Fortune's Algorithm for Voronoi Diagrams

Theoretical Description

Fortune's algorithm is a **sweep line algorithm** to construct the **Voronoi diagram** of a set of points (called *sites*) in the plane. The Voronoi diagram partitions the plane into regions where each region contains all points closest to a particular site.

The algorithm uses a horizontal sweep line that moves from top to bottom, maintaining a dynamic structure called the **beach line** which represents the boundary between processed and unprocessed parts of the plane. The beach line consists of arcs of parabolas, each associated with a site.

Key data structures:

- **Event Queue:** A priority queue storing *site events* (when the sweep line reaches a site) and *circle events* (when an arc disappears).
- **Beach Line Status:** A balanced binary search tree representing the sequence of arcs on the beach line.

Types of events:

- **Site Event:** Occurs when sweep line encounters a new site; a new arc is inserted into the beach line.
- **Circle Event:** Occurs when an arc on the beach line shrinks to a point and disappears, creating a Voronoi vertex.

Algorithm

Algorithm 1 Fortune's Algorithm for Voronoi Diagram

```
Input: Set of sites $S = \{s_1, s_2, \ldots, s_n\}$
Output: Voronoi diagram of $S$
Initialize event queue $Q$ with all site events, ordered by decreasing $y$-coordinate
Initialize empty beach line data structure $T$
while $Q$ is not empty do
    $e \leftarrow$ next event from $Q$
    if $e$ is a site event then
        HandleSiteEvent($e$, $T$, $Q$)
    else if $e$ is a circle event then
        HandleCircleEvent($e$, $T$, $Q$)
    end if
end while
Finalize all unfinished edges in the Voronoi diagram
```

Site Event Handling:

```
procedure HandleSiteEvent($e$, $T$, $Q$)
    Find arc $\alpha$ above site $e$
    Replace $\alpha$ with three arcs: left, new (for $e$), right
    Check for new circle events caused by breakpoints and add to $Q$
end procedure
```

Circle Event Handling:

```
procedure HandleCircleEvent($e$, $T$, $Q$)
    Remove disappearing arc from $T$
    Add Voronoi vertex at event point
    Update neighboring arcs and check for new circle events, add to $Q$
end procedure
```

Complexity Analysis

- **Time Complexity:** Each event is processed in $O(\log n)$ time due to balanced tree and priority queue operations. Since there are $O(n)$ site events and $O(n)$ circle events, total time complexity is

$$\boxed{O(n \log n)}$$

- **Space Complexity:** The data structures (event queue and beach line) require $O(n)$ space.

Summary Table

Step	**Description**	**Complexity**
Initialization	Insert all site events into event queue	$O(n \log n)$
Event Processing	Handle site and circle events	$O(n \log n)$
Finalization	Complete unfinished edges	$O(n)$