Programming Assignment 1: Polygon Overlapping

Sankeerth Durvasula
EE13B102

March 29, 2016

1 Breifing

Two types of verteices have been defined as structs: struct point, and struct pair. The difference is that the coordinates in point are taken in as integer: int x, y, whereas in pair, they're taken in as floating point numbers. Since polygon vertices are integer, it would be a great improvement computationally and algorithmically if we stick to point type structutes. So pair is used only in very particular instances. The struct polygon has also been defined, which is simply a vector of points.

The main routines of the program include the findOverlapPoly(polygon A, polygon B), which is supposed to return true or false if the polygons correspondingly inersect or not, and isBadPoly(int* inputArr, int length), which takes in numbered array obrained from input file, along with it's length.

Dijkstra's algorithm has been used to calculate all pairs shortest paths from the initial polygon.

2 Algorithm

The main function takes in and parses the input line by line. After taking in the first entry, it loops over each line and stores the coordinates in each line in an array, which is then passed to isBadPoly. If it returns true, loop breaks and it prints out "BAD x",x being number of the line and then it exits the program using return 0.

If none of the polygons are bad, a 2D array named distance[n][n] is defined, n is the number of polygons. Running a loop between every pair of polygons, one can find all values of distance matrix. Note that distance[i][j] = -1 is designated as infinite distance.

In the usual way, Dijkstra's algorithm is implemented on this array and the minimum distances have been stored in a minDist[n] array,with minDist[0] = 0. While using Dijkstra's algorithm, a boolen visited[n] array is used. So, after Dijkstra is completed, a loop runs to check whether all visited[n] gives true. If not, "DISCONNECTED x" is printed, where x is such that visited[x] == 0. Then, it exits the program.

If the execution came until this stage, "CONNECTED" is printed out, and row by row, the values of minDist[1], minDist[2], ... minDist[n] are printed out and return 0 then returns from main function.

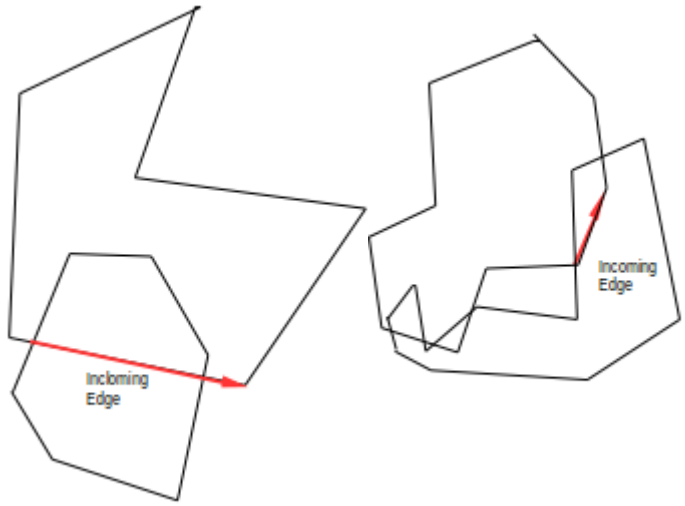
3 Function to find whether polygons intersect

In the function findOverlapPoly(polygon A, polygon B), the main idea is that two overlapping polygons satisfy atleast one of these three properties:

1. A is inside B
2. B is inside A
3. A has an "incoming edge"(defined below)

The two polygon vertices are given in anticlockwise direction. Exploiting this fact, one can assign a direction to each of the edges of the polygon.

Call the first edge of polygon A to cross the boundary of polygon B the "incoming edge" of polygon B. As i turns out, detecting this incoming edge can be done in an elegant fashion, as shown in the code.



Scan each all pairs of line segments of B while traversing A in the anticlockwise direction. At everypair, the following cases might arise:

1. The line segments do not intersect: In this case, continue the search.
2. The line segments intersect with point of intersection not being on the end points: return TRUE
3. The line segments are parallel/coincident: Continue the search.
4. The line segments intersect at one of the end points: There will be multiple cases to take care of to check that the segment from polygon A is indeed going inside B. If so, return true.
5. After this scan, check if any single point in A is inside B. if yes, return true.
6. Do the same for som point B in A. If yes, return true.
7. return false(after all above tests/conditions fail).

4 isInside function

isInside function takes in a pair and polygon as input, and returns true if the pair is strictly inside the polygon, else false. This is done by drawing a horizonral line from (-1, y) to (x, y) where x and y are coordinates of the pair. Now count the number of intersections strictly inbetween these two points with all the sides of the polygon. (Of course there will be some degenerate cases, which have been taken care of in the program, such as the pair shouldn't be on any one of the edges, or

the horizontal line passes through the vertex). If this number turns out to be odd, then the point is inside, otherwise it's outside.

5 Result

Since integer values only have been used to detect incoming edges, and no floating point inequality/equality conditions have been imposed, the algorithm is quite robust in figuring out if the polynomials are indeed robust.