14. Computational Geometry Introduction

CPSC 535

Kevin A. Wortman

CALIFORNIA STATE UNIVERSITY FULLERTON

Big Idea: Problem Duality

problem duality: when the input/output mathematical definition of a problem can be interpreted by humans in two (or more) very different ways

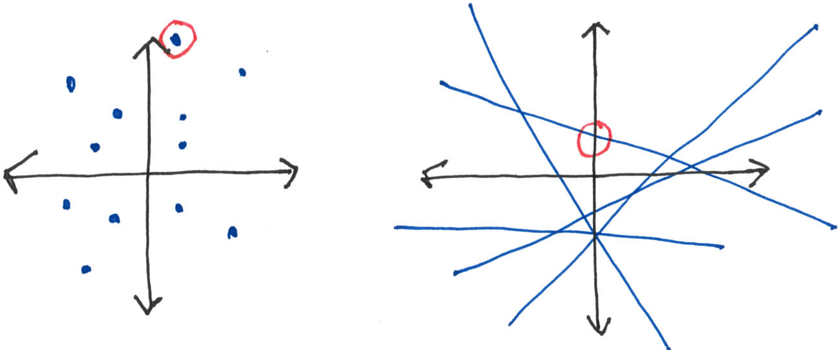
- one algorithm can solve multiple problems with different "stories"
- algorithms, computers, don't actually care what data values mean
- turns out max-flow and min-cut are two different stories for the same problem
- max-flow and min-cut are the *dual of each other*

Duality Example

maximum y coordinate
input: a set of (x, y) points $S = \{(x, y) \mid x, y \in \mathbb{R}\}$
output: the greatest y-coordinate in S

highest y-intercept point problem
input: a set of $y = mx + b$ lines $L = \{(m, b) \mid m, b \in \mathbb{R}\}$
output: the greatest y-intercept b in L

Geometry Sketch



C++ functions for these would be declared like:

```
double maximum_y_coord(vector<pair<double, double>>& points);
double highest_y_intercept(vector<pair<double, double>>& lines);
```

As far as the computer is concerned, these are interchangeable!

Only the human story differs. The **maximum *y* coordinate** and **highest *y*-intercept point problem** problems are the dual of each other.

Big Idea: Output Sensitive Algorithm

- **input sensitive**: time efficiency is a function of the input e.g. size n, # edges m
- **output sensitive**: efficiency is also a function of the *output* size e.g. # items returned
- most relevant when the size of the output could be the bottleneck

Computational Geometry

computational X: interdisciplinary study of computer science with X

(computational finance, epidemiology, physics, finance, etc.)

computational geometry (CG): algorithms, data structures, asymptotic analysis, of geometric objects: points, lines, circles, triangle meshes, etc.

Computational Geometry Applications

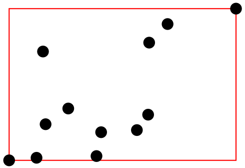
Applications of CG:

- 3D computer graphics
- graphical user interfaces (GUIs)
- geographic information systems (GIS), geographic databases
- scene reconstruction, self-driving cars (e.g. LIDAR)
- business operations research (e.g. linear programming, aircraft control)
- manufacturing (e.g. feasibility of assembly, castings)

Putting the Geo in CG

Some general algorithms can actually solve geometric problems efficiently, without any awareness of geometry.

bounding box problem
input: set of 2D points $P = \{p_1, p_2, \ldots, p_n\}$
output: points $tl = (x_l, y_t)$ and $rb = (x_r, y_b)$ such that the rectangle with top-left corner tl and bottom-right corner rb contains P



Naïve, optimal algorithm:
$x_l = \min x, y_t = \max y, x_r = \max x, y_b = \min y$; $\Theta(n)$ time

Computational geometers are more interested when geometric properties matter.

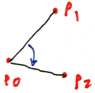
Line Segment Predicates

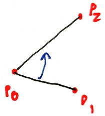
We can use arithmetic to answer any of the following predicates (questions) about points p_0, p_1, p_2, p_3 in $\Theta(1)$ time:

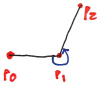
1. Is line segment $\overline{p_0 p_1}$ clockwise from $\overline{p_0 p_2}$ around the common endpoint p_0?
2. If we follow $\overline{p_0 p_1}$ and then $\overline{p_1 p_2}$, do we turn right or left?
3. Do line segments $\overline{p_0 p_1}$ and $\overline{p_2 p_3}$ intersect?

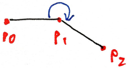
$\Longrightarrow$ We may use any of these in pseudocode.

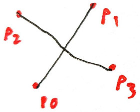
Line Segment Predicates

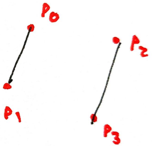








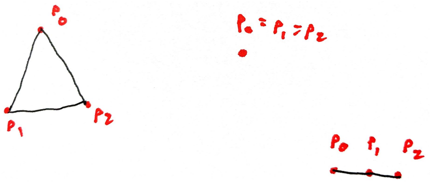




Degeneracy and Non-Degeneracy Assumptions

degenerate object: has the proper shape/type, but the values are a special case that betrays the spirit of the definition

Example: triangle ≡ three points (p_1, p_2, p_3)
degenerate triangle: $p_1 = p_2 = p_3$, or all points colinear



Non-Degeneracy Assumptions

non-degeneracy assumption:

- constraint that input to a CG algorithm is not degenerate in specific ways
- simplifies algorithm design
- assume that in practice, some combination of
  - degeneracies do not occur
  - input can be preprocessed to remove degeneracies
  - implementer can modify algorithm to handle degeneracies

Sweep Algorithms

A pattern in CG algorithms:

- *line sweep:* envision a line "sweeping" through the input
- e.g. a vertical line sweeping left-to-right
- helps us visualize a 2D situation as a 1D situation that changes over time
- like duality, doesn't actually change the problem, but might help us problem-solve
- generalizes to higher dimensions e.g. plane sweep in 3D, hyperplane sweep in any dimension

Sweep Algorithms