Point Location

(most slides by Sergi Elizalde and David Pritchard)
Definition

• Given: a planar subdivision $S$
• Goal: build a data structure that, given a query point, determines which face of the planar subdivision that point lies in
• Details: planar subdivision given by:
  – Vertices, directed edges and faces
  – Perimeters of polygons stored in doubly linked lists
  – Can switch between faces, edges and vertices in constant time
First attempt

- Want to divide the plane into easily manageable sections.
- Idea: Divide the graph into slabs, by drawing a vertical line through every vertex of the graph.
- Given the query point, do binary search in the proper slab.
Analysis

- **Query time**: $O(\log n)$
- **Space**: $O(n^2)$
- As a few people in the audience observed, the space can be reduced to $O(n)$ by using “persistent” data structures. See 6.854, Lecture 5 for details.
Second attempt

- Too much splitting!
- Idea: stop the splitting lines at the first segment of the subdivision
- We get a trapezoidal decomposition $T(S)$ of $S$
- The number of edges still $O(n)$
Assumptions/Simplifications

• Add a bounding box that contains $S$
• Assume that the $x$-coordinates of coordinates and query are distinct
  1. Randomly rotate the plane, or
  2. Use lexicographic order
Answering the query

• Build a decision tree:
  – Leaves: individual trapezoids
  – Internal nodes: YES/NO queries:
    • *point query*: does $q$ lie to the left or the right of a given point?
    • *segment query*: does $q$ lie above or below a given line segment?
Decision tree: Example
DT Construction: Overview

1. Initialization: create a $T$ with the bounding box $R$ as the only trapezoid, and corresponding DT $D$

2. Compute a random permutation of segments $s_1 \ldots s_n$

3. For each segment $s_i$:
   A. Find the set of trapezoids in $T$ properly intersected by $s_i$
   B. Remove them from $T$ and replace them by the new trapezoids that appear because of the insertion of $s_i$
   C. Remove the leaves of $D$ for the old trapezoids and create leaves for the new ones + update links
Some notation

Segments $\text{top}(\Delta)$ and $\text{bottom}(\Delta)$:
Some notation, ctd.

Points $\text{leftp}(\Delta)$ and $\text{rightp}(\Delta)$:

Each $\Delta$ is defined by $\text{top}(\Delta)$, $\text{bottom}(\Delta)$, $\text{leftp}(\Delta)$, $\text{rightp}(\Delta)$
Some notation, ctd.

- Two trapezoids are *adjacent* if they share a vertical boundary.
- How many trapezoids can be adjacent to $\Delta$?
Adding new segment $s_i$

- Let $\Delta_0 \ldots \Delta_k$ be the trapezoids intersected by $s_i$ (left to right)
- To find them:
  - $\Delta_0$ is the trapezoid containing the left endpoint $p$ of $s_i$ – find it by querying the data structure built so far
  - $\Delta_{j+1}$ must be a right neighbor of $\Delta_j$
Updating T

- Draw vertical extensions through the endpoints of $s_i$ that were not present, partitioning $\Delta_0 \ldots \Delta_k$
- Shorten the vertical extensions that now end at $s_i$, merging the appropriate trapezoids
Updating D

- Remove the leaves for $\Delta_0 \ldots \Delta_k$
- Create leaves for the new trapezoids
- If $\Delta_0$ has the left endpoint $p$ of $s_i$ in its interior, replace the leaf for $\Delta_0$ with a point node for $p$ and a segment node for $s_i$ (similarly with $\Delta_k$)
- Replace the leaves of the other trapezoids with single segment nodes for $s_i$
- Make the outgoing edges of the inner nodes point to the correct leaves
Analysis

- **Theorem:** In the expectation we have
  - Running time: $O(n \log n)$
  - Storage: $O(n)$
  - Query time $O(\log n)$ for a fixed $q$
Expected Query Time

• Fix a query point \( q \), and consider the path in \( D \) traversed by the query.

• Define
  – \( S_i = \{s_1, s_2, ..., s_i\} \)
  – \( X_i \) = number of nodes added to the search path for \( q \) during iteration \( i \)
  – \( P_i \) = probability that some node on the search path of \( q \) is created in iteration \( i \)
  – \( \Delta_q(S_i) \) = trapezoid containing \( q \) in \( T(S_i) \)

• From our construction, \( X_i \leq 3 \); thus \( E[X_i] \leq 3P_i \)

• Note that \( P_i = Pr[\Delta_q(S_i) \leftrightarrow \Delta_q(S_{i-1})] \)
Expected Query Time ctd.

• What is $P_i = \Pr[\Delta_q(S_i) <> \Delta_q(S_{i-1})]$?
• Backward analysis: How many segments in $S_i$ affect $\Delta_q(S_i)$ when they are removed?
• At most 4
• Since they have been chosen in random order, each one has probability $1/i$ of being $s_i$
• Thus $P_i \leq 4/i$
• $E[\sum_i X_i] = \sum_i E[ X_i ] \leq \sum_i 3P_i \leq \sum_i 12/i = O(\log n)$
Expected Storage

• Number of nodes bounded by $O(n) + \sum_{i} k_{i}$, where $k_{i} =$ number of new trapezoids created in iteration $i$

• Define $d(\Delta, s)$ to be 1 iff $\Delta$ disappears from $T(S_{i})$ when $s$ removed from $S_{i}$

• We have $\sum_{s \in S_{i}} \sum_{\Delta \in T(S_{i})} d(\Delta, s) \leq 4|T(S_{i})| = O(i)$

• $E[k_{i}] = [\sum_{s \in S_{i}} \sum_{\Delta \in T(S_{i})} d(\Delta, s)]/i = O(1)$
Expected Time

• The time needed to insert $s_i$ is $O(k_i)$ plus the time needed to locate the left endpoint of $s_i$ in $T(S_i)$

• Expected running time $= O(n \log n)$
Extensions

• Can obtain worst-case $O(\log n)$ query time
  – Show $O(\log n)$ for a fixed query holds with probability $1 - 1/(Cn^2)$ for large $C$
  – There are $O(n^2)$ truly different queries