COL100, Assignment 8
Semester II, 2015-2016

Triangulate Points on a Plane

In this assignment, given a set of points in a plane, you will compute the *triangulation* of these points. We discuss below what triangulation is and how to compute it. You are advised to write functions to perform these tasks.

- Read points: Read points from the input file and store in an array. Assume the input points to have positive integer coordinates. Number these points as 1 to n, where n is the number of points in the input file. Let's call this numbering *input numbering* of points. Later we will sort these points and number them differently. **Please remember that you need to output triangles using input numbers of their end-points**.
- Enclosing rectangle: Assume that the set of points given will have 4 points forming a rectangle that encloses the remaining points.
- Sort points: Order/Sort these points in non-decreasing order of their x-coordinate values. If two points have same x-coordinate values, order them in non-decreasing order of their y-coordinate values.
- Perform triangulation: Triangulation of a set of points returns a set of triangles that cover all the given points among their vertices, and which do not intersect with each other. Implement the following algorithm to compute the triangulation for a given set of points.

  We start at the leftmost point and traverse the points from left to right. Let us number the points 1 to n in left to right ordering. Let's call this numbering *sorted numbering*. Note that sorted numbering may be different from the input numbering and used here to describe the algorithm. The numbers below correspond to sorted numbering of the points. Add an edge (line segment) connecting points 1 and 2. Let E_i denote the set of edges when we reach point i. That is $E_1 = \phi$ and $E_2 = \{1, 2\}$. For $i = 2$ to n, we repeat the following. Suppose we have already traversed from point 1 to i. Consider point $(i + 1)$. In this step we add edges connecting points j and $(i + 1)$ if edge $(j, i + 1)$ does not intersect any other edge already present in E_i, where $1 \leq j \leq i$. In Figure 2 *valid* and *invalid* edges are shown, where valid edges are added to E_i but invalid edges are not added. Invalid edges are shown in dashed lines.

  Let E_n denote the set of edges when we reach point n. The set of triangles formed by edges in E_n produces the desired triangulation. You may need to declare arrays of appropriate size to keep track of edges and triangles. Let there be t triangles. Output the triangles such that the input numbers of the three end-points are in non-decreasing order. For example if you want to output a triangle whose corner points have input numbers $10, 6, 2$, then output $2, 6, 10$ for that triangle. If you want to output triangles having input numbers are $11, 8, 9$ and $10, 6, 2$ respectively, then output $2, 6, 10$ first and then $8, 9, 11$. The triangles should be output in non-decreasing order of the minimum input number associated with its end-points.

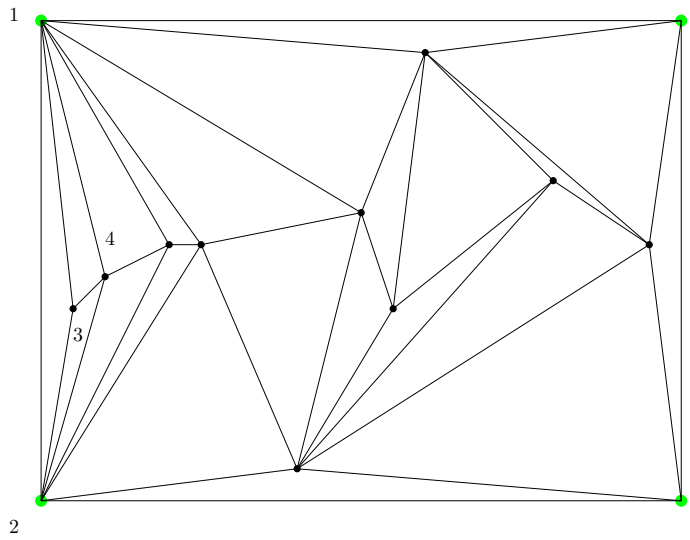

Figure 1: Triangulation of Points

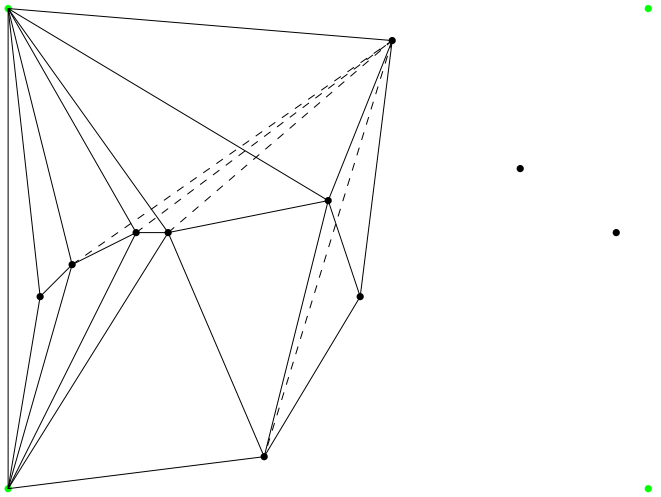

Figure 2: Valid and Invalid Edges

Figure 1 shows the triangulation of a set of points. Notice that while 123, 234 and 134 are valid triangles, 124 is not a valid triangle because it intersects with other valid triangles like 123. In Figure 2, we show valid and invalid edges with some invalid edges shown in dashed lines. Invalid edges are not added into E_i.

Submission Instructions

Submit a single .cpp file. Please adhere to the following submission format.

Input Specifications You will be provided with 'input.txt' which contains (x, y) coordinates for 14 points where all the coordinates are assumed to be positive integers. Input file contains lines like as follows.

5 9
12 15

Output Specifications Write the input numbers for the end-points of the triangles to a file named 'output.txt'. These numbers for each triangle should be in non-decreasing order. Each line should contain details of only one triangle. Each line should look like the following.

2, 7, 9
4, 11, 17

Further Hints

Triangulation requires you to check whether an edge is valid or not. This is done by making sure that the new edge does not intersect with any other edge already included. The crucial part here is checking whether two line segments intersect. Use the following fact to do that. Given two extreme points $(x_1, y_1), (x_2, y_2)$ of a line segment, any intermediate point on the line segment would have coordinates $(\alpha x_1 + (1-\alpha)x_2, \alpha y_1 + (1-\alpha)y_2)$ for some $0 \leq \alpha \leq 1$.

Note that although the input points have positive integer coordinates, the point of intersection of two line segments may have coordinates with floating point values. You may need to do type-casting for that.