Week 8 Mon 11/14/16 Spatial Analysis (cont.)

When you think about solving problems, ponder these two modes of spatial analysis:

1. Exploratory Analysis, vs.
2. Confirmatory Analysis
3. Exploratory Analysis: Interacting with the data, analyzing it and visualizing it, searching for possible patterns, for statements you can make about the data, for relationships in the data
4. Confirmatory Analysis: You have an idea about phenomena and their relationships in the world and you’re seeking to confirm (or often, make sure you can’t easily reject) that relationship by analyzing data

These are not entirely incompatible, and you’ll often be doing degrees of both

How does a spatial operation happen?

Algorithms: Literal, step-by=step procedures that allow a computer to solve a problem

Someone write them down in code

But they should be understood in human language, too

You may ask:

What does it (the algorithm) seek to do?

What sort of data does it assume you will give it?

What data model is used to represent data?

How does performance ‘scale up’?

Is the algorithm only feasible when the amount of data is small?

Week 8 Wed 11/16/16 Spatial Analysis (cont.)

Spatial operation happens with *algorithms*

Computer science uses ‘Big-O Notation’ to describe how fast or complex an algorithm is

O(*n*): “Has a big-Oh of n”, “Is of order of *n*”

If an algorithm’s solution is O(*n*) for a problem involving *n* points, that means that the solution time tends to increase linearly with the number of points, *n*

If the number of input points doubles, the solution time doubles

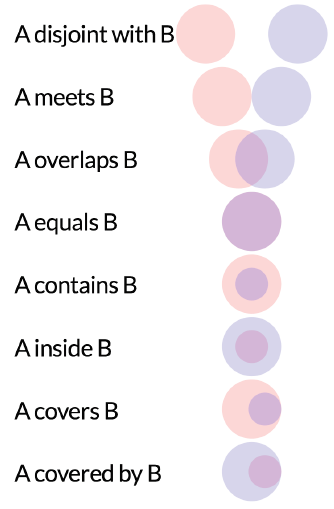
O(n^2) means the solution time goes up with the square of the number of points, *n*

If the number of input points doubles, the solution time quadruples

O(1) means that the solution time is roughly constant and is independent of the number of points, *n*

If the number of input points doubles, the solution time stays the same

Considering both attribute and spatial relationships in analyzing data:



Finding algorithms to solve a problem: How can you tell if a point is inside a polygon?

Something your GIS calculates often

Questions that this might help answer:

Which state is this point in?

Am I on land or on water?

Your GIS needs to not only know how to answer such a question, but how to *know how to do it fast*

**Can you come up with procedures you can write down that a computer might be able to use to figure out if a point is in a polygon?**

Angle-Summation Algorithm: O(*avg-number-vertices-per-poly*)

“The worst algorithm in the world for testing points is the angle summation method. It’s simple to describe: sum the signed angles formed at the point by each edge’s endpoints. If the sum is near zero, the point is outside; if not, it’s inside. The winding number can be computed by finding the nearest multiple of 360 degrees. The problem with this scheme is that it involves a square root, arc-cosine, division, dot and cross product for each edge tested.”

‘Ray-Casting’ or ‘Even-Odd Rule’ Algorithm: O(*avg-number-vertices-per-poly*)

The number of intersections for a ray passing from the exterior of the polygon to any point; if odd, it shows that the point lies inside the polygon. If it is even, the point lies outside the polygon; this test also works in three dimensions.

When you have many polygons and many points

Think about the case when you have a long list of longitude and latitude points and you want to know which county each belongs to. Do you really want to check each county polygon for whether each point is within it?

This “naïve solution” of endlessly repeating a point-in-polygon method (like the ray-casting algorithm) is *too expensive*: O(num-of-polygons *x* avg-vertices-per-polygon *x* num-of-points-to-check)

**Clever algorithms are often interconnected with clever ‘data structures’: ways of storing the data ahead of time that allow for easier computations.**

One simplistic possibility: Imagine making a raster where each cell’s value is the ID of the polygon that contains that cell. Building that raster is ‘costly’, but then you can tell which polygon/county a new coordinate is in within O(*avg-num-vertices-per-poly*) time

**This is the power of algorithms and data structures: they make things *feasible*, and thus, in a practical sense, *possible***

We wouldn’t have GIS without spatial databases and their algorithms

We wouldn’t have Google without an algorithm called ‘Page Rank’

What algorithm can you use to figure out how far a point is from a big city?

This is not just a point-in-polygon style calculation

You need to calculate the shortest ‘distance’ measured in time. There are many paths form a point to all potential cities. And the ‘shortest’ path may not be a straight line

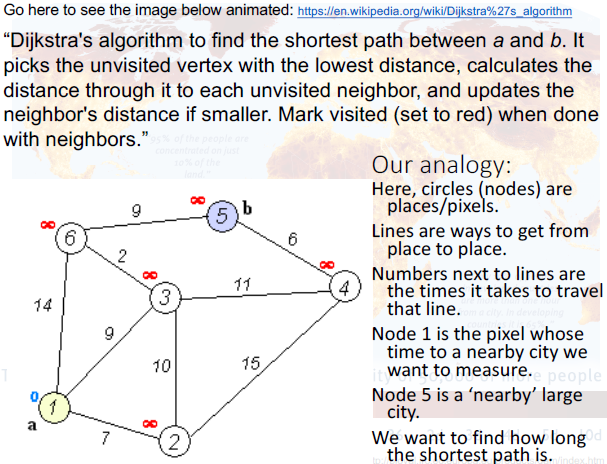
You need 3 layers:

Point layer

City layer

‘Cost’ network showing how much time it takes to move from each place to each neighboring place. Speed varies by terrain and infrastructure

You need a ‘shortest path’ algorithm. One classic one you learn about early in computation is ‘Dijkstra’s algorithm’. It’s not perfect for this application, but it’s close enough to be illustrative You then need to apply it repeatedly for each point on the earth so you can map the time-to-nearest-city for each.



Week 8 Fri 11/18/16 Spatial Analysis: Overlay Operations

A few useful (vector data) spatial analysis techniques:

Overlay

Union / Intersection

Clip / Merge / Append

Buffering

Relies on the simpler spatial-analytical operation: ‘calculate distance’

Dissolve

Overlays

An overlay operation takes two or more layers as inputs

It puts them together to produce an output layer

This is very general, so there are many possible variations on this theme

There are **many use cases**

