CSC279 HW5

Hanzhang Yin

Nov/5/2023

Collaborator

Chenxi Xu, Yekai Pan, Yiling Zou, Boyi Zhang

Question 15

PROBLEM 15—Closest point, farthest point.

1. Assume q is inside P. We want to find the closest point to q in $\{p_1, \ldots, p_n\}$.
2. Assume q is outside P. We want to find the closest point to q in $\{p_1, \ldots, p_n\}$.
3. Assume q is inside P. We want to find the farthest point to q in $\{p_1, \ldots, p_n\}$.
4. Assume q is outside P. We want to find the farthest point to q in $\{p_1, \ldots, p_n\}$.
5. Assume q is inside P. We want to find the closest point to q on P.
6. Assume q is outside P. We want to find the closest point to q on P.
7. Assume q is inside P. We want to find the farthest point to q on P.
8. Assume q is outside P. We want to find the farthest point to q on P.

Answer:
Question 1 - 4 Reasoning: The distance from q to the vertices p_i is unimodal function along the ordered sequence. This is true whether q is inside or outside P. Along the vertices, we can use ternary (binary) search on sequence of vertices to find MIN or MAX distance.

1. **Problem 1:** q inside P; find the closest point to q in $\{p_1, \ldots, p_n\}$.
    - **Solution:** $O(\log n)$ time.
    - **Reasoning:** The distance function is unimodal along the vertices, allowing ternary search.
2. **Problem 2:** q outside P; find the closest point to q in $\{p_1, \ldots, p_n\}$.

- **Solution:** $O(\log n)$ time.
- **Reasoning:** Similar to Problem 1, use ternary search due to the unimodal distance function.

3. **Problem 3:** q inside P; find the farthest point from q in $\{p_1, \ldots, p_n\}$.
   - **Solution:** $O(\log n)$ time.
   - **Reasoning:** Unimodal distance function allows ternary search for the maximum.
4. **Problem 4:** q outside P; find the farthest point from q in $\{p_1, \ldots, p_n\}$.
   - **Solution:** $O(\log n)$ time.
   - **Reasoning:** Same as Problem 3, apply ternary search on the unimodal distance function.

Question 5 - 6 Reasoning:
The distance from q to the boundary of the convex polygon P is a convex function along the perimeter, regardless of whether q is inside or outside P. (i.e. The distance decreases to a minimum point and then increases, forming a single through)

1. **Problem 5:** q inside P; find the closest point to q on P.
   - **Solution:** $O(\log n)$ time.
   - **Reasoning:** Perform binary search over edges to find the closest point where the minimum distance occurs.
2. **Problem 6:** q outside P; find the closest point to q on P.
   - **Solution:** $O(\log n)$ time.
   - **Reasoning:** Same as Problem 5, use binary search to find the closest point on edges.

Question 7 - 8 Reasoning:
The farthest point from q can lie anywhere along the perimeter of the convex polygon P, necessitating a check of all edges and vertices. The distance function for farthest points is not unimodal, preventing the use of efficient binary or ternary search methods. Hence, without preprocessing, we need $\Omega(n)$ time to do them.

1. **Problem 7:** q inside P; find the farthest point from q on P.
   - **Solution:** $\Omega(n)$ time.
   - **Reasoning:** Requires examining all edges using rotating calipers for antipodal points.
2. **Problem 8:** q outside P; find the farthest point from q on P.
   - **Solution:** $\Omega(n)$ time.
   - **Reasoning:** Similar to Problem 7, needs $\Omega(n)$ time to check all edges.

PROBLEM 16

Proof. □

PROBLEM 17

```
Input:
    - P = {p1, p2, ..., pn}: Set of n points in the plane
    - r: Radius of each circle Ci
    - l: Radius of the target circle Q, where l > r

Output:
    - G: Set of "good" points

Algorithm FindGoodPoints(p_1, ..., p_n, r, l):

1. Build the Delaunay triangulation (DT) of the points {p_1, ..., p_n}.
    - Time complexity: O(n log n)

2. Initialize an empty list GoodPoints.

3. For each point p_i in {p_1, ..., p_n}:
    - Time complexity: O(n)
    a. Initialize an empty list of intervals I_i.

    b. For each neighbor p_j of p_i in DT:
        - Time complexity: O(1).
        i. Compute d = distance between p_i and p_j.

        ii. If d <= 2l:
            - Compute theta_j = angle between vector (p_j - p_i) and the x-axis.
            - Compute phi_j = arccos(d / (2 * (l + r))).
                (Ensure the argument of arccos is within [-1, 1].)

            - If phi_j is real:
                - Add the interval [theta_j - phi_j, theta_j + phi_j] to I_i.

    c. Sort the intervals in I_i.
        - Time complexity: O(1), every Voronoi point have constant neighbor.
        - Merge overlapping intervals to get the union.


    d. If the union of intervals in I_i covers [0, 2pi]:
        - p_i is not good.
    e. Else:
        - p_i is good.
```

```
    - Add p_i to GoodPoints.

4. Return GoodPoints.
```

PROBLEM 18