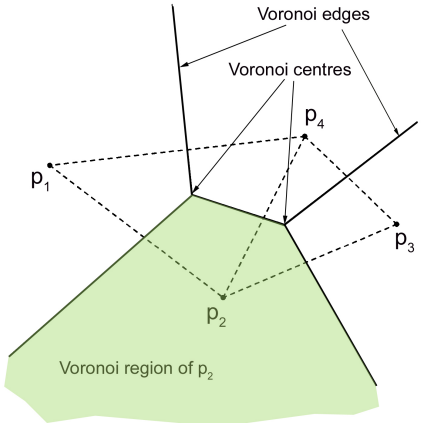

Fig. 2. An example Voronoi diagram showing Voronoi regions, edges and centres

The dual of a Voronoi diagram is a Delaunay triangulation. The dual is obtained by denoting every face by a vertex and adding an edge between two vertices if the two corresponding faces are adjacent. Delaunay triangulation has the good property of producing fat triangles i.e. all the angles of the triangle are maximized. Delaunay triangulation has very useful application in finite element methods, surface reconstruction problems etc. Thus, Voronoi diagrams are useful as a means to construct its dual and since, they are planar subdivisions, therefore, the dual can be constructed in $O(n)$ time and vice versa. However, due to the development of efficient, direct divide-and-conquer algorithms for constructing the Delaunay triangulation of a set of points, this approach of first creating the Voronoi diagram and then its dual is not used in practice due to the overhead of storing the Voronoi diagram.

Construction of Voronoi diagrams

A Voronoi diagram for a set, S, of n points can be constructed by computing the intersection of n half-spaces. We know that the intersection of 2 half-spaces is a dual of convex hulls and therefore, takes $O(n)$ time. Thus, we can easily construct the Vornoin diagram in $O(n^2 \log n)$ time and with certain minor modifications in $O(n^2)$ time. However, we aim to achieve the construction in $O(n \log n)$ time. For this, we use a divide-and-conquer approach, detailed in the next section.

3 Divide-and-Conquer Algorithm for Construction of Voronoi Diagrams

We wish to use the divide-and-conquer paradigm to construct a Voronoi diagram for a set, S, of n points in $O(n \log n)$ time. We know, that the recurrence relation

established by a divide-and-conquer approach is of the form

$$T(n) = 2 \times T(n/2) + f(n)$$

, where, f(n) is the time taken to merge the two half solutions at each step. Thus, to achieve $O(n \log n)$ time complexity, the merge operation must be performed in $O(n)$ time. Thus, a divide-and-conquer algorithm essentially deals with an effecient method of merging, while the divide step can be done easily in $O(n)$ time by dividing the points in to two halves, L and R, about the median x-coordinate.

Thus, the set S has been divided in to two equal halves, L and R, as depicted in fig. 3(a) with L containing all points to the left of the median x-coordinate and R containing all points to the right. Further, we assume that, by recursion, the Voronoi diagrams of L and R, $V(L)$ and $V(R)$ have been constructed (fig. 3(b)). For the base cases, where S contains 2 or 3 points, the Voronoi diagram can be made by perpendicular bisector construction, as outlined earlier. By visual inspection, we notice that we can create a chain about the join of L and R, which is the locus of points equidistant from a point in L and a point in R. Importantly, we claim that the line segments in this chain are the new edges of the Voronoi diagram $V(S)$ i.e. $V(L \cup R)$. And, the edges of $V(L)$ and $V(R)$ remain in $V(L \cup R)$ except for those which are intersected by this chain. We prove this claim in the following.

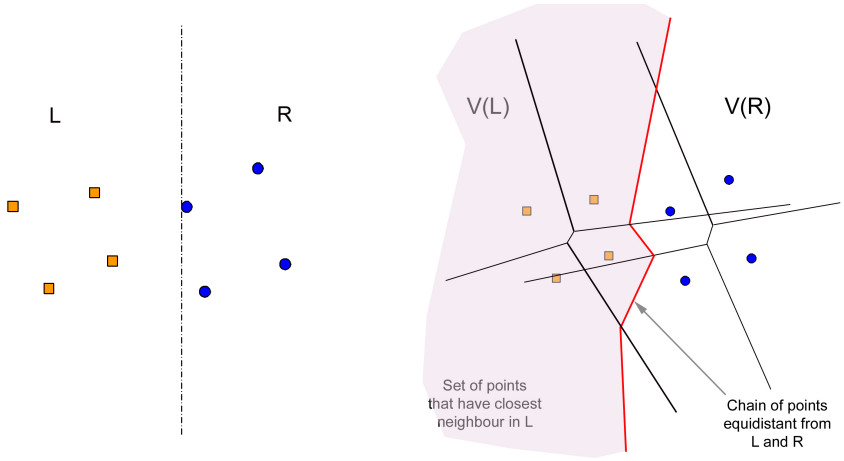

Fig. 3. 3(a) shows the division of S into L and R about median x-coordinate. 3(b) shows the individual Voronoi diagrams $V(L)$ and $V(R)$ of L and R overlaid together.

Let,
C denote the set of points equidistant from a point in L and a point in R,

$dist(p, L)$ = the distance of a point p to the closest point in L, and,
$dist(p, R)$ = the distance of a point p to the closest point in R.

Lemma 1. *C consists of all those Voronoi edges of $V(L \cup R)$ which are defined be a point $l \in L$ and a point $r \in R$ and the edge is defined by l and r.*

Proof. Let $H(L)$ and $H(R)$ denote the set of Voronoi edges of L and R. Suppose a point p has as its nearest left neighbour $l \in L$ and its nearest right neighbour $r \in R$ and is equidistant from l and r, then it should belong to $H(L \cup R)$, since, otherwise, it has exactly one point as its nearest neighbour, contradicting equidistance from l and r.

Also, let e be an edge of $H(L \cup R)$ that is defined by $l \in L$ and $r \in R$. By definition of a Voronoi edge, e is a segment of the perpendicular bisector of $\overline{lr}$. Thus, all points on e are equidistant from l and r and hence, $e \in C$. □

Lemma 2. *C is monotone, i.e. if we direct all edges upwards (or all downwards) then no arrowheads are incident on the same point and C is not disjoint.*

Proof. We show the validity of the claim by some case analysis. We consider cases that contradict Lemma 2 and show that such cases cannot occur. We essentially show that C is a linearly circular group.

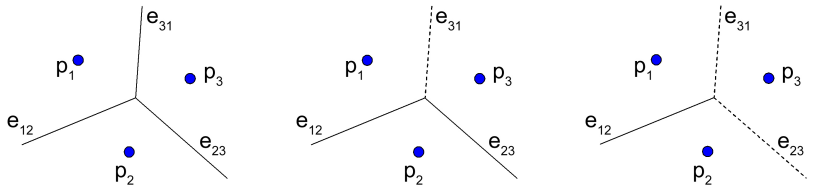

Fig. 4. Proof of lemma 2 by case analysis. 4(a) Case 1, 4(b) Case 2, 4(c) Case 3. Solid edges refer to edges which are both Voronoi edges as well as part of the chain C. Dashed edges refer to edges which are Voronoi edges but are not part of C.

Case 1. See fig. 4(a). If $p_1 \in L$, then $p_2 \in R$ since $e_{12} \in C$. Now, if $p_3 \in R$, then since p_2 and p_3 both lie on the same side, $e_{23} \notin C$, which is a contradiction. And, if $p_3 \in L$, then since now p_1 and p_3 both lie on the same side, $e_{31} \notin C$, which is also a contradiction. Thus, this case cannot occur.

Case 2. See fig. 4(b). If $p_1 \in L$, then $p_2 \in R$ since $e_{12} \in C$. Thus, due to $e_{23} \in C$, $p_3 \in L$. However, this is not possible since both $p_1, p_3 \in L$ and $p_2 \in R$ lies between them. Thus, this case cannot occur.

Case 3. See fig. 4(c). If $p_1 \in L$, then $p_2 \in R$ since $e_{12} \in C$. Now, since $e_{23} \notin C$, therefore $p_3 \in R$. However, then $e_{31} \in C$, which is a contradiction. Thus, this case cannot occur.

$\vdots$

$\vdots$

and so on.

□

Thus, we see that to merge $V(L)$ and $V(R)$, we can construct C and thus

$$V(L \cup R) = C \cup (V(L) \cup V(R)) \setminus (\text{set of edges of } (V(L) \cup V(R)) \text{which are intersected by } C)$$

Now, we describe the method by which the chain C can be costructed.

To construct the chain C,

1. Start from the pair of points that define the lower tangent (or upper tangent) of the convex hulls of L and R.
2. Make the edge consisiting of the perpendicular bisector of the two points extending to $-\infty$.
3. Each time this edge hits an existing edge, one of the pair of equidistant points changes. If the existing edge hit is $\in V(R)$ then current r changes, else current l changes.
   - Make the edge equidistant to this new pair of points. Add this edge to C.
4. Continue in the above manner up till the upper tangent (or lower tangent) of the hulls of L and R is reached.

The above technique requires that we use an efficient method of keeping track of which edge is intersected by C next. This is described below.

1. Denote 2 escort edges, ee_l and ee_r , one each from $V(L)$ and $V(R)$.
2. Initially, $ee_l \leftarrow$ edge denoted by l in $V(L)$ which intersects C first and $ee_r \leftarrow$ edge denoted by r in $V(R)$ which intersects C first.
3. **Repeat**
   (a) Walk along the escort ee_l and shoot a ray from C to find the point of intersection.
   (b) **If** this point of intersection is ahead of the end point of ee_l **then** take a right turn i.e. $ee_l \leftarrow$ next edge of $V(L)$ which shares that end-point and is to the right of current ee_l
   (c) **until** the point of intersection lies on ee_l
4. **Repeat**
   (a) Walk along the escort ee_r and shoot a ray from C to find the point of intersection.
   (b) **If** this point of intersection is ahead of the end point of ee_r **then** take a left turn i.e. $ee_r \leftarrow$ next edge of $V(R)$ which shares that end-point and is to the left of current ee_r
   (c) **until** the point of intersection lies on ee_r
5. Choose i as that point of intersection out of i_l and i_r that has lower y-coordinate.
6. Change escort edge for the one corresponding to i and repeat this process.

Time Complexity. Step 3 above ensures that each edge is traversed only once, since each escort edge once rejected is never required again. Thus, the merging algorithm is $O(n)$. Thus, the entire divide-and-conquer algorithm runs in $O(n \log n)$ time.

In the next lectures, we will continue the discussion on Voronoi diagrams and their construction based on the property of intersection of half-spaces using a lifting transform. However, it must be noted that, the algorithms discussed ahead as well as the direct divide-and-conquer algorithm have asymptotically optimal time complexity of $O(n \log n)$