Bangladesh University of Engineering & Technology
Department of Computer Science & Engineering
CSE 464: Computational Geometry Sessional
Session January 2018

Assignment 3

Delaunay Triangulation & Voronoi Diagram

May 22, 2018

1 Introduction

In this assignment you will compute Delaunay Triangulation of a set of n points in $O(n^2)$ time according to the algorithm described in chapter 9 of [1]. In addition to that you will compute Voronoi diagram of those points by constructing straight line dual of Delaunay Triangulation. Figure 1 shows Delauney Triangulation and Voronoi Diagram for a set of points. The black points are the input points or sites, the black edges are Delaunay edges and the graph shown in black is a Delaunay Triangulation of the sites. The points colored as red are voronoi vertices, the red colored edges are Voronoi edges and the whole red diagram forms Voronoi diagram.

2 Input

You will have to take input from a file. The first line of the input is a number n indicating the number of points. Each of the next n lines contains a pair of numbers indicating the x-coordinate and y-coordinate of each point.

3 Output

You have to show the Delaunay Triangulation and the Voronoi Diagram graphically. You have to keep option (by pressing some button) of showing i) the Delaunay Triangulation, ii) the Voronoi Diagram.

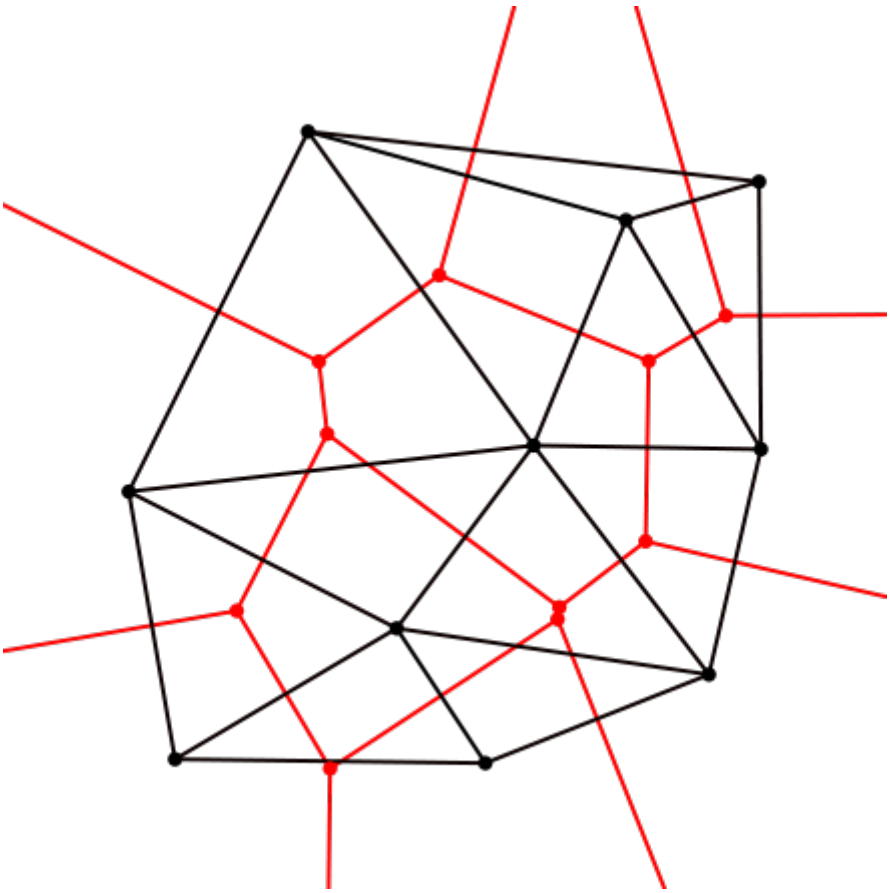

Figure 1: Voronoi Diagram and Delaunay Triangulation (Source: Wikipedia).

4 Guidelines

To construct Delaunay Triangulation, you may want to write a class named Point and Triangle to store the input points and triangles of Delaunay Triangulation. Each triangle consists of three points and it might be helpful to store reference or index of the opposite triangles for each corner of a triangle. For example, consider the triangle T1 in Figure 2. In the object of T1, you may store the reference of index of T2, T3 and T4 along with proper mapping of which triangle is opposite to which corner. Whenever you modify a triangle (either by insertion of new point or by some legalization step), you need to modify the reference of adjacent triangles. For instance, if a point 7 is found to be inside triangle T1 in a later step, the algorithm first creates three new triangles T11, T12, T13 (deleting T1) as shown in Figure 2. So, in this situation you may need to reassign appropriate reference of adjacent triangles for triangles T2, T3 and T4. In the LegalizeEdge function, you might need to pass a point and a triangle as parameters. You may then find the appropriate adjacent triangle (opposite to the point). For example if you call LegalizeEdge with the T11 triangle and the point 7, you would need to find the point 5 (via T3) and then check whether it is inside the circumcircle of the triangle T11.

To construct Voronoi Diagram from Delaunay Triangulation, you need to find circumcenter of each Delaunay triangle. Those circumcenters will form the set of Voronoi vertices. Most of the Voronoi edges are easy to construct. For each Delauney triangle, you need to add edge

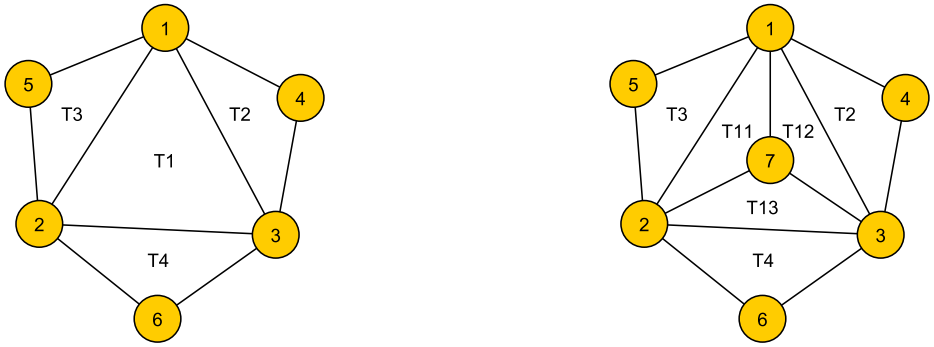

Figure 2: Triangles in Delauney Triangulation.

between the Voronoi vertex defined by that triangle and the adjacent triangles. Finally there are some Voronoi edges which are incident to only one Voronoi vertex (a ray originated from that vertex). Do you get what are the properties of those edges and the Voronoi vertices where they are incident to? If not then you should look closely around such edges, vertices and triangles in Figure 1. Another thing is two Voronoi vertices may coincide (if the points of the corresponding two Delaunay triangles lies on same circle). You actually can handle this case by thinking you have two voronoi vertices with same co-ordinate and an edge (with 0 length!) between them.

5 Important Notes

Please follow the instructions listed below while implementing your assignment:

- Implement using C++/Java programming language.
- Be cautious about floating point arithmetic.

6 Marks Distribution

- Constructing Delaunay Triangulation: 10
- Constructing Voronoi Diagram: 6
- Test case generation: 2
- Graphical Output: 2
- [Bonus] Implement in $O(n \log n)$ time: 5

7 Rules

- You have to submit all your source codes via moodle. All the file name will be in following format

<your 7 digit student id>_<additional name>

For example, the submitted file name would look like 1305999_Delauney.cpp if it is submitted by a student having 1305999 as student id. Name your input file as <your 7 digit student id>_input1.txt, <your 7 digit student id>_input2.txt and so on. Put all your source files, input files and report in a folder (even if you put all your codes in only one file) named after your 7 digit student id and create a **zipped** archive of the folder. Then submit the zip file in moodle. Failure to submit properly will cause 10% deduction of marks.

- Any type of plagiarism is strongly forbidden. -100% marks will be given to the students who will be found to be involved in plagiarism (from book, internet, from senior/classmates code etc.). It does not matter who is the server and who is the client.

8 Deadline

Deadline is set at 10:00 pm, June 22, 2018 for all lab groups.

References

[1] M. d. Berg, O. Cheong, M. v. Kreveld, and M. Overmars. *Computational Geometry: Algorithms and Applications*. Springer-Verlag TELOS, Santa Clara, CA, USA, 3rd ed. edition, 2008.