CSE 464: Computational Geometry Sessional

Assignment 1:
Doubly connected edge list, triangulation of a simple polygon, and guarding the polygon

Introduction: In this assignment you shall have a gentle introduction to implementing computational geometry algorithms. We shall implement a naive triangulation algorithm by an uncommon data structure, called doubly connected edge list. As we shall see/have seen in the theory classes, there exist several triangulation algorithms which are better in running time and are much more involved. However, in this very first assignment, instead of emphasizing on better algorithms, we shall learn how a good and specific data structure can be used in the implementation of a simple algorithm.

Step 1: Read the text: Learn Doubly connected edge list from [dB] Section 2.2. This is a data structure that is used in many geometric representations. We shall use it to represent a simple polygon and its triangulation. The text describes an example on how a doubly connected edge list can represent a simple polygon that also has holes. The main motivation of this data structure is to move around a vertex and around a face and from one triangle to another efficiently. Moving around a vertex or a face should not take more than linear steps and moving from one triangle to another should take a single step.

Step 2: Implement doubly connected edge list: Once you understand that, you shall implement it in C so that a simple polygon without any holes can be represented by a doubly connected edge list (easier than a polygon with holes, right?). Write a program that can take as input the x-y coordinates of the vertices of a simple polygon in clockwise order. Store this polygon by a doubly connected edge list. Draw the polygon and the corresponding doubly connected edge list in the output. Show the doubly connected edge list by double arrows as well as a table as shown in the text [dB]. Try some examples to store a polygon and its triangulation and verify visually whether your representation is correct or not (start with small polygons of only 4-5 vertices).

Step 3: Triangulation: This step is the most involved. Implement the "triangulation algorithm by ear removal" as described in the text [O] Section 1.4-1.6. Although the text gives the algorithm and the pseudo code, it does not talk about doubly connected edge list to store the triangulation. You can get the idea about the pseudo code from the text, but you shall have to implement the doubly connected edge list of your own. As you proceed and partially finish the triangulation, you shall have to update your doubly connected edge list to store the triangulation completed so far. Once the triangulation is finished, your data structure should contain the entire triangulation. Show step by step the triangulation and the update in the doubly connected edge list in the output.

Step 4: Guarding the polygon: After Step 3, you have the input polygon and its triangulation ready in your doubly connected edge list. Read Section 3.1 of [dB]. Implement the 3-coloring of the polygon. Use the doubly connected edge list to move from triangle to triangle. Show step by step in the output the coloring of the vertices of the polygon. Then, place guards as described in the text and show them in the output. (Note that you may not need to find the dual tree of the triangulation. That was only for justification.) Also show how many guards you have used.

Remember from the theory class that this number should not be more than n/3, where n is the number of vertices in the input polygon.

What we shall see in your assignment:

- We shall ask you to give as input a simple polygon that does not have any holes and then to show its doubly connected edge list (Step 2), its triangulation steps and the update of the doubly connected edge list (Step 3), and finally its coloring and guarding (Step 4). Be prepared to take as input a polygon that has many vertices, say 20.
- We shall ask you to explain how you have updated your doubly connected edge list for triangulation and how you have used the list for finding the coloring and guarding. We shall ask you to explain how many steps your algorithm is taking to move around a vertex or a face and from one triangle to another.
- We shall ask you to explain your code and algorithms. We may also ask you to change the code to accommodate some changes and to verify whether you have done the assignment yourself.
- We shall ask basic questions (both implementation related and theoretical questions) related to doubly connected edge list, triangulation, coloring, guarding, etc. This is like a viva.