# CS310 Computational Geometry
# Spring 2019


| | | | |
|---|---|---|---|
| **Assignment:** | **5** | **Professor:** | **Dianna Xu** |
| **Due Date:** | **3/28/19** | **E-mail:** | dxu@cs.brynmawr.edu |
| **Office:** | **Park 203** | **URL:** | http://cs.brynmawr.edu/cs310 |


## Assignment 5

**1.** Let $P = \{p_1, p_2, \ldots, p_n\}$ be a set of $n$ points.

1. Give an $O(n \log n)$ algorithm to find: for each point $p_i \in P$, the point $p_j \in P$ $(i \neq j)$ that is closest to it. Your output should be a list of $n$ pairs $(p_i, p_j), i = 1, \ldots, n$, where $p_j$ is the nearest neighbor of $p_i$. If you think of each pair as a directed edge from $p_i$ to $p_j$, then your algorithm produces the Nearest Neighbor Graph as known in graph theory.
2. Note that what we described above is a form of nearest-neighbor search (more specifically, the 1-nearest-neighbor search or 1NNS. However, in the classic definition of the NNS problem, we are given a query point $q \notin P$, and asked to find the nearest neighbor of $q$ in $P$. Can you adapt your algorithm to perform 1NNS? What is your algorithm's run time for $m$ query points?

**2.** This problem demonstrates an important application of computational geometry to the field of amphibian navigation (AKA wireless network planning and routing). A frog who is a afraid of the water wants to cross a stream that is defined by two horizontal lines $y = y^-$ and $y = y^+$. On the surface there are $n$ circular lily pads whose centers are at the points $P = \{p_1, ..., p_n\}$. The frog crosses by hopping across the circular lily pads. The lily pads grow at the same rate, so that at time $t$ they all have radius $t$ (see Fig.1(a)). Help the frog out by presenting an algorithm that in $O(n \log n)$ time determines the earliest time $t^*$ such that there exists a path from one side of the stream to the other by traveling along the lily pads (see Fig. 1(b)).

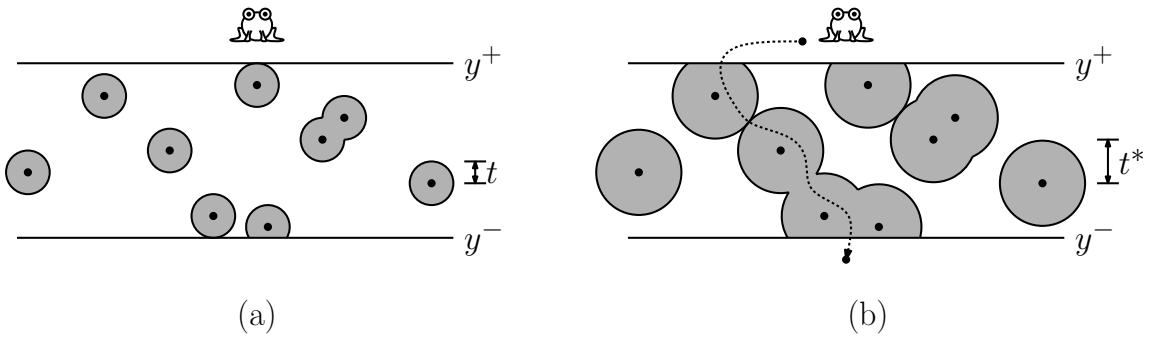


Figure 1: Problem 2

**3.** In this problem we will investigate the properties of a variant of the Voronoi diagram, in which we replace the notion of "nearest" with "farthest". Given a set $P = \{p_1, p_2, \ldots, p_n\}$ of point sites and site $p_i \in P$, recall that we defined $\mathcal{V}(p_i)$ to be the set of all points in the plane that are closer to $p_i$ than to any other site in $P$. Define $\mathcal{F}(p_i)$ to be the set of all points in the plane that are *farther* from $p_i$ than from any other site of $P$. As we did with Voronoi diagrams, we define

the *farthest-point Voronoi diagram* to be the union of the boundaries of $\mathcal{F}(p_i)$ for all points in $P$. Let $\mathcal{NVD}(P)$ denote the *nearest-point Voronoi diagram* of $P$ and let $\mathcal{FVD}(P)$ denote the *farthest-point Voronoi diagram* of $P$.

1. Show that $\mathcal{F}(p_i)$ is a convex polygon (possibly unbounded).
2. Unlike the nearest-point Voronoi diagram, a site may not contribute a region to the farthest point diagram. Show that $\mathcal{F}(p_i)$ is nonempty *if and only if* $p_i$ is a vertex of the convex hull of $P$.
3. Draw a picture of three points in the plane. Show (in different colors) $\mathcal{NVD}(P)$ and $\mathcal{FVD}(P)$. Label each region according to which sites are closest and farthest.
4. Repeat (c), but this time with four points, such that only three of the four points are on the convex hull.
5. Repeat (d), but this time with all four points on the convex hull. (Try to avoid degeneracies, such as four cocircular points or parallel sides.)

**4.** The Euclidean metric is but one way to measure distances (this is the metric that we used when discussing Voronoi diagrams). In this question, we will examine another distance metric, called the $L_\infty$ metric, which is defined as follows

$$\text{dist}_\infty(p, q) = \max(|p_x - q_x|, |p_y - q_y|).$$

1. Given a point $q$, describe the set of points that are at $L_\infty$ distance $w$ from $q$.
2. Given two distinct points $p$ and $q$, describe the set of points that are equidistant from $p$ and $q$ in the $L_\infty$ distance. *Hint:* The bisector is a polygonal curve. As a function of the coordinates of $p$ and $q$, indicate how many vertices this curve has, and where these vertices are located. I would like the exact formula, and not simply a high-level description. If you are having trouble visualizing, I recommend writing a small plotting program in Processing, Mathematica or some other language that allows easy graphics output. It is not necessary to submit the plotting program or provide its sample runs. It is only meant to guide your insights.