Chapter 1 – Introduction

Slides are gratitude of Craig Gotsman

Course notes, D. Mount
Course slides, C. Gotsman

Assessment

- 6 Homework Assignments (65%). Primarily theoretical problems.
  - (7 homework, only the 6 better ones are counted)
- Final Exam (10%)
- Midterm (10%)
- Max(Final, Midterm) (10%)
- Class Participation (5%).

Grads have one more question in each hw.

Syllabus

- Introduction
- Basic techniques
- Basic data structures
- Polygon triangulation
- Linear programming
- Range searching
- Point location
- Voronoi diagrams
- Duality and Arrangements
- Delaunay triangulations
- Computer graphics applications

Lecture Topics

- Sample problems
- Basic concepts
- Convex hull algorithms

Questions?
Nearest Neighbor
- Problem definition:
  - Input: a set of points (sites) $P$ in the plane and a query point $q$.
  - Output: The point $p \in P$ closest to $q$ among all points in $P$.

- Rules of the game:
  - One point set, multiple queries

- Applications:
  - Store Locator
  - Cellphones

The Voronoi Diagram
- Problem definition:
  - Input: a set of points (sites) $P$ in the plane.
  - Output: A planar subdivision $S$ into cells. One cell per site. A point $q$ lies in the cell corresponding to a site $p \in P$ if $p$ is the nearest site to $q$.

Point Location
- Problem definition:
  - Input: A partition $S$ of the plane into cells and a query point $p$.
  - Output: The cell $C \in S$ containing $p$.

- Rules of the game:
  - One partition, multiple queries

- Applications:
  - Nearest neighbor
  - State locator

Point in Polygon
- Problem definition:
  - Input: A polygon $P$ in the plane and a query point $p$.
  - Output: true if $p \in P$, else false.

- Rules of the game:
  - One polygon, multiple queries

Convex Hull
- Problem definition:
  - Input: a set of points $S$ in the plane.
  - Output: Minimal convex polygon containing $S$.

Shortest Path
- Problem definition:
  - Input: Obstacles locations and query endpoints $s$ and $t$.
  - Output: the shortest path between $s$ and $t$ that avoids all obstacles.

- Application: Robotics.
**Range Searching and Counting**

- **Problem definition:**
  - Input: A set of points $P$ in the plane and a query rectangle $R$.
  - Output: (report) The subset $Q \subseteq P$ contained in $R$.
  - (count) The size of $Q$.

- **Rules of the game:**
  - One point set, multiple queries.
  - Application: Urban planning, databases.

**Visibility**

- **Problem definition:**
  - Input: a polygon $P$ in the plane and a query point $p$.
  - Output: Polygon $Q \subseteq P$, visible to $p$.

- **Rules of the game:**
  - One polygon, multiple queries.
  - Applications: Security.

---

**Questions?**

**Basic Concepts**
# Complexity (reminder)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Definition</th>
<th>&quot;Nickname&quot;</th>
</tr>
</thead>
<tbody>
<tr>
<td>( f(n) = \mathcal{O}(g(n)) )</td>
<td>( \exists N, C \forall n \geq N f(n) \leq Cg(n) )</td>
<td>( \leq )</td>
</tr>
<tr>
<td>( f(n) = \mathcal{Ω}(g(n)) )</td>
<td>( g(n) = \mathcal{O}(f(n)) )</td>
<td>( \geq )</td>
</tr>
</tbody>
</table>

# Representing Geometric Elements

- Representation of a line segment by four real numbers:
  - Two endpoints \( (p_1, p_2) \)
  - One endpoint \( p_1 \), a slope \( \alpha \), and length \( d \)
  - One endpoint \( p_1 \), vector direction \( v \) and parameter interval length \( t \)
- Parametric form:
  \[
p(t) = p_1 + t(p_2 - p_1) = (1-t)p_1 + tp_2, \quad t \in [0,1]
\]
- Different representations may affect the numeric accuracy of algorithms...

# Convex Hull Algorithms

- The convex hull of a set \( S \) is unique.
- The boundary of the convex hull of a point set is a polygon on a subset of the points.
Convex Hull – Naive Algorithm

- **Description:**
  - For each pair of points construct its connecting segment and supporting line.
  - Find all the segments whose supporting lines divide the plane into two halves, such that one half plane contains all the other points.
  - Construct the convex hull out of these segments.

- **Time complexity:**
  - All pairs: \( O(n^2) \)
  - Check all points for each pair: \( O(n) \)
  - Total: \( O(n^3) \)

Possible Pitfalls

- Degenerate cases – e.g., 3 collinear points. Might harm the correctness of the algorithm. Segments AB, BC, and AC will all be included in the convex hull.

- Numerical problems – We might conclude that none of the three segments belongs to the convex hull.

Convex Hull – Graham’s Scan

- **Idea:** Sort the points according to their x coordinates. First we construct only the upper CH.
- Process the points from the leftmost to rightmost.
- Maintain the upper CH of all points from the leftmost one to the currently processed scanned point.
- Develop the left-turn criteria for the last 3 processed points:
  - If we need to turn left when traveling along these points, the middle one is NOT on the upper CH, and we delete it.
  - Note: After deletion, we have new 3 points to consider.

The Algorithm

- Sort the points in increasing order of x-coord:
  - \( p_1, \ldots, p_n \)
  - \(^*\) Note – this is the only part not done in \( O(n) \) time \(^*/\)
  - Push(S, \( p_1 \)); Push(S, \( p_2 \));
  - For \( i = 3 \) to \( n \) do
    - While Size(S) \( \geq 2 \) and Orient(\( p_i \), top(S), second(S)) \( \leq 0 \) \(^*\) left turn \(^*/\)
      - Pop(S);
    - Push(S, \( p_i \));
  - Print(S);

Graham’s Scan – Time Complexity

- **Sorting** – \( O(n \log n) \)
- If \( D_i \) is number of points popped on processing \( p_i \)
  - Time \( \sum_{i=1}^{n} (D_i + 1) = n \sum_{i=1}^{n} D_i \)
  - Each point is pushed on the stack only once.
  - Once a point is popped – it cannot be popped again.
  - Hence \( \sum_{i=1}^{n} D_i \leq n \)
  - Question: What is actually \( \sum_{i=1}^{n} D_i \)?

Graham’s Scan – a Variant

- Assume the points are given in increasing x-coord order.
- **Time Complexity:** \( O(n \log n) \)

**Question:** What are the pros and cons of this algorithm relative to the previous?
**Convex Hull - Divide and Conquer**

- **Algorithm:**
  - Find a point with a median x coordinate (time: $O(n)$)
  - Compute the convex hull of each half (recursive execution)
  - Combine the two convex hulls by finding common tangents.
  
- **Complexity:** $O(n \log n)$

\[ T(n) = 2T\left(\frac{n}{3}\right) + O(n) \]

---

**Finding Common Tangents**

A tangent line - a line cutting the CH at a single point

Consider a line passing through a vertex $v'$ of $H_B$. How can we determine if $v'$ is a tangent to $H_B$?

---

**Finding Common Tangents**

To find lower tangent:

- Find $a$ - the rightmost point of $H_A$ $O(n)$
- Find $b$ – the leftmost point of $H_B$
- While $ab$ is not a lower tangent for $H_A$ and $H_B$, do:
  - If $ab$ is not a lower tangent to $H_A$, do $a = a-1$
  - Move one point clockwise
  - If $ab$ is not a lower tangent to $H_B$, do $b = b-1$
  - Move one point counterclockwise

---

**Finding Common Tangents**

- $H_A$
- $H_B$
Algorithm:
- Find a point $p_1$ on the convex hull (e.g. the lowest point).
- Rotate counterclockwise a line through $p_1$ until it touches one of the other points (start from a horizontal orientation).

Question: How is this done?
- Repeat the last step for the new point.
- Stop when $p_1$ is reached again.

Time Complexity: $O(nh)$, where $n$ is the input size and $h$ is the output (hull) size.
- Best alg in 2D: $O(n \log h)$
When designing a geometric algorithm, we first make some simplifying assumptions, e.g:
- No 3 colinear points.
- No two points with the same x coordinate.
- etc.
Later, we consider the general case:
- How should the algorithm react to degenerate cases?
- Will the correctness be preserved?
- Will the runtime remain the same?

A reduction from sorting to convex hull is:
- Given n real values $x_i$, generate n 2D points on the graph of a convex function, e.g. $(x_i, x_i^2)$
- Compute the (ordered) convex hull of the points.
- The order of the convex hull points is the numerical order of the $x_i$.
- So $CH = \Omega(n \log n)$