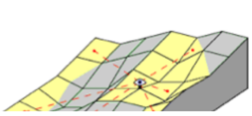
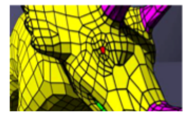
Strategies for computational problems

- Brute force …
- Incremental – solve a small problem and add to it
- Divide and conquer – break down problem into small parts, then merge results
- Sweep-line – common for geometric problems, where inputs can be naturally ordered in the plane (e.g., along x or y axis)

How operations on geometries work

- Principles of computational geometry
- A branch of computer science / applied mathematics concerned with efficient algorithms operating on geometries
- Examples:
  - Computer vision, robotics
  - Gaming (3D etc)
  - Logistics
  - Spatial analysis





Length and area

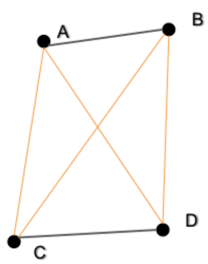
- *Length* of a line segment can be computed as the distance points between successive pairs of points

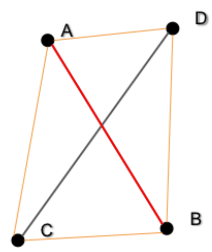
$$|pq| = \sqrt{(x_q - x_p)^2 + (y_q - y_p)^2}$$

- *Area* of a polygon: Let P be a simple polygon (no boundary self-intersections) with vertex vectors $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ where $(x_1, y_1) = (x_n, y_n)$ (i.e., the polygon is closed and has the same start and end vertices). Then the area is:

$$\textbf{area}(P) = \frac{1}{2} \sum_{i=1}^{n-1} x_i y_{i+1} - x_{i+1} y_i$$

Line segment intersection



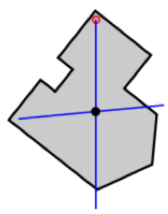


$$\textbf{side}(a, b, c) \neq \textbf{side}(a, b, d)$$
$$\textbf{side}(c, d, a) \neq \textbf{side}(c, d, b)$$

If these inequalities hold, the lines intersect

Centroid of a polygon

- The *centroid* of a polygon (or *center of gravity*) of a (simple) polygonal object ($P =< (x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n) >$ where $(x_1, y_1) = (x_n, y_n)$) is the point at which it would balance if it were cut out of a sheet of material of uniform density:

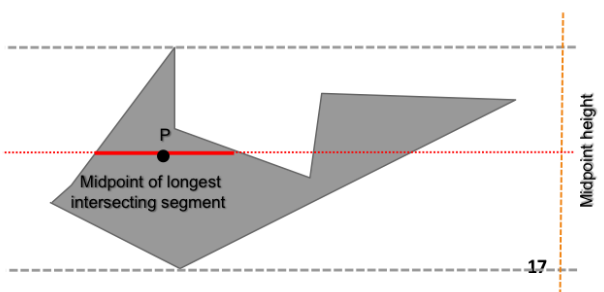


$$\textbf{centroid}_x(P) = \frac{1}{6.\textbf{area}(P)} \sum_{i=1}^{n-1} (x_i + x_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

$$\textbf{centroid}_y(P) = \frac{1}{6.\textbf{area}(P)} \sum_{i=1}^{n-1} (y_i + y_{i+1})(x_i y_{i+1} - x_{i+1} y_i)$$

- ST_Centroid – but no guarantee the point is inside the polygon!

ST_PointOnSurface (Postgis)



Point-in-polygon

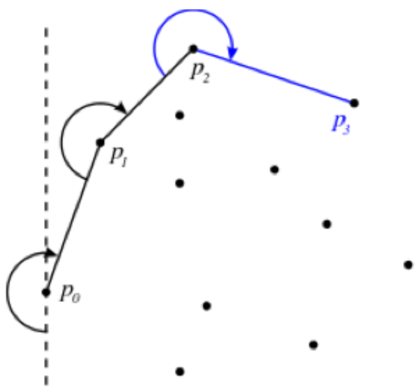
- Semi-line algorithm: check the number of intersections of any semi-line from the tested point:
  - If even: NOT in polygon
  - If odd: in polygon
- Winding line algorithm: Sums the angles from the point to all the vertices of the polygon:
  - If 360 deg: in polygon
  - If <360 deg: outside of polygon

Convex hull

- The convex hull of a set of points S is the smallest convex polygon that encloses S
  - Recall: A convex polygon has every point inter-visible

Covex hull – Quickhull

Covex hull – Jarvis march (gift wrap)



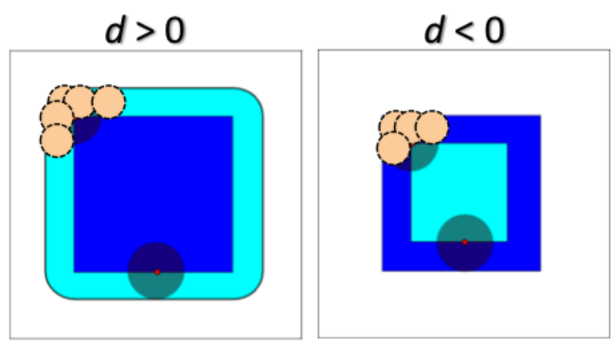
Convex hull – Graham's scan

- **O(n log n)**
  1. Find pt with lowest y coord (or x): P_0
  2. Sort all points in S in increasing order of angle between x axis and P_0
  3. Add the next point to the stack
  4. Add the next point to the stack, and check whether it's to the right or left of the previous segment
  5. If right, keep going
  6. If left, remove the point, add next, check again

Buffer

- **A geometry defined by a set of points *S* with a maximum distance *d* from all points of the source geometry *g*.**
- **Equivalent to dilation (positive buffer) or erosion (negative buffer) in image processing**

- **Buffering by a positive or negative distance**

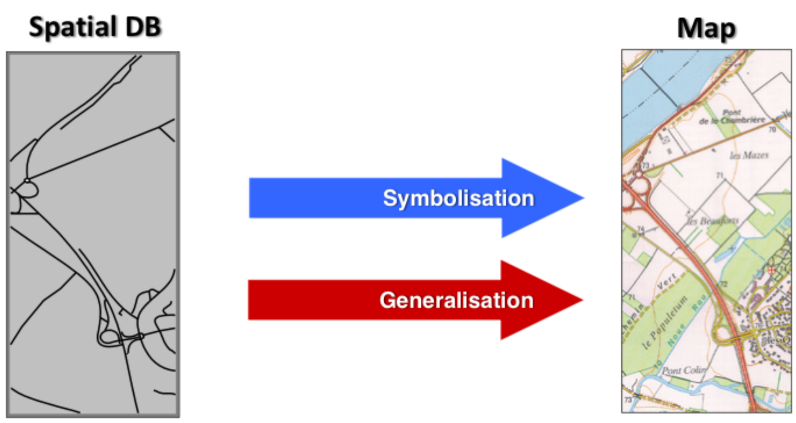


Applications

Generalization

- **Generalization: the process of selecting and processing information for display on a map appropriate for the scale and purpose of the map, and to the medium of presentation;**

From DB to display

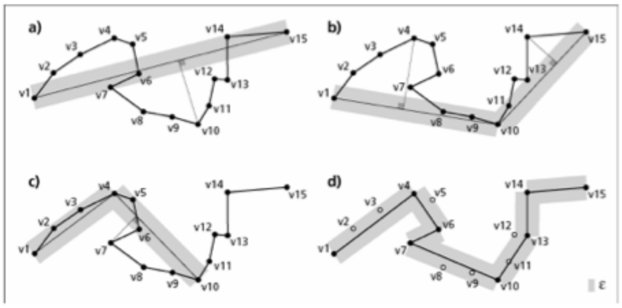


Line simplification

- *Simplify the shape of a cartographic line by removal of 'unnecessary' vertices*". (McMaster and Shea, 1992)… which are these?

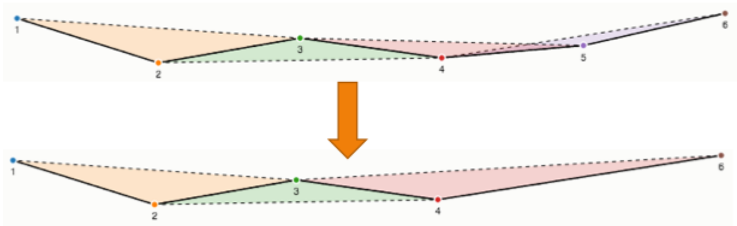
Douglas Peuker algorithm

- *Vertices that are outside a tolerance band are considered important and are retained (beware, does not preserve topology!)*



Visvalingham-Whyatt algorithm

- https://bost.ocks.org/mike/simplify/
  - Uses areal thresholds



Computational complexity

Notation: O() (Big-Oh)

- **O(log n): binary search –** `time required grows slower than problem set`
- **O(n): max** `(l1) – single pass`
- **O(nlogn): distances of all points to all in a matrix –** `symmetrical relationships do not need to be checked again`

Approximate values of common functions

n	$\log_e n$	$\sqrt{n}$	n^2	2^n
1	0.0	1.0	1	2.0
25	3.1	5.0	625	3.3×10^{7}
50	3.9	7.1	2500	1.1×10^{15}
75	4.3	8.6	5625	3.8×10^{22}
100	4.6	10.0	10000	1.3×10^{30}

Common time complexity orders

$O(1)$	Constant time	Very fast
$O(\log n)$	Logarithmic time	Fast
$O(n)$	Linear time	Moderate
$O(n \log n)$	Sub-linear time	Moderate
$O(n^k)$	Polynomial time	Slow
$O(k^n)$	Exponential time	Intractable

Computational complexity

- Worst case complexity: usually of most interest in algorithm analysis;
- Average case complexity: more representative than worst case, but harder to calculate due to stochastics;
- Optimality: is this algorithm the best possible?
- Underlying data structures may be part of the complexity of an algorithm (see indexes lectures)
- Complexity analysis may include time (steps) and space (required memory size) complexity