

# Maps in Computers

## Storing and Organizing Geographic Information

### Geographic Data Organization

Geographic data come in three categories. The first category consists of the basic geometric entities we use to represent geographic features. The second category incorporates the ways in which we organize the basic geometric entities to represent more complex spatial forms. The last category includes the organizations of the attribute data we associate with the geographic features.

#### *Geographic Data Types*

Geographic features are identified at a specific location (nodal), along a path (linear), within a bounded region (areal), over a surface (surficial), or within a volume (volumetric).

#### Basic Geographic Data Types

Type	Dimensionality	Examples
Nodal	1	Cities, wells, control points
Linear	2	Roads, flight routes, political boundaries
Areal	2	Counties, Census Tracts, soil zones
Surficial	3	Topography, precipitation, population density
Volumetric	3	Pollution plume, aquifer, ore body

#### *Organizing Basic Geographic Data Types*

A geographic database represents sets of geographic features. These can be represented most simply as collections of the basic geographic data types. However, we are often interested in using our geographic information as efficiently (or effectively) as possible. This means thinking about how we plan to collect and use the data in order to develop a more effective data organization.

One of the more common objectives for any database is to ask for information about the features represented in the database. This involves obtaining information about the geographic as well as attribute properties of features represented in a geographic information system. The kinds of things we might ask include “where the restaurant”, “how do we get to the restaurant”, or “who owns the parcel next to mine?” These questions involve obtaining basic information about location, but can also involve using more complex spatial relationships, such as *next to*.

Geographic Question	Spatial Relation
Where is the restaurant?	Location
What are the names of all the Chinese restaurants in San Francisco?	Containment (only those restaurants <i>contained</i> in San Francisco)
Where are the Chinese restaurants within walking distance of Union Square?	Proximity
What is the shortest path from Union Square to the restaurant?	Connection (find the shortest <i>connected</i> path; i.e., dead-ends are not helpful).
Which gift shops are located next to a Chinese restaurant?	Adjacency

The challenge in organizing geographic information is to enable us to answer these kinds of questions easily and quickly. Doing so requires that we think about the kinds of information we need to have available. We have to decide whether to calculate the information we need from the information in the database (e.g., the distance between two places), or to calculate in advance and store the results in the database. This is similar to using mileage charts on road maps. You can calculate (measure) the road distances between any two places. However, this is inconvenient (or even annoying) for the more commonly asked questions, such as “what is the road distance from San Francisco to Los Angeles?” We place the mileage charts on the maps so that you can easily obtain distances between commonly traveled places. Similarly, we can construct geographic databases in which some of the information is predefined, so that we don’t have to measure or calculate it when we want to use it.

Ironically, one of the more difficult operations to perform by computer is to establish spatial relationships. Tasks that you or I can accomplish with relative ease, such as determining which parcel is next to ours, are in fact difficult to do by computer. This means that we have to consider how we are going to organize our databases in order to use the computer as a tool for analyzing the spatial properties of geographic features.

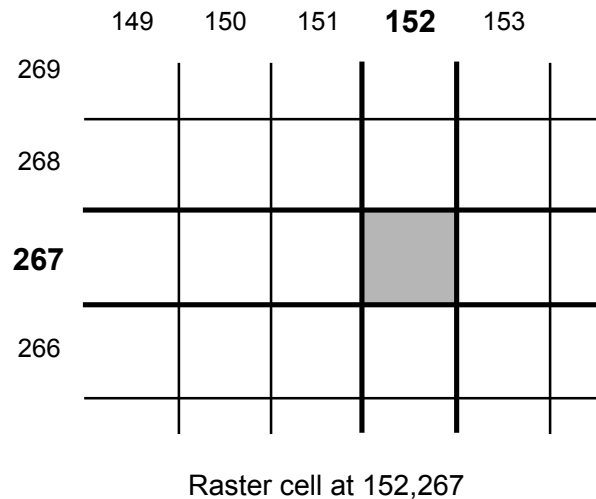
### Cartographic Data Structures

We organize our geographic information into one or more *cartographic data structures*. A data structure is a way of organizing information in a computer. A cartographic data structure is a way to organize geographic (or cartographic) information. This involves primarily arranging the data so that we can perform a variety of spatial operations on that data.

The two primary cartographic data structures are the *raster* and *vector* data organizations. The raster structure arranges data into a regular array of data elements. These data elements are typically square cells in rows and columns. The vector structure is based on representing the geographic feature (nodal, linear, areal) with a corresponding geometric entity (point, line, area). Raster and vector structures provide us with different ways of treating geographic information, but neither is perfect, so we have to evaluate which type of data organization best fits our analytical problem.

*Raster Data Structures*

The raster data structure's row and column organization makes accessing the information very easy, both for us and for the computer. Data collection is very easy, as all data elements are the same size and shape. Data retrieval is also easy, because the location of any raster element is defined from its row and column position. Measuring areas simply involves counting raster elements and multiplying by the area of the element. Finally, identification of adjacent cells may be obtained simply from the row and column positions.

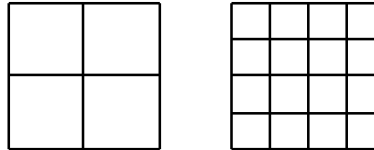


The raster structure is best used with data that forms a continuous geographic coverage, and for which we are not concerned with trying to show the exact configuration of a feature's boundaries. This is because the raster elements are the finest level of data we have. The edges of a "feature" are shown as a collection of these raster elements, and so any boundary that is not parallel to a row or column will appear as stair-stepped (this phenomenon is known by the highly technical term "the jaggies"). Therefore, raster data structures appear crude when viewed at larger scales, where the jaggies start to appear.

Raster data structures are generally not appropriate for nodal or linear data. Nodal data would be generalized to the raster element containing the point, so the exact location would be lost, with little corresponding benefit to other operations we may wish to perform. Linear data suffers from the jaggies, just like the boundaries of areas, so that distance measurements would be more difficult to make accurately. Finally, raster data structures are not easily used when the map projection has to change frequently.

We control the raster data structure by adjusting raster element size, orientation, and origin. The major controlling factor is the element size. Large elements generalize more, and therefore, are less accurate representations. However, small element sizes lead to significant increases in the number of raster

elements needed to represent a given study area. For example, if we have a study area that we can cover with 1 million 1x1 km raster elements, if we reduce the raster element to 500x500 m (0.5 x 0.5 km), then we would need 4 million raster elements to cover the same area. This is because each raster element now covers only 25% of the area of the original raster elements ( $0.5 \text{ km} \times 0.5 \text{ km} = 0.25 \text{ km}^2$  instead of  $1 \text{ km}^2$ ). We still face some limits as to the size of a database, even though data storage is much less expensive and computers are faster than they've ever been. So we still have to give some thought as to the level of detail required as we select an appropriate raster element size.



Decreasing cell size to half  
quadruples the number of cells.

### *Vector Data Structures*

The vector cartographic data structure stores information primarily as a set of coordinates. The basic information structure is the X-Y coordinate pair, indicating a specific location in a Cartesian coordinate system. All features are represented as one or more locations. For example, a benchmark is represented as a single location, a road may be represented as a series of locations marking the major changes in the road's direction, and a county may be represented as a series of locations delimiting its boundary.

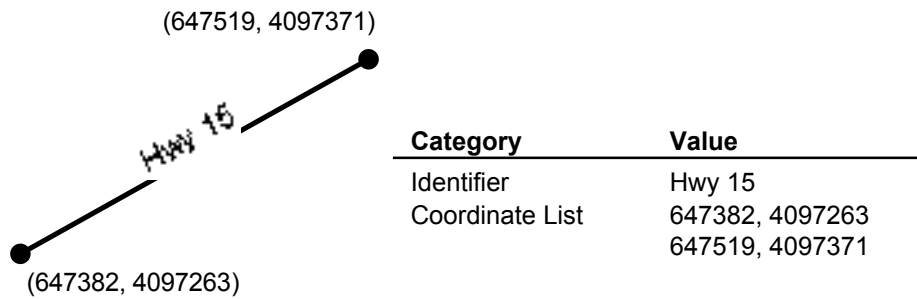
<b>ID</b>	<b>X</b>	<b>Y</b>
Tree14	647980	4092780

Locations may be designated in any coordinate system: latitude and longitude (geographic coordinates), UTM, or state plane. We can also transform a location from one coordinate system to another through the projection process. Selection of the base coordinate system for a geographic database depends on the primary applications for the database. Geographic coordinates are most appropriate if the database covers large geographic areas or if the database will be converted into a variety of different projections. Plane coordinates are appropriate for limited geographic regions for which distances and area calculations are important.

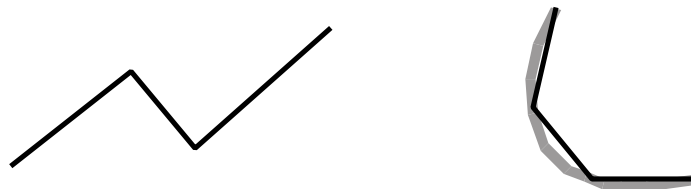
Nodal features are represented by a single *location*. Attribute information (e.g., soil type for a soil sample) is then associated with that location. We can then assign a symbol to the location, based on the attribute to create maps showing the geographic variation of that attribute.

ID	X	Y	Soil Type
Tree14	647980	4092780	Clay

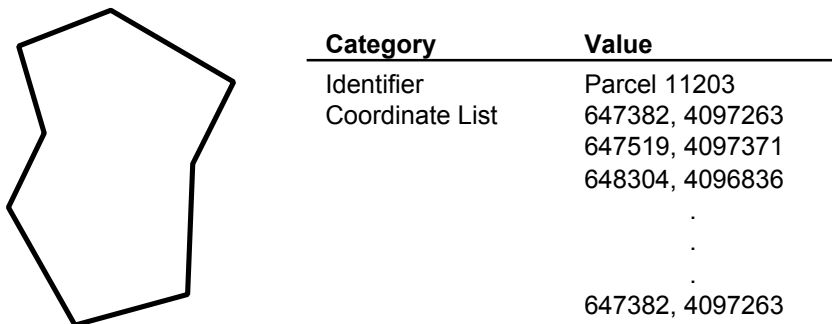
Linear features are represented as one or more line segments. A *segment* is a single straight line between two locations.



More complex linear features can be represented as a series of segments chained together. Even curved features may be approximated by a series of segments.

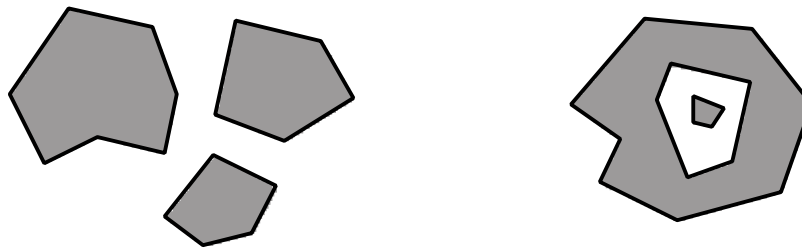


An areal feature, such as a county, may be represented as a closed set of segments, or a *polygon*, corresponding to its boundary. The closed boundary encloses the region's territory.

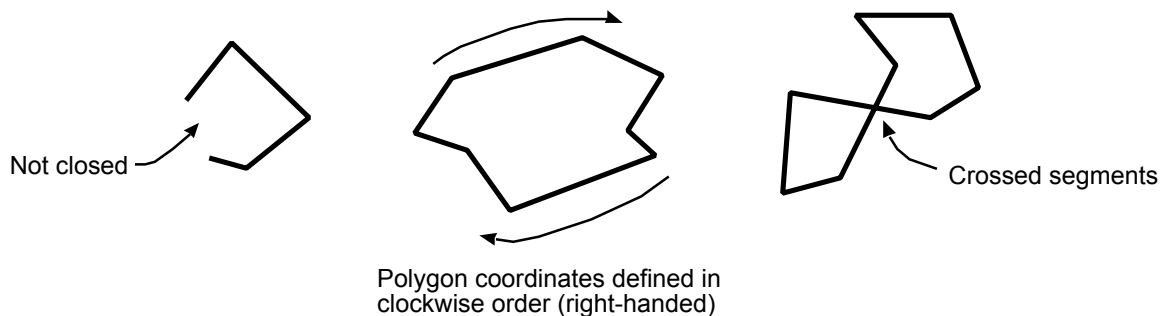


The polygon is commonly used for geographic information, since so much data area assigned to areas. Polygons can be grouped together to provide contiguous coverages over a larger region (such as the counties making up a state). Polygons can also be grouped together to form a collection of

discontinuous polygons, such as an island group. Finally, polygons may include holes (such as a lake in a national park), and holes within holes (such as an island in a lake in a national park).



Polygons must be defined according to specific rules in order to represent complex geographic relationships in a geographic database effectively. First, the polygon must be closed. This means that the last point and first point of the boundary must be the same. All polygons in a database must have a *consistent specification*; that is, all points making up the polygon boundary are in order around the boundary. The most common specification is a *clockwise*, or *right-handed*, definition, where following the boundary in order of definition results in moving around the boundary in a clockwise direction and the interior of the polygon is always to the right of the boundary. Finally, the boundary segments cannot cross. These characteristics result in a *well-defined polygon*.



### *Topological Structures*

The basic vector data structures provide the means for describing and storing a variety of geographic data. However, we often wish to access the information in more complex ways, and the basic structures must be extended in order to make these operations more effective. For example, we might create a street database if to help find the shortest route to a restaurant. The database would consist of line segments or chains of line segments representing the streets, and we would then need to select the streets to follow in order get to the restaurant, beginning with our starting street. However, how do we know which streets are connected to our street, so that we can select one to follow? If we looked at a map, we can immediately see which streets are connected to one another. However, if we ask our GIS to follow the streets, we won't be doing the connecting, the computer will. And computers can be very limited when it comes to processing geographic or spatial relationships.

This is because computers generally work on one thing at a time. So for example, when following a street, the GIS software will get one street segment, then find the segments connected to that one, select one to follow, get the next connected segments, and so forth. One major limitation of the basic cartographic data structures and of the architecture of modern computers is that each basic entity is completely independent of other entities. That is, each point, line, and polygon is separate from others, and when we are dealing with any particular entity, we do not have information about others. So we have to search through the database to find connecting streets.

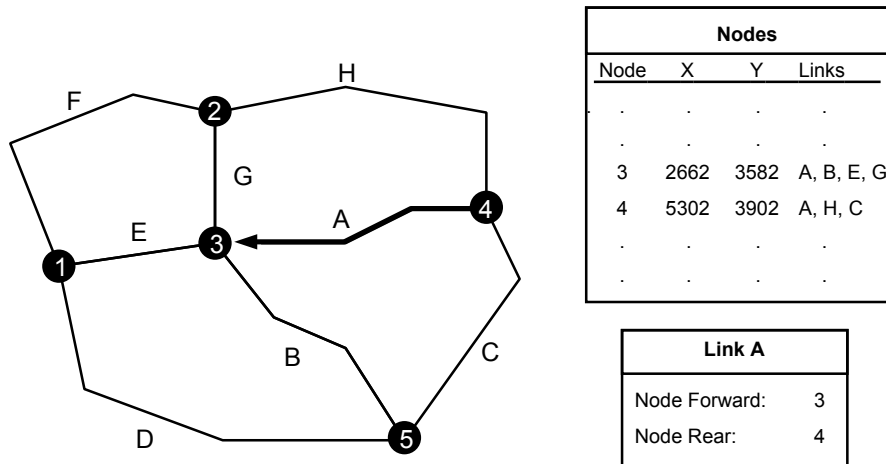
Another way to visualize this computer tunnel vision is to imagine driving in fog. Now think about driving down the street. You'd probably want to know if another car was approaching from the left or right when you are about to cross an intersection, but you cannot see beyond what is immediately around your car. If the fog is thick enough, you might not even know that you are crossing an intersection. The same situation applies when we are working with a geographic database. The application software generally examines one feature at a time. When it is dealing with a line segment representing a portion of a street, it deals only with that segment. In a sense, computer mapping systems cannot "see" the entire map, whereas when you look at a map, you can easily see which streets are connected to our street. This characteristic of computer systems and basic cartographic data structures limits our ability to perform more complex geographic operations.

One way to solve this for our restaurant trip is to use basic geometry to "find" all of the streets having common street intersections. We are able to do this because we know something about how streets in a street network are related to one another. That is, we know that streets are connected to other streets. This kind of relationship is called *topological* because it is based on the branch of mathematics called *topology*. Topology examines the relationships among geometric objects, such as lines, independent of their actual positions or even their specific shapes. For example, the segments of the street network are connected to each other. If we change the map projection, the apparent positions and shapes of the streets may change, but the way in which they are connected will not change. Topology, then, enables us to describe the relationships among the streets in the network in a way that supports our route-finding application.

Returning to our restaurant problem, we would look for all lines with one end point at the same location as either end of our starting street. We would then select one of the connecting lines and find the next connecting line. This would enable us to follow (or "trace") the street network to get to our destination. The problem with this approach is that it can be very time consuming if the street network is very large. Also, we would have to locate connecting streets each time we wanted to find the route to a new destination, so that we could face long waits while the geographic information system identifies our route.

An alternative to calculating the connecting streets is to define the relationships in advance and store information about which streets are connected to one another. In so doing, we create a data structure that includes topological relationships. We refer to such data structures as *topological data structures*,

because they make the topology explicit. So in this case, we would use the information about which streets connect with our starting street to limit our search. We don't have to examine every street in the network for connection to our street; we have that information already. This is like having the fog lift enough so that we know that we are at an intersection as we are driving. We would then know that we are at a place where other streets connect to our street. We can now spend our time deciding which connecting street to follow.



A topological network structure includes a table of nodes, as well as a table of links. Nodes are the end points of links, and are shared among several links (three or more). A link is defined by referring to the identifiers of its nodes, rather than by coordinates. We can determine which links share a node by comparing identifiers, rather than comparing coordinates. For example, links G, A, and C share node 4.

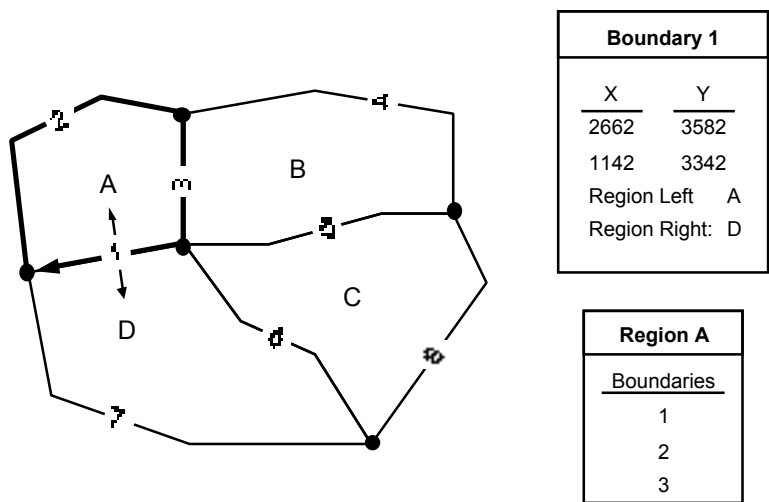
We can trace a path through the network by moving between the node and link tables. For example, if we start on link A, we can find all of the links connected to A by obtaining its nodes (e.g., node 3), then examining all of the links connected to that node (A, B, E, and G). We can then continue tracing the network by selecting one of the connecting links (B, E, or G), and finding the links connected to it.

Topological data structures also have some advantages for representing polygon coverages. One of the problems we face with polygon coverages is that many of the polygons share boundaries. This is a problem for simple polygon data structures for three reasons. First, each shared boundary is stored twice, once for each of the neighboring polygons. Second, calculations, such as intersection of a road with a set of polygons, are often doubled, once for each of the shared boundaries. Finally, when we are creating the database, we have to digitize each shared boundary twice. We can minimize or eliminate these problems by representing the polygon set as a topological structure.

The topological relationship of interest in this case is *adjacency*. We know that many of the polygons share common boundaries, or are adjacent to one another. We can eliminate the redundancy problem if

we restructure our view of the database to use the boundary section as the fundamental entity. We do this by first defining the individual boundary section as a line representing the portion of the boundary shared by two polygons. This would minimally be a line, and could be a complete polygon for a polygon contained completely within another polygon (e.g., a lake). The end points of the boundary section are those locations shared by three or more polygons.

We would then define any polygon as a collection of boundary sections. We would also identify and store references to the polygons to either side of each boundary section (this is the adjacency information). The result is a complete description of the polygons making up the coverage and the adjacencies between polygons, with the boundaries defined in the database only once.



Here we have a polygon defined as a set of boundaries. Region A is made up of boundaries 1, 2, and 3. Each boundary consists of a set of X and Y coordinates, and also includes information about which regions are to its left and right. We can easily determine A's neighbors by examining its boundaries. For example, finding A's neighbor across boundary 1 involves examining 1's left and right regions, which are A and D.

**Standard data formats**

Digital Line Graph (DLG) is a standard USGS format for vector data. DLGs can represent nodal, linear, and areal map features. Areal data are represented in topological form, with the boundaries of an area being defined by a series of line features. The line features include an indication of the areas on the left and right sides of the line. The format is used to convert and distribute standard USGS topographic maps in digital form. It is also used for specialized data sets, such as the national land use/land cover mapping program. The data are organized by topographic quadrangle, and are available for data derived from maps at 1:24,000 (7.5 minute series), 1:100,000, and 1:250,000. The standard form of the DLG provides coordinates in map inches. The optional format, used for most GIS

applications, provides coordinates in UTM. The most recent version of the DLG format has been created to support the Spatial Data Transfer Standard (see below).

Digital elevation models (DEM) is a standard USGS format for terrain data. Elevation data are organized into grids of individual elevation values. The easting and northing for the elevation points are determined by parameters specific to the scale of map from which the DEM was derived. The 1:24,000 DEM uses a 30-meter grid spacing in the UTM coordinate system. The 1:250,000 DEM uses a 3-arc second spacing. Most DEMs use a single spacing for the entire set of elevations in the file, although the format will support different spacings in different parts of the file.

Digital Orthophoto Quads (DOQ) are raster images derived from aerial photography. The photographic data have been rectified to provide correct planimetric locations for the features shown in the image. The image data are in a UTM grid with a 1 meter resolution. The files are broken into quarters, resulting in Digital Orthophoto Quarter Quads (DOQQs).

Digital Raster Graphics (DRG) are scanned versions of the standard USGS topographic maps. The images show all of the graphic information on the topographic map, but specific feature information cannot be easily extracted. The DRGs serve as a useful graphic representation, and can act as a background for other data.

TIGER (Topologically-Integrated Geographic Encoding and Referencing) is a standard format used by the Census Bureau to distribute census geography. These include the boundaries of census tracts, blocks, and block groups. The format is topologically-based, as the name indicates. The primary unit is boundary section between adjacent polygons.