Logo, company name

Description automatically generated

CS 478 HOMEWORK 3  
  
ALP TUĞRUL AĞÇALI

21801799

Q1-) The smallest number appears when one polygon completely covers the other. In this case the smallest number is n or m which is the outer polygon’s vertex amount.

Shape, polygon

Description automatically generated

For this example, let’s say the polygon with black edges is called P with n edges (in this example n = 4), and the polygon with red edges is called Q with m edges (int this example m = 5). In this example, The Convex Hull of P ∪ Q is equal to P and number of vertices of the hull is equal to n which is 4.

Figure 1: Two Polygons, One Cover Another

The largest number appears when all edges of small polygon (in terms of number of vertices) are intersecting the bigger one. This time the number of vertices becomes n + m.

Chart, shape, line chart, polygon

Description automatically generated

For this example, let’s say the polygon with black edges is called P with n edges (in this example n = 4), and the polygon with red edges is called Q with m edges (int this example m = 5). In this example, The Convex Hull of P ∪ Q consists of every vertex of these two polygons and vertex amount is n + m which is 9.

Figure 2: Two Polygons

Q2-) If we consider one triangle in a Delanuay Triangulation, we will have three circles for one triangle, which are containing two vertices of the triangle, let’s call them edge circle. And let’s say we have one circumcircle of this triangle. If this circumcircle has a point inside (does not satisfy circumcircle property), one of three edge circle must contain it.

A picture containing accessory

Description automatically generated If one of the edge circles has a point inside, and this point is outside of the given circumcircle (green) it will be inside of the another circumcircle in triangulation.

So, if the circumcircle property is satisfied, the given definition is satisfied too. Therefore, it is the same as the dual graph of the Voronoi diagram of the set of points.

Figure 3: Circumcircle (GREEN) and edge circles (RED)

So, why does this definition give a unique triangulation? This definition holds for the Delanuay Triangulation but may not hold for all triangulations. Thanks to this definition we get a unique triangulation. Because let’s say there is a point inside of this angle circles. This means that the angle between the edge which is the center of the circle, and third point is not big enough. This situation can cause illegal edges too. We know that Delanuay Triangulation does not contain illegal edges and maximizes the minimum angles. So, we can get a unique triangulation.

3-)   
function markCells(M, X, Y):

//empty list for holding points

fullCells = []

//find points that are not full (for calculating distance)

for x in range X:

for y in range Y:

if M[x][y] is not None:

fullCells.add(M[x][y], x, y)

//find empty point’s distance to the full points and set the closest one’s name to it

for x in range X:

for y in range Y:

if M[x][y] is None:

minDistance = maxInt

for pointName, xPoint, yPoint in fullCells:

// manhatanDistance function calculates distance btw points

distance = manhatanDistance(x, y, xPoint, yPoint)

if distance < minDistance:

minDistance = distance

M[x][y] = pointName

Time Complexity = O(2\*(XY) = O(XY)

4-) For finding closest to point in two sets A and B, we firstly need to construct a Voronoi Diagram. (O(NlogN)) and get Delanuay Triangulation using it (O(N)). Then we can find the closest points to points of B using one of the point location algorithms, Triangle Refinement method (firstly find triangle that b lies on then find closest vertex) (O(logN) for each, O(MlogN) in total where M is the point amount of set B)(We used Triangle Refinement method because if given PSLG is triangulation giving it a triangular boundary is O(1) time. So preprocessing is O(1)). While checking points of B we must keep the shortest pair and its distance too.

function closestPairVoronoiDelaunay(A, B):

voronoiDiagram = constructVoronoiDiagram(A) //O(NlogN)

delaunayTriangulation = computeDelaunayTriangulation(voronoiDiagram) //O(N)

points = []

minDistance = infinity

for pointB in B: O(M)

nearestPointA = findNearestPoint (pointB, delaunayTriangulation) //O(logN)

distance = euclideanDistance(pointB, nearestPointA)

if distance < minDistance:

points[0] nearestPointA

points[1]=PointB

minDistance = distance

return minDistance, points

-this problem’s worst case is O(NlogN) + o(N) + O(N)\*O(LogN) => O(NlogN).

-Its best case is O(NlogN) too because of constructing Voronoi diagram.

-If sets are linearly seperable we can use same algorithm with larger boundary triangle.

5-) To solve this question we first need to get a Voronoi Diagram in o(NlogN) time and transform it to a Delanuay triangulation in O(N) time.

function VoronoiDiagram(S):

S1, S2 = Partition(S)

Vor\_S1 = VoronoiDiagram(S1)

Vor\_S2 = VoronoiDiagram(S2)

Vor\_S = Merge(Vor\_S1, Vor\_S2)

return Vor\_S

In this algorithm Partition(s) function is used for partitioning S into two subsets S1 and S2. Merge function is in the following

function Merge(Vor\_S1, Vor\_S2):

pL, pR = initial\_points(Vor\_S1, Vor\_S2)

e = initial\_edge(pL, pR)

v = initial\_vertex(e)

eL = initial\_edgeL(pL)

eR = initial\_edgeR(pR)

while not (pL, pR == t2):

while (intersection(e, eL) == None):

eL = next\_edge\_1(eL)

while (intersection(e, eR) == None):

eR = next\_edge\_2(eR)

intersectionL = intersection(e, eL)

intersectionR = intersection(e, eR)

if is\_closer(intersectionL, v, intersectionR):

v = intersectionL

pL = opposite\_point(eL)

e = bisector(pL, pR)

eL = reverse(eL)

else:

v = intersectionR

pR = opposite\_point(eR)

e = bisector(pL, pR)

eR = reverse(eR)

Vor\_S = DiscardAndCombineEdges(Vor\_S1, Vor\_S2, e)

return Vor\_S

Lastly we need to convert it to the Delanuay Triangulation.

function ComputeDelaunayEdges(Vor\_S):

Delaunay\_edges = []

for edge in Vor\_S.edges:

circumcenter1, circumcenter2 = GetCircumcenters(edge)

Delaunay\_edge = (circumcenter1, circumcenter2)

Delaunay\_edges.append(Delaunay\_edge)

return Delaunay\_edges