Geometric Computing: Introduction

Piotr Indyk
Welcome to 6.850!

- Overview and goals
- Course Information
- Closest Pair
Geometric Computing

- Geometric computation occurs everywhere:
  - Robotics: motion planning, map construction and localization
  - Geographic Information Systems (GIS): range search, nearest neighbor
  - Computer graphics: visibility tests for rendering
  - Computer vision: pattern matching
  - Computational drug design: spatial indexing
Computational Geometry

• Started in mid 70’s
• Focused on design and analysis of algorithms for geometric problems
• Many problems well-solved, e.g., Voronoi diagrams, convex hulls
• Many other problems remain open
Course Goals

• Introduction to Computational Geometry
  – Well-established results and techniques
  – New directions
Syllabus

• Part I - Classic CG:
  – Closest pair
  – Segment intersection
  – LP in low dimensions
  – Polygon triangulation
  – Range searching
  – Point location
  – Arrangements and duality
  – Voronoi diagrams
  – Delaunay triangulations
  – Binary space partitions
  – Motion planning and Minkowski sum

Use “Computational Geometry: Algorithms and Applications” by de Berg, van Kreveld, Overmars, Schwarzkopf.
Syllabus ctd.

• Part II - New directions:
  – Closest pair in low dimensions
  – Approximate nearest neighbor in low dimensions
  – Approximate nearest neighbor in high dimensions
  – Low-distortion embeddings
  – Geometric algorithms for external memory
  – Geometric algorithms for streaming data
  – Pattern matching
  – Combinatorial geometry
  – Geometric optimization
  – Conclusions
Course Information

• 3-0-9 H-level Graduate Credit
• Grading:
  – Project
  – 2 problem sets (see calendar):
  – In each PSet:
    • Core component (mandatory): 6.046-style
    • Two optional components:
      – More theoretical problems
      – Java programming assignments
  – Can collaborate, but solutions written separately
  – Midterm but no final 😊
• Prerequisites: understanding of algorithms and probability (6.046 level)
Questions ?
Closest Pair

- Given: a set of points $P = \{p_1 \ldots p_n\}$ in the plane, such that $p_i = (x_i, y_i)$
- Goal: find a pair $p_i \neq p_j$ that minimizes $||p_i - p_j||$
- Easy to do in $O(n^2)$ time
  - For all $p_i \neq p_j$, compute $||p_i - p_j||$ and choose the minimum
- We will aim for $O(n \log n)$ time

$||p-q|| = [(p_x-q_x)^2 + (p_y-q_y)^2]^{1/2}$
Divide and conquer

• Divide:
  – Compute the median of x-coordinates
  – Split the points into $P_L$ and $P_R$, each of size $n/2$
• Conquer: compute the closest pairs for $P_L$ and $P_R$
• Combine the results (the hard part)
Combine

• Let \( d = \min(d_1,d_2) \)
• Observe:
  – Need to check only pairs which cross the dividing line
  – Only interested in pairs within distance \(< d\)
• Suffices to look at points in the \(2d\)-width strip around the median line
Scanning the strip

- Sort all points in the strip by their y-coordinates, forming \( q_1 \ldots q_k, \ k \leq n \).
- Let \( y_i \) be the y-coordinate of \( q_i \).
- \( d_{\text{min}} = d \).
- For \( i=1 \) to \( k \)
  - \( j=i-1 \)
  - While \( y_i - y_j < d \)
    - If \( ||q_i - q_j|| < d_{\text{min}} \) then \( d_{\text{min}} = ||q_i - q_j|| \)
    - \( j:=j-1 \)
- Report \( d_{\text{min}} \) (and the corresponding pair)
Analysis

- Correctness: easy
- Running time is more involved
- Can we have many $q_i$’s that are within distance $d$ from $q_i$?
  - No
- Proof by packing argument
Theorem: there are at most 7 $q_j$'s such that $y_i - y_j \leq d$.

Proof:

- Each such $q_j$ must lie either in the left or in the right $d \times d$ square.
- Within each square, all points have distance distance $\geq d$ from others.
- We can pack at most 4 such points into one square, so we have 8 points total (incl. $q_i$).
Packing bound

- Split into 4 disjoint sub-squares
- Each square has diameter $< d$
- At most 1 point per sub-square
Running time

- **Divide:** $O(n)$
- **Combine:** $O(n \log n)$ because we sort by $y$
- However, we can:
  - Sort all points by $y$ at the beginning
  - Divide preserves the $y$-order of points
    Then combine takes only $O(n)$
- We get $T(n) = 2T(n/2) + O(n)$, so
  $T(n) = O(n \log n)$