Carlos Segarra Thursday, September 26th

Discrete and Algorithmic Geometry: Problems 4 and 5

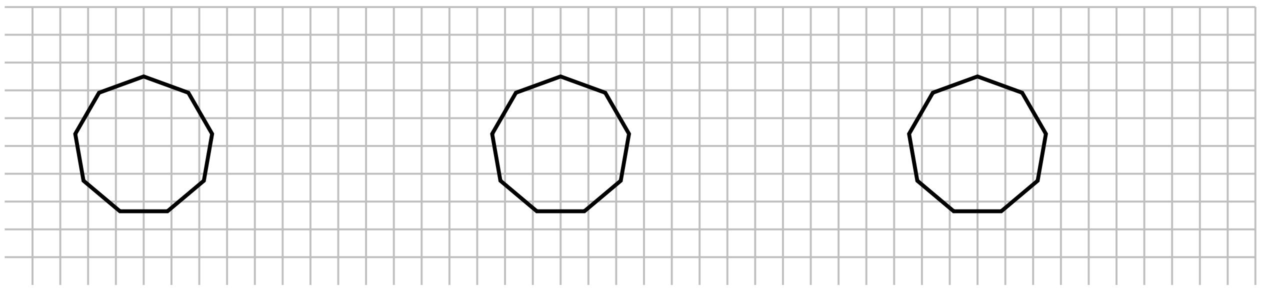
4. Propose an algorithm that, given a point p external to a convex polygon $\mathcal{P}$, finds the point of $\mathcal{P}$ closest to p. What happens if instead of finding the closest point we look for the farthest? What if we restrict the search to the vertices of $\mathcal{P}$?

Let $\{q_1, \dots, q_n\}$ be the set of vertices of $\mathcal{P}$ in clock-wise order. And let f be the function s.t. for all $q \in \mathcal{P}$, sends p to $d(q,p)$. Before starting with the proof, let us make two observations:

Observation 1. *f is an uni-modal application. Alternatively, its derivative is a linear funcion that has only one zero.*

Observation 2. *Let q_{min} and q_{min+1} be the closest and second to closest vertices from $\mathcal{P}$ to p. Then, the closest point in $\mathcal{P}$ to p is either (i) q_{min} or (ii) contained in the $q_{min}q_{min+1}$ segment.*

Note that the second observation applies symetrically to the farthest point in $\mathcal{P}$. Further, once we find the two closest points, checking wether we are in case (i) or case (ii) from Obs 2 can be done in constant time testing if p belongs to the triangle drawn by the line throguh p parallel to $q_{min}q_{min+1}$ and the two lines through p_{min} orthogonal to the segments incident to p_{min}.



Below, we include the Algorithm for computing the closest point to a polygon, note that, in order to find the farthest we could use the same exact algorithm

Algorithm 1 Given a polygon $\mathcal{P}$ given by its vertices $\{q_1, \dots, q_n\}$ and a point p, find the closest point in $\mathcal{P}$ to p.

```
1: $start \leftarrow 1$
2: $end \leftarrow \frac{n}{2}$
3: while $start \leq end$ do
4:     $\Delta_A \leftarrow d(q_{start+1}, p) - d(q_{start}, p)$
5:     $\Delta_B \leftarrow d(q_{end+1}, p) - d(q_{end}, p)$
6:     switch sign($\Delta_A$), sign($\Delta_B$) do
7:         case ++
8:         case +−
9:             $start \leftarrow end$
10:            $end \leftarrow end + \frac{end-start}{2}$
11:        case −+
12:            $end \leftarrow end - \frac{end-start}{2}$
13:        case −−
14:            $end \leftarrow \frac{end}{2}$
15: if $q \in \Delta(q_{min}, \bar{p_1}, \bar{p_2})$ then
16:     $q_{min} = \underset{q \in \{q_{start}, q_{end}\}}{\text{argmin}} (d(q,p))$
17: else
18:     $q_{min} \leftarrow \text{line}(\perp q_{start}q_{end}, p) \cap q_{start}q_{end}$
19: $d_{min} \leftarrow d(q_{min}, p)$
20: return $d_{min}$
```

Running Time Assuming all operations with fixed argument size take constant time, the only step proportional to the size of the input is the while loop. Within the loop we only do steps that take constant time and we iter at most $\log(n)$ times, being n the number of vertices. Hence the complexity of our algorithm is $\mathcal{O}(n)$.

5. Propose an algorithm that, given two disjoitn convex polygons, $\mathcal{P}$ and $\mathcal{Q}$, finds the closest pair of points $p \in \mathcal{P}$ and $q \in \mathcal{Q}$.