UNIT 32 - SIMPLE ALGORITHMS I - INTERSECTION OF LINES

UNIT 32 - SIMPLE ALGORITHMS I - INTERSECTION OF LINES

Compiled with assistance from David H. Douglas, University of Ottawa and David M. Mark, State University of New York at Buffalo

UNIT 32 - SIMPLE ALGORITHMS I - INTERSECTION OF LINES

Compiled with assistance from David H. Douglas, University of Ottawa and David M. Mark, State University of New York at Buffalo

A. INTRODUCTION

- the intersection of two lines is a critical operation in GIS
    - is used in polygon overlay operations, merging and dissolving polygons and lines
    - is the basis for point in polygon operations and critical for sliver removal

- therefore, an efficient algorithm to determine the intersection of two lines is very important in any GIS.

- GIS algorithms for complex processes are often built up from simple ones
    - this section will review a few simple algorithms, later sections will show how they can be built up into complex operations

- the first operation here determines if two lines cross
    - begin by examining the algorithm for two straight lines, then go on to two complex lines or polygons.
    - eventually will see that this algorithm forms the core of numerous GIS operations, including the point in polygon and polygon overlay processes.

- the algorithm illustrates one of the principles of this type of programming, that there are numerous special cases which have to be dealt with.

B. DEFINITIONS

Algorithms

- an algorithm is a procedure consisting of a set of unambiguous rules which specify a finite sequence of operations that provides the solution to a problem, or to a specific class of problems

- each step of an algorithm must be unambiguous and precisely defined
    - the actions to be carried out must be rigorously specified for each case

- an algorithm must always arrive at a problem solution after a finite number of steps
    - this must also be a reasonable number of steps

- every meaningful algorithm provides one or more outputs

- it is preferable that the algorithm be applicable to any member of a class of problems rather than only to a single problem

- in general, the cost of obtaining a solution increases with the problem size
    - if the size of the problem is sufficiently small, even an inefficient algorithm will not cost much to run
    - consequently, the choice of an algorithm for a small problem is not critical unless the problem has to be solved many times.

Heuristics

- an heuristic is a rule of thumb, strategy, trick, simplification, or any other kind of device which drastically limits the search for solutions in large problem spaces

- they do not guarantee optimal solutions, in fact they do not guarantee any solution at all
  - (definition from Feigenbaum, E.A. and J. Feldman, eds., 1963, Computers and Thought, McGraw-Hill, p.6)

C. SIMPLEST CASE

Question

Does the line from (4,2) to (2,0) cross the line from (0,4) to (4,0)? If so, where?

Procedure

- find the equations of the two lines, and solve them simultaneously for the intersection
  - the equation of a line is:

    $y = a + bx$ where b is the slope

- given two points on the line at $(x1,y1)$ and $(x2,y2)$, the slope b can be determined by the expression:

  $b = (y1 - y2) / (x1 - x2)$

  - the value of a can then be found by solving the equation using either point

Solution

- for line 1 above

  $b = (2 - 0)/(4 - 2) = 1$

  - using point 1 $2 = a + 4$ thus $a = -2$
  - the equation is $y = -2 + x$

- for line 2 similarly, $y = 4 - x$

- solving simultaneously, the two lines intersect at (3,1)

General form

- in general, the two lines

  $y = a1 + b1x$ and $y = a2 + b2x$

  intersect at:

  $xi = - (a1 - a2) / (b1 - b2)$ $yi = a1 + b1xi$

- however, this merely finds the intersection point between two lines of infinite length

passing through each pair of points
  - it is not yet established that the intersection point lies between the pairs, rather than on the imaginary extensions of one or both lines:

diagram

- an intersection point at xi lies between x1 and x2, i.e. on line 1, if:

  (x1 - xi) (xi - x2) >= 0

- similarly the point lies on line 2 with endpoints (u1,v1) and (u2,v2) if:

  (u1 - xi) (xi - u2) >= 0

- because either line can be vertical (e.g. x1 = xi = x2) we must also check similar conditions on the y coordinates to be sure the lines intersect

Simple program

overhead/handout - Simple program to compute the intersection of two lines

- the handout lists a rudimentary program for determining if two lines cross
  - x and y are used for line 1, u and v for line 2

    input x1,y1 input x2,y2 input u1,v1 input u2,v2

    b1 = (y2-y1)/(x2-x1) (A) b2 = (v2-v1)/(u2-u1) (B)

    a1 = y1-b1*x1 a2 = v1-b2*u1

    xi = - (a1-a2)/(b1-b2) (C) yi = a1+b1*xi if (x1-xi)*(xi-x2)>=0 AND (u1-xi)*(xi-u2)>=0 AND (y1-yi)*(yi-y2)>=0 AND (v1-yi)*(yi-v2)>=0 then print "lines cross at",xi,yi else print "lines do not cross" end if

D. SPECIAL CASES

- unfortunately this program will get into trouble in certain special cases:

Vertical lines

- if line 1 is vertical, the instruction labeled (A) will cause an error because of an attempt to divide by zero, as numerical processors cannot deal with infinity

- similarly if line 2 is vertical, line (B) will cause an error

Parallel lines

- if the two lines are parallel, line (C) will cause an error

Solution

- to deal with these special cases we must make the program a little more complex:

  handout - Revised program to calculate intersection

- these special cases occur in many simple geometrical algorithms.
  - they can be avoided to some extent by using different approaches

E. COMPLEX LINES

- consider two complex lines of n1 and n2 straight line segments respectively:

  diagram

  - these can be processed for all intersections by looping the simple algorithm, testing every segment in one line against every segment in the other
  - the amount of work to be done is proportional to the product (n1 x n2)

- can reduce the amount of work by using a heuristic that will save future computation
  - although this requires an additional processing step, overall processing time should be reduced

- examples of such methods are:

Minimum enclosing rectangle

- a minimum enclosing rectangle (MER) of a line is defined by the minimum and maximum x and y coordinates of the line

  diagram

- a very rough check for intersection can be made by seeing if the enclosing rectangles of two lines overlap overhead - Minimum enclosing rectangles
  - if they do not intersect, the lines cannot intersect
  - if they do intersect, then find the MERs for each straight segment of each line to see which, if any of these intersect

Monotonic sections

- can divide each line into sections which are monotonically increasing or decreasing in x and y

- monotonic segments are such that:
  - a straight line parallel to either x or y axis cuts the section at most once
  - there is a break where ever x or y hits a local maximum or minimum

  overhead - Monotonic sections

- this sets up conditions which can be used to reduce the amount of work done in looking for intersections.

- as the segment continues to increase in one direction, it cannot turn and intersect the other line again

overhead - Intersection of monotonic sections

- this is the approach used by ARC/INFO and other vendor GISs
    - the number of calculations required to determine the intersection of two complex lines drops from a value proportional to (n1 x n2) to a value proportional to (n1 + n2)

Sorting lines

- when there are many lines with many intersections to be processed, as in the overlay of two complex coverages
    - can sort the lines by their ranges so that only lines in similar ranges will be considered together

REFERENCES

Douglas, David H., 1974. "It makes me so cross," a paper distributed by the Laboratory for Computer Graphics and Spatial Analysis, Graduate School of Design, Harvard University, September 1974. (This note has been reprinted numerous times in computing magazines, and in Marble, Calkins and Peuquet, 1984.)

Lee, D.T. and F.P. Preparata, 1984. "Computational geometry: a survey," IEEE Transactions on Computers C-33(12):1072- 1101. A good introduction to basic algorithms for geometrical problems.

Little, J.J. and T.K. Peucker, 1979. "A recursive procedure for finding the intersection of two digital curves," Computer Graphics and Image Processing 10:159-71.

Marble, Duane F., Calkins, Hugh W. and Peuquet, Donna J., eds., 1984. Basic Readings in Geographic Information Systems, Williamsville N.Y., SPAD Systems Ltd.,

Saalfeld, Alan, 1987. "It doesn't make me nearly as CROSS," International Journal of Geographical Information Systems 1(4), pp 379-386

Taylor, George, 1989. "Letters," International Journal of Geographical Information Systems 3(20):192-3. Another look at the line intersection problem.

DISCUSSION AND EXAM QUESTIONS

1. Why is the "crossing segment" or line intersection problem so important in GIS?

2. Identify the rules which can be used to limit searching for intersections between two lines which are both monotonic in x and y. Each line will be one of four combinations - either increasing or decreasing in x, and either increasing or decreasing in y. You will need to deal with 16 combinations when discussing the options for two lines.

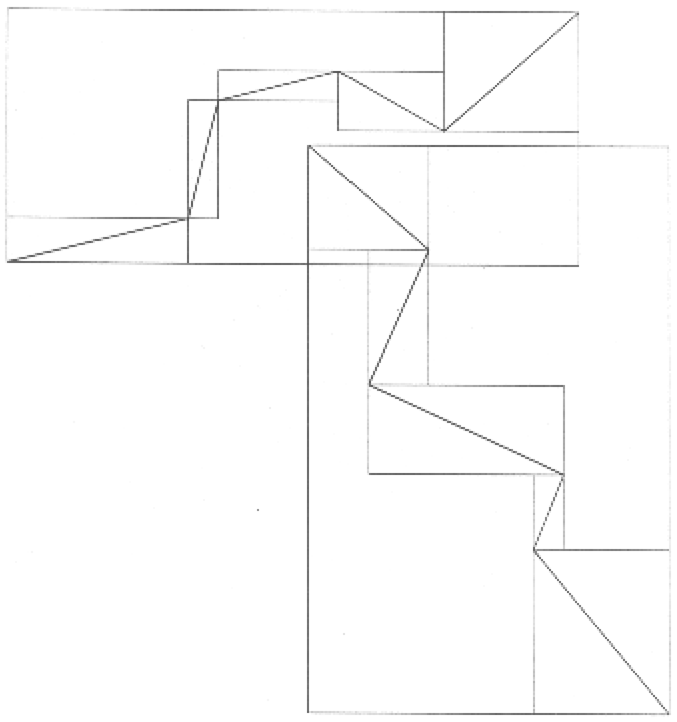
3. Review and discuss the technique for line intersection described in Little and Peucker, 1979.

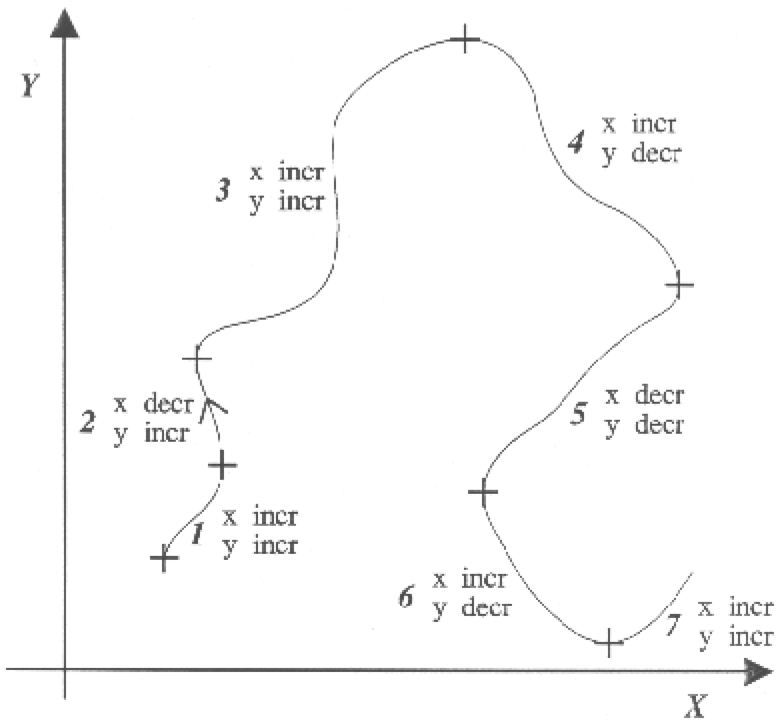
4. Compare raster and vector approaches to the determination of the intersection between two lines. Are there circumstances under which a raster approach might be preferable?

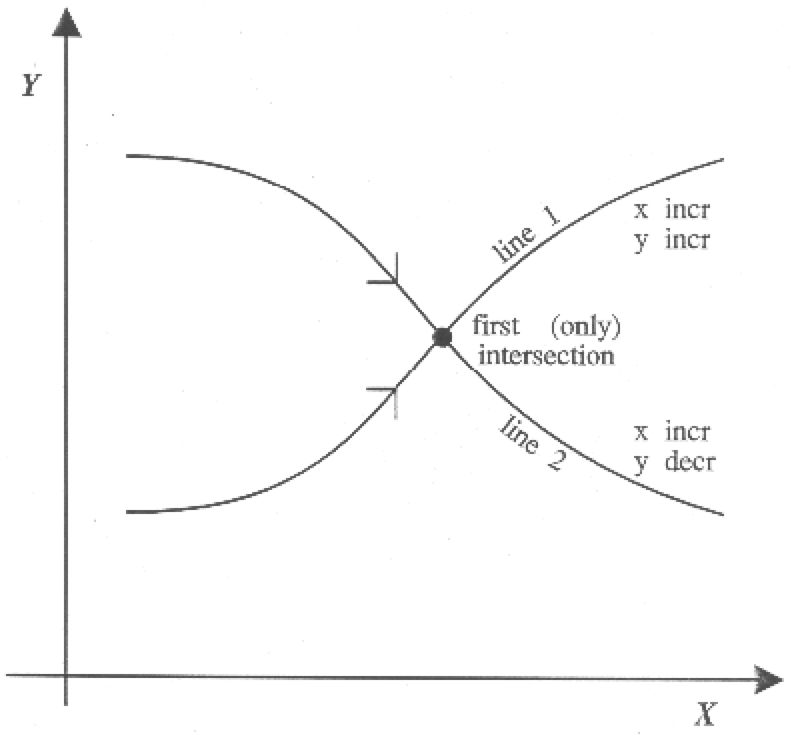
5. Since geographical data is never perfectly accurate, the special cases identified in the algorithm should never occur precisely. Modify the algorithm to deal with special cases by treating data as imprecise. What are the advantages and potential problems with such an approach?

Last Updated: August 30, 1997.

UNIT 32 IMAGES






Y
line 1
x incr
y incr
first (only)
intersection
line 2
x incr
y decr
X