CS472 Module 3 Part B - Brute Force Computational Geometry

Athens State University

2013-11-18 Mon

Outline

Contents

1 Overview

What is computational geometry

- Study of algorithms that can be stated in terms of geometry
- Two types
  - *Combinatorial computational geometry*
    - Study of algorithms for solving problems stated in terms of basic geometrical objects: points, line segments, polygons,...
  - *Numerical computational geometry*
    - Problems expressed in terms of curves and surfaces
    - Related to computer-aided design (CAD)

Relationship to algorithm analysis

- Large data sets with ten to hundreds of millions of points
- Note the difference in performance between $O(n^2)$ and $O(n * ln(n))$ algorithms

Consider: computational geometry problems of real world interest will have more that $1,000,000$ data points. So, compare the difference between

$$1,000,000^2 = (1 * 10^6)^2 = (1 * 10^{12})$$

operations for an $O(n^2)$ algorithm compared to

$$(10^6) * log(10^6) = 6 * log(10) * 10^6 = 6 * 10^6 = 6,000,000$$

operations for a $O(n * log(n))$ algorithm.

Applications

- Robotics: Motion planning and visibility problems
- Geographic Information Systems (GIS): route planning
- Circuit Design: IC geometry design and verification
- Computer-aided engineering: mesh generation
- Computer vision: 3D reconstruction

2 Closest-Pair Problem

Closest-Pair Problem

Suppose that we want to find the two closest points in a set of n points in a plane

- Points can be location of physical objects
- Or database records upon which we want to perform cluster analysis

Closest-Pair Problem: Definition of distance

- What do we mean by "closest point"?
- For numeric data, use the concept of Euclidean distance

$$d(p_1, p_2) = \sqrt{(}\left((x_2 - x_1)^2 + (y_2 - y_1)^2\right))$$

- For example, in communications we can define the *Hamming distance* between two strings of equal length as being number of positions in which the strings differ
- Note that we aren't limited to just two dimensions as well
- We will use a function *dmetric()* in our algorithms
  - For numeric points in the plane, assume this is the Euclidean distance

Closest-Pair Problem: Brute-Force Method

Compute the distance between every pair of distinct points and return the indexes of the points for which the distance is smallest

Closest-Pair Problem: Algorithm

Algorithm 1: Brute-force method for finding closest pair of points

```
Input: A list P of n points, n must be greater than 2
Output: Indexes of the closest pair of points
dmin ← ∞;
for i ← 1 .. (n-1) do
    for j ← (i+1) .. n do
        d = dmetric(P[i], P[j]);
        if d < dmin then
            dmin ← d;
            index1 ← i;
            index2 ← j;
return index1, index2;
```

Closest-Pair Problem: Analysis

- The basic operation in our algorithm is the function *dmetric()*
- This can be a very complex operation
- Note that with Euclidean distance, we have to compute a square root
  - But do we? Suppose we only compute the squares?
  - Mathematically, the square root function is strictly increasing
- So, we now have squaring a number as our basic operation

Closest-Pair Problem: Analysis

Note now how many times we have to execute our basic operation:

$$\begin{aligned} C(n) &= \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 2 \\ &= 2 \sum_{i=1}^{n-1} (n-i) \\ &= 2[(n-1) + (n-2) + \ldots + 1] \\ &= (n-1)n \\ &\in \Theta(n^2) \end{aligned}$$

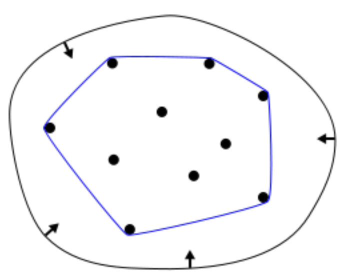
3 Convex-Hull Problem

Convex-Hull Problem: Definition

A set of points (finite or infinite) in the plane is called *convex* if for any two points p and q in the set, the entire line segment with endpoints of p and q is also in the set.

The *convex hull* of a set S of points is the smallest convex set containing S. The "smallest" requirement says that the convex hull of S must be a subset of any convex set containing S.

Convex-Hull Problem: The elastic-band analogy



In the plane, image stretching a rubber ban around the points in S and then releasing it. The convex hull of S is the set enclosed by rubber band when it becomes taut.

Convex-Hull Problem: Applications

Finding convex hulls is used in

- pattern recognition
- image processing
- statistical analysis
- mapping applications
- static analysis of behavior of computer programs

Convex-Hull Problem: Brute force algorithm

- Draw a straight line between two points P_i and P_j in S.
- If there are points in S on both sides of the line
  - Then the line drawn between P_i and P_j is not in the convex hull
- Suppose all the points in S are on one side of the line (or on the line)
  - This implies that the line is on the boundary of the convex hull

Convex-Hull Problem: Brute force algorithm

- If $P_i = (x_i, y_i)$ and $P_j = (x_j, y_j)$, then we can define a nonzero solution for the straight line between the points as

$$\begin{aligned} ax_i + by_i &= c \\ ax_y + by_j &= c \end{aligned}$$

- We can solve this system of equations for a, b, and c

$$\begin{aligned} a &= y_j - y_i \\ b &= x_i - x_j \\ c &= x_i y_j - y_i x_j \end{aligned}$$

- So, a line segment from P_i to P_j is on the convex hull if either $ax + by \geq c$ or $ax + by \leq C$ is true for all points in S
- We can simplify this point by noting that we need only to compute the sign of $ax + by - c$ for all points

Convex-Hull Problem: Analysis

- Note that we will have $\dfrac{n(n-1)}{2}$ possible distinct points.
- We will need to find the sign of $ax + by - c$ for each of the other $n - 2$ points in S
- Thus our algorithm is $O(n^3)$

4 Key Points

We will revisit these problems

- Much better algorithms exist for both closest-pair and convex-hull
- However, the more efficient algorithms are very sensitive to characteristics of the data sets
- Sometimes better to use the simpler implementation

Key Points

- What is computational geometry?
- Design and analysis of the closest-pair algorithm
- Design and analysis of the convex-hull problem
- Brute-force vs. other methods