CSCI 534: Homework 02

Nathan Stouffer – Collaborated with Elliott Pryor

Problem 1

Assume you are given a planar subdivision with n faces in a DCEL. (You may assume that the planar subdivision does not contain any holes, i.e., there are no nested faces.) Give pseudo-code for an algorithms that given a vertex v of the DCEL, outputs all neighbors of v.

Answer: Here is a quick prose description of the algorithm and the pseudo-code is given below in Algorithm 1. We are given a vertex v in a DCEL and we want to compute the neighbors of v. We begin by initializing a stack. Since we have a DCEL, we can get the twin of an incident edge of v; label it e (e has v as its destination). Then push the origin of e to the stack. Then jump to a new edge $\overline{e}$: the twin of e.next. The edge $\overline{e}$ also has v as its destination so we can push the origin of $\overline{e}$ and repeat this process until we return to the original edge e.

Algorithm 1 Computing the neighbors of v

```
1:  function NEIGHBORS(v)
2:      nbhd ← empty stack
3:      e ← v.inc_edge.twin
4:      nbhd.push(e.orig)
5:      ē ← e.next.twin
6:      while e ≠ ē do
7:          nbhd.push(ē.orig)
8:          ē ← ē.next.twin
9:      end while
10:     return nbhd
11: end function
```

Now we will present a proof of correctness.

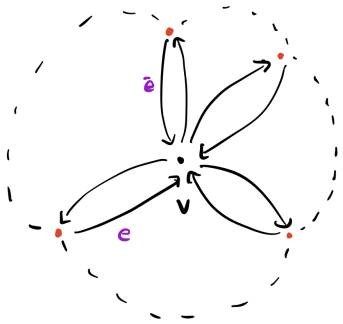

Figure 1: Neighbors of v

Problem 2

Assume you are given a planar subdivision of $O(n)$ size in a DCEL. (You may assume that the planar subdivision does not contain any holes, i.e., there are no nested faces.) Describe an algorithm that for a given point p in the plane finds the face in the subdivision that contains it. Your algorithm should run in $O(n)$ time. You do not have to write pseudo-code, but please make clear what DCEL operations you are using. Also please make sure the analysis is detailed enough to justify the $O(n)$ runtime clearly.

Answer: PROSE DESCRIPTION

We assume that no vertices in the DCEL share x coordinates and that the point p is entirely inside a face in the DCEL (not a vertex or on an edge).

Algorithm 2 Find the face where p resides

```
1: function FindFace(DCEL, p)
2:     for face in DCEL do
3:         cnt ← CountIntersections(face, p)
4:         if cnt ≡ 1 mod 2 then
5:             return face
6:         end if
7:     end for
8: end function
1: function CountIntersections(face, p)
2:     cntr ← 0
3:     h ← horizontal line passing through p_y
4:     for each edge e incident to face do
5:         l ← line passing through the edge e
6:         x ← x coordinate of the intersection between h and l
7:         if x ≥ p_x and x is between e.orig_x and e.twin.orig_x then
8:             cntr ← cntr + 1
9:         end if
10:    end for
11:    return cntr
12: end function
```

We claim the run time of this algorithm is $O(n)$.
PROOF OF CORRECTNESS

Problem 3

Assume you are given a collection of n circles $\{C_1, \ldots, C_n\}$ in $\mathbb{R}^2$, where circle C_i is presented as its center point $q_i = (x_i, y_i)$ and radius $r_i > 0$. Present an $O(n \log n)$ time algorithm that determines whether any two circles intersect. Note that one circle may be nested within another without intersecting (see Figure 1). Your algorithm should either output that there is no intersection, or that there is at least one intersection, and if so it will output the indices of i and j of two circles C_i and C_j that intersect. Irrespective of the number of intersecting pairs, it need only output one intersecting pair.

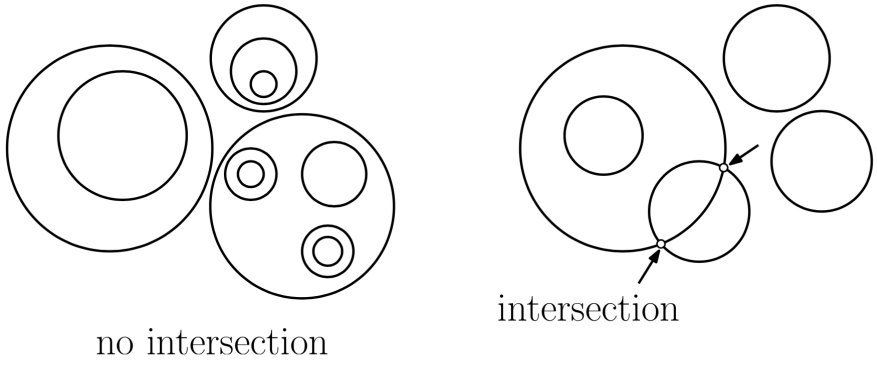

Figure 2: Problem 3: Intersection

Hint: Use plane-sweep. Explain clearly (1) what the sweep-line status stores and what data structure is used to store this information and (2) what future events are stored and what data structure is used. You may assume that you have access to whatever primitive operations that you need in constant time. For example, if you want to determine (a) whether two circles intersect, (b) the coordinates of an intersection, (c) the intersection of a line with a circle, (d) whether a point is contained within a circle's interior, etc., you may simply assume the existence of a function that runs in $O(1)$ time. As always, you may make whatever general-position assumptions you like.

Problem 4

I have had a few people ask about drawings and making figures. One tool that I like to use is Ipe (written by Otfried Cheong). Ipe allows you to draw content on layers and show and hide the different layers. Layers are very helpful if, for example you want to draw a point set and then show how some data structures in an algorithm change as you sweep across the point set. Other vector graphics tools such asIllustrator and Inkscape are also quite good.

Setup Ipe `http://ipe.otfried.org/`, Illustrator, or Inkscape (or another vector graphics tool) to create 3 images of the state of the sweep line algorithm described in problem 3.