Ahmet Faruk Ulutaş - 21803717 – CS478 – HW2

1-

procedure bestScoringTriangulation(A, N)

begin

delaunay = divide-and-conquer or another algorithm to create delaunay triangulation

edges = delaunay.GetEdges()

for each edge in edges

begin

A = edge.getA()

B = edge.getB()

otherEdgesA = fetch edges with the same vertex

otherEdgesB = fetch edges with the same vertex

get the edges that have a common point in two lists independent of the principal edge as A and B.

TriangleScore(point a, point b, point c)

end

end

2-

Using the divide-and-conquer algorithm, we can find the two closest points in two different sets in O(nlogn) time.

For this, we first sort all the points according to the x coordinate. Then we divide the plane with all the points into two parts. Recursively, we look for the smallest distance in two subarrays. We create a new array and in this array store all points which are at most smallest distance away from the middle line dividing the two sets. Then we find the smallest distance from the new array and return.

Let T be the time complexity of the above algorithm T(n). Assume we're using an O(nLogn) sorting method. The method separates all points into two sets and calls for two sets recursively. It finds the new array and smallest value in O(n) time after division. In addition, dividing the array around the mid vertical line requires O(n) time. Finally, in O(n) time, the nearest spots in the new array are found. As a result, T(n) may be written as follows:

$T(n) = 2T(n/2) + O(n) + O(n) + O(n)$

$T(n) = 2T(n/2) + O(n)$

$T(n) = T(nLogn)$

3-

The most efficient algorithm that can be used to create a Voronoi diagram is Fortune's algorithm O(nlogn).

```
procedure testVoronoi( planarMap)

flag = Chcek planarMap has valid triangulation

S = set of points list

if flag

        S = fortunesAlgorithm( planarMap)

        return   S

else

        return S

procedure fortunesAlgorithm( planarMap)

Initialize empty eventQueue

Fill the eventQueue with site events for each element in the planarMap.

While the eventQueue is not empty:

   If the next event on the queue is a site event:

      Add the new site to the beachline

   Otherwise it must be an edge-intersection event:

      Remove the squeezed cell from the beachline
```

Cleanup any remaining intermediate state

4-

Using the Closest Pair Property, proves that the set of points is the same as the binary graph of the Voronoi diagram. Closest Pair Property: The circle with diameter pi, pj cannot contain any other point inside, since otherwise pi, pj cannot be closest, and so the center of this circle is on a Voronoi edge common to V(pi) and V(pj) .

To demonstrate this, we show that pi, pj meets the empty circle requirement if and only if;

V(pi) n V(pj) != ∅.

If V(pi) n V(pj) != ∅, pick any point x on the shared edge e(ij) = V(pi) n V(pj). By property of the Voronoi diagram, we have d(x, pi) = d(x, pj) < d(x, pk), for any k != i, j. As a result, the empty circle claim is satisfied by the circle with a center at a x and a radius of d(x, pi).

On the other hand, if C is an empty circle passing through pi, pj, then let x be its center.

Since d(x, pi) = d(x, pj), we must have x ∈ V(pi) n V(pj). Since P is a finite point set in non-degenerate position, we can move x infinitesimally without violating the empty circle condition. This demonstrates that x is on the common border of V(pi) and V(pj).

Using the Closest Pair Property, proves that the set of points is the same as the binary graph of the Voronoi diagram.

“Closest Pair Property: The circle with diameter pi, pj cannot contain any other point inside, since otherwise pi, pj cannot be closest, and so the center of this circle is on a Voronoi edge common to V(pi) and V(pj).”

A Delaunay triangulation is unique if and only if the circumcircle of every triangle contains exactly three points on its circumference: the vertices of the triangle. However, if 4 points are cocircular or there is no triangulation, the Delaunay triangulation is not unique.

5-
Using the divide and conquer strategy, find the voronoi diagram. O(N logN)

Draw lines connecting point pairs with a shared edge in their proximity polygons. O(N)

This approach can also be solved in O(nlogn) since the voronoi diagram may be found in O(nlogn).