CHAPTER 2
Geometric Searching

- Problems frequently arise in
  - geographic applications
  - database management

- Four separate cost measures

1. Query time. How much time is required, in both the average and worst cases, to respond to a single query?
2. Storage. How much memory is required for the data structure?
3. Preprocessing time. How much time is needed to arrange the data for searching?
4. Update time. Given a specific item, how long will it take to add it to or to delete it from the data structure?

- First problem

Problem S.1 (RANGE SEARCHING—COUNT). Given $N$ points in the plane, how many lie in a given rectangle with sides parallel to the coordinate axes?
Search - Counting

- **Given:** \( n \) points in the plane.
- **Find:** \( N(a, b, c, d) = \) number of points within the rectangle defined by corners \( a, b, c, d \).

Related Problem

- **Given:** \( n \) points in the plane.
- **Find:** \( Q(p) = \) number of points with smaller \( x \)- and \( y \)-coordinates than respectively \( p.x \) and \( p.y \).
- \( N(a, b, c, d) = Q(a) - Q(b) - Q(d) + Q(c) \)
Computational Geometry

Search - Counting; Locus Strategy

- Subdivision can be determined in $O(n^2)$ time.
- Space required: $O(n^2)$.
- Query time: $O(\log n)$. 
Table I

<table>
<thead>
<tr>
<th>Query</th>
<th>Storage</th>
<th>Preprocessing</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(\log N)$</td>
<td>$O(N^2)$</td>
<td>$O(N^2)$</td>
<td>Above method</td>
</tr>
<tr>
<td>$O(\log^2 N)$</td>
<td>$O(N\log N)$</td>
<td>$O(N\log N)$</td>
<td>4</td>
</tr>
<tr>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td>$O(N)$</td>
<td>No preprocessing</td>
</tr>
</tbody>
</table>

1. *Location problems*, where the file represents a partition of the geometric space into regions, and the query is a point. Location consists of identifying the region the query point lies in.

2. *Range-search problems*, where the file represents a collection of points in space, and the query is some standard geometric shape arbitrarily translatable in space (typically, the query in 3-space is a ball or a box). Range-search consists either of retrieving (*report* problems) or of counting (*census* or *count* problems) all points contained within the query domain.
2.2 Point-Location Problems

Problem S.3 (Convex Polygon Inclusion). Given a convex polygon $P$ and a point $z$, is $z$ internal to $P$?

Problem S.2 (Polygon Inclusion). Given a simple polygon $P$ and a point $z$, determine whether or not $z$ is internal to $P$.

Theorem 2.1. Whether a point $z$ is internal to a simple $N$-gon $P$ can be determined in $O(N)$ time, without preprocessing.
2 Geometric Searching

Figure 2.5 Single-shot inclusion in a simple polygon. There is one intersection of $l$ with $P$ to the left of $z$, so $z$ is inside the polygon.

- **degenerate situations**

- to avoid them choose $p' = (x'_1, y'_1)$

\[ y' = (\min_i y_i) - 1 \]

\[ x' = x_i*, \text{ where} \]

\[ x_i* : |x_i* - x| = \min_{i \in \{1, \ldots, n\}} |x_i - x| \]

Here $p = (x, y)$ is a query point.
2.2 Point-Location Problems

Figure 2.6 Division into wedges for the convex inclusion problem. 1. By binary search we learn that \( z \) lies in wedge \( p_1q_2p_2 \). 2. By comparing \( z \) against edge \( p_1p_2 \) we find that it is external.

**Theorem 2.2.** The inclusion question for a convex \( N \)-gon can be answered in \( O(\log N) \) time, given \( O(N) \) space and \( O(N) \) preprocessing time.
• extension to star-shaped polygons
  
  (wedge method)

2. Geometric Searching

Figure 2.7 A star-shaped polygon.

• The kernel of a simple \( N \)-gon can be found in \( O(N) \) time (later)

Theorem 2.3. The inclusion question for an \( N \)-vertex star-shaped polygon can be answered in \( O(\log N) \) time and \( O(N) \) storage, after \( O(N) \) preprocessing time.

• extension to monotone polygons
  
  (interval method)

- query in \( O(\log n) \)
- storage in \( O(n) \)
- prep. in \( O(n) \)

\( \text{convex} \subseteq \text{star-shaped} \subseteq \text{general} \)

\( \subseteq \text{monotone} \)