Teaching GIScience: A Computational Perspective

Marc P. Armstrong

1. Professor and Chair, Department of Geography
2. Professor, Applied Mathematical and Computational Sciences
3. Interim Director, School of Journalism and Mass Communication
4. CLAS Collegiate Fellow
5. Flood Survivor
According to BoK report:

- GC. Geocomputation
  - GC1 History and trends in geocomputation
  - GC2 Uncertainty
  - GC3 Computational aspects and neurocomputing
  - GC4 Fuzzy sets
  - GC5 Cellular automata (CA) models
  - GC6 Heuristics
  - GC7 Genetic algorithms
  - GC8 Agent-based models
  - GC9 Activity analysis
Computational GIScience?

- According to BoK report:
- GC. Geocomputation
  - These topics look pretty familiar

- Purpose: to present some ideas about new/old approaches to computational GIScience education at advanced UG and graduate levels
Relation to Marble Pyramid

An Outcome Oriented Model of GIS Education

- GIS Research and Software Development
- GIS System Design
- GIS Application Design and Development
- Higher Level Modeling Applications
- Routine Use of Basic GIS Technology
- Basic Spatial and Computer Understanding

Duane Marble  December 1997
Duane asserts, and I agree, that there is a problem with education and training at the “top level” (which I construe as ~ PhD)

But I also think that the structure here doesn’t quite reflect what is going on in at least some areas of GIScience education
Evidence to support this assertion?

Most undergraduate intro textbooks:
- Baby math
- No algorithms
- No code

That wasn’t always the case, but the market for enrollments has driven us into a race to the bottom

The idea seems to be “make it easy and snag them”
To get to the top of Duane’s pyramid, to do research in computational GIScience, students must be able to:

- Formulate problems and design new, unique solutions
- Understand algorithms and program well
- Conduct experiments
- Test and evaluate innovative software “solutions”
Marc’s Marble Modification

3M®

Based on Duane Marble’s Outcome Oriented Model of GIS Education
I am not criticizing the approach taken by programs that focus on generating a “GIS Workforce” of software users.

I am focused on the “blue peak” instead.
Coding & Algorithm Knowledge

- Obvious solution: Take CS classes
- My PhD students go to “CS boot camp” & some get MCS degrees (or near to it)
- Some have CS faculty as external members of their PhD committee
- But CS is not a source for application-specific knowledge (and makes no such claim)
- GIScience must play a prominent role
It is often useful to study what has been done in a particular area, before moving forward.

A computational GIScience student should become familiar with previously described algorithms and codes.
The Bad Old Days

- We had books with icky Fortran & Basic
- Enter code by hand, modify and learn
- These were sources of knowledge that could be improved
Books with Code

- **Tobler, 1970** (FTN, *Selected Computer Programs*)
- **Rushton, Goodchild & Ostresh, 1973** (FTN, Camp Algorithm)
- **David Douglas, 1974** (FTN, *Collected Algorithms*)
- **MacDougall, 1976** (FTN)
- **Eytran, 1979** (Eyton, J. R., and Roseman, C. C., *An Introduction to Fortran and the Programming of Spatial Data*, Paper Number 13, Occasional Publications of the Department of Geography, University of Illinois at Urbana-Champaign, 139 pp.)
- **MacDougall, 1983** (BASIC)
- **Goodchild and Noronha, 1983** (BASIC, Loc/Alloc)
- **Lincoln Institute of Land Policy, 1984** (UDMS, Interpolation)
- **Clarke, 1990** (C, *Analytical Cartography*)
Outcome of “Camp Algorithm”

COMPUTER PROGRAMS FOR LOCATION-ALLOCATION PROBLEMS

Monograph Number 6
Department of Geography
The University of Iowa
Iowa City, Iowa 52242

July, 1973

Gerard Rushton
Michael F. Goodchild
Lawrence M. Ostresh, Jr.
Most by DD, but some others

<table>
<thead>
<tr>
<th>SUBROUTINE</th>
<th>LINE Oriented Subroutines</th>
</tr>
</thead>
<tbody>
<tr>
<td>BRDLIN</td>
<td>1</td>
</tr>
<tr>
<td>BRDDIV</td>
<td>3</td>
</tr>
<tr>
<td>LIN</td>
<td>5</td>
</tr>
<tr>
<td>REDUCE</td>
<td>6</td>
</tr>
<tr>
<td>CROSS</td>
<td>8</td>
</tr>
<tr>
<td>POLLIN</td>
<td>11</td>
</tr>
<tr>
<td>SMTHLN</td>
<td>15</td>
</tr>
<tr>
<td>LACLIN</td>
<td>20</td>
</tr>
<tr>
<td>LINCLN</td>
<td>24</td>
</tr>
<tr>
<td>WIGGLE</td>
<td>26</td>
</tr>
<tr>
<td>DELOOP</td>
<td>29</td>
</tr>
<tr>
<td>TRACKS</td>
<td>32</td>
</tr>
<tr>
<td>DBLLIN</td>
<td>36</td>
</tr>
<tr>
<td>DOTLIN</td>
<td>40</td>
</tr>
<tr>
<td>PLANT</td>
<td>42</td>
</tr>
</tbody>
</table>
Bruce's FORTRAN 1976

The main program which calls INTERP must first read the coordinates and values of the data points and, if they are not scaled or in row and column form, call SCALE. (The final argument to SCALE must be set to 0 so that there is no reduction of column coordinates to line numbers.) The program then calls INTERP with arguments identifying the vectors of row and column coordinates and values, the matrix size, the number of data points to be used in the interpolation calculations, and an argument set to 0 or 1 to indicate a square or rectangular grid. If elements representing squares are desired, the argument is set to 0, and the matrix size in the arguments for INTERP is the same as that for SCALE. If a rectangular grid is desired, the argument is 1, and the number of columns specified for INTERP must be three-fifths of the number for SCALE. The final part of the main program will call MAP2 or MAP4. Writing such a main program is left as an exercise for the reader.

The initial section of INTERP specifies the variables used and describes their purposes:

SUBROUTINE INTERP(X,Y,WS,J,WS,WS,MESS,MAXCELL,MESS,MAXCELL,MESS)

1. TWO DIMENSIONAL INTERPOLATION BY WEIGHTED AVERAGE OF NEAREST N POINTS
2. SUBROUTINE REPORT TOT
3. COORDINATES ARE STORED IN X (ROW), Y (COLUMN)
4. ROW COORDINATES ARE STORED IN X, COLUMN COORDINATES IN Y
5. NPTS IS NUMBER OF POINTS
6. NPTS IS NUMBER OF POINTS
7. MAP IS THE OUTPUT ARRAY NAME BY NECESS
8. REAL MESS,MAXCELL
9. IF MESS+1, MAX REPRESENT A RECTANGULAR GRID FOR COMPUTER MAPPING
10. WITH A S1 LINE PER INCH PREPARE
11. M IS THE NUMBER OF DATA POINTS TO BE USED IN THE INTERPOLATION
12. DISTANCES FROM THE CENTER OF A GRID CELL TO ALL
13. DATA POINTS ERADIUS MAXIMUM OF 1000
14. DIMENSION DIST(1000)
15. WRITE TO ENSURE OF 100 AND VALUES OF (F(X)Y)
16. DATA POINTS WITHIN SEARCH RAIDUS
17. DIMENSION (MAXCELL,100)

The interpolation computations for each matrix element consists of three steps:

1. compute the distances between the centre of the grid cell and all data points;
2. determine the M smallest distances (using SORT);
3. solve the interpolation equation

\[ X_i = \frac{\sum_{j=1}^{N} (x_{ij}D_j^2)}{\sum_{j=1}^{N} (1/D_j^2) } \]

and store the result in the matrix element.

The FORTRAN coding for these three steps is presented below under the control of two DO statements. Also incorporated in these statements is the allowance for either square or rectangular cells (cards 24, 25, and 31), a check for cases where a data point coincides with the centre of a cell (cards 36, 39, and 40), and a technique which reduces the time required in the second step. This consists of a search radius set to an initial value before the loop (card 23), and used to limit the size of arrays sent to SORT (card 44). If this radius is too small, it is reset to a value of 25 per cent larger (card 51), and if it is too large, it is reduced by 10 per cent (card 65).

C RADICAL INITIAL SEARCH AREA
C ARE INITIALLY TO MXR HD REQUISITE TO CONTAIN N POINTS IF POINTS
C ARE SORTED, MXRHD IS ACCEL/1 (NPTS)
C SCALER=1.
C IF MXRHD+13 SCALER=1.75.

The FORTRAN coding for these three steps is presented below under the control of two DO statements. Also incorporated in these statements is the allowance for either square or rectangular cells (cards 24, 25, and 31), a check for cases where a data point coincides with the centre of a cell (cards 36, 39, and 40), and a technique which reduces the time required in the second step. This consists of a search radius set to an initial value before the loop (card 23), and used to limit the size of arrays sent to SORT (card 44). If this radius is too small, it is reset to a value of 25 per cent larger (card 51), and if it is too large, it is reduced by 10 per cent (card 65).
LISTING 6.1 SLOPE

18 REM SLOPE
28 REM Program to compute slope magnitude and direction for a DTM (matrix of
29 REM elevations).
38 REM Programmed by E. Bruce MacDougall, January 1983.
48 REM Variables (alphabetically):
78 REM A - DTM orientation
48 REM E1 - matrix of elevations (DTM)
98 REM E2 - south-north elevation difference at a point
118 REM F1 - file names
128 REM M - number of rows in DTM
138 REM N - number of columns in DTM
158 REM N1 - N1
168 REM N2 - N2
178 REM PS - screen clearing code
188 REM R - spacing between mesh points in DTM
198 REM SE - slope values
208 REM S1, S2, S3 - slope magnitude in per cent
218 REM S1, S2, S3, S4 - slope aspect as an azimuth in degrees
228 REM U8 - name of data set
238 REM ..............................
258 REM Set screen clearing code.
268 PRINT CHR$(27)+CHR$(12)
278 REM ..............................
288 PRINT * "File name of DTM ";
298 INPUT FS
308 OPEN "", FS, FS
318 INPUT #1, FS
328 PRINT "DTM name ".
338 PRINT WF
348 PRINT "In this the correct DTM (Y/N) ";
358 INPUT DG
368 IF DG = "Y" THEN 400
378 CLOSE #1
388 REM ..............................
398 GOTO 200
408 INPUT #1, M, N, R, A
418 REM Double mesh spacing to speed calculations.
428 R = R/2
438 M = M/2
448 N = N/2
458 A = A/2
468 REM ..............................
478 DO N1 = 1, N1, M1
488 DO N2 = 1, N2, M2
498 PRINT FS
508 PRINT "Please wait while I compute."
518 REM ..............................
528 REM Input elevation data.
538 FOR I = 1 TO M
548 FOR J = 1 TO N
558 INPUT #1, E1(I, J)
568 NEXT J
578 NEXT I
588 REM Compute slope magnitudes and azimuths.
598 FOR I = 1 TO M
608 FOR J = 1 TO N
618 REM South-north difference.
Result of Micro Workshop

Lincoln Institute of Land Policy

SPATIAL ALGORITHMS FOR PROCESSING LAND DATA WITH A MICROCOMPUTER

Lincoln Institute Monograph #84-2

Vince Robinson’s UDMS

Rushston and Kohler (1973) have shown that this algorithm is superior to others in terms of finding the minimum value. However, this superiority is paid for in processing time. In the example of the Jamaican development project, a 5 center by 84 node problem took approximately 25 minutes to process using the CBASIC version of NETLOC1, NETLOC2, and NETLOC3 on a Radio Shack Model II. To reduce the processing time the distance variables, \( D \), MINN2%, and MINN% (Figures 4-6, 4-7, & 4-8) are integer variables. Another aspect of this algorithm is that the distance matrix \( D \) is stored as a vector rather than a matrix to be more efficient in terms of memory management.

Output consists of a table indicating the nodes allocated to each of the centers, total aggregate distance, starting vertices, and ending vertices (Figures 4-9 & 4-10). These results are printed at each cycle of the Teitz and Bart method. A final table presents the beginning source node configuration and the final solution configuration (Figure 4-11).

Fig. 4-6: Location/Allocation on a Network Algorithm:
First Program Module

COMMON ME,MP%,ICON%,LMS%,NIL%,IOF%
COMMON MFF%,NVT%,JG%,LBF%
COMMON DX(1),IX(1),IXE(1),ilo(1)
COMMON INVX(1),IASEX(1),IASEX(1),JESC(1)
COMMON JOX(1),MINN%(1),MINN%(1)
REM
REM OPTIMUM FACILITY LOCATION-ALLOCATION ON A NETWORK
REM SETS UP ALGORITHM PARAMETERS
REM
INPUT " ENTER NUMBER OF CENTERS TO BE LOCATED ";MP%
NIL%-0: NIL%=0
INPUT " DO YOU HAVE A CONSTRAINT FILE ON DISK(Y/N) ";YN$
IF YN$ = "N" THEN 100
PRINT " ENTER NAME OF CONSTRAINT FILE 
INPUT ";:LINE CONSt.FILS
OPEN CONSt.FILS AS 1
100 PRINT " ENTER NAME OF NETWORK DISTANCE MATRIX 
INPUT ";:LINE NDIST.MATS
7.7 HILLS

1000 1 HILLS - HILLSMAN WEIGHTED DISTANCE EDITING ROUTINE
1010 DIM TOW(1000), WT(B1000)
1020 MAXL=700
1030 PENALTY=.5
1040 1 GET INPUT FILE
1050 INPUT 'NAME OF INPUT FILE*FILE#' INPUT=FILE
1060 OPEN FILE1 FOR INPUT AS FILE 1
1070 OPEN FILE2 FOR OUTPUT AS FILE 2
1080 OPEN FILE3 FOR OUTPUT AS FILE 3
1090 1 GET OPTION
1100 PRINT 'EDITING OPTIONS?*
1110 PRINT * 1 RMX WITH MAX DISTANCE CONSTRAINT*
1120 PRINT * 1 RN CENTRES WITH MAX DISTANCE CONSTRAINT*
1130 PRINT * 1 MAXIMAL COVERING PROBLEM*
1140 PRINT * 2 MAXIMAL COVERING WITH MAX DISTANCE CONSTRAINT*
1150 PRINT * 3 ATTENDANCE MAXIMIZE (LINEAR DECAY)*
1160 PRINT * 4 MINIMIZE TOTAL POWERED DISTANCE*
1170 INPUT 'OPTION/OPTION' OPTION
1180 1 CHECK DOMAIN OF OPTION
1190 IF OPTION=1 THEN 1220
1200 PRINT 'RANGE OF OPTIONS IS 1 TO 4'
1210 GO TO 1100
1220 1 GET SUPPLEMENTARY CONSTANTS
1230 IF OPTION=2 THEN 1290
1240 IF OPTION=4 THEN 1290
1250 IF OPTION=6 THEN 1290
1260 IF OPTION=2 THEN 1290
1270 IF OPTION=4 THEN 1290
1280 IF OPTION=6 THEN 1290
1290 1 BEGIN READING INPUT FILE
1300 1 END OF FILE CHECK
1310 1 ERROR CODE TO 2300
1320 PRINT 'WHICH XES PLAY-WEIGHT-CANDIDATE'
1330 1 CHECK STRING LENGTH
1340 IF STRL<MAXLEN THEN 1390
1350 PRINT 'TOO MANY HES IN STRING - LIMIT IS*MAXLEN'
1360 STOP
1370 IF STRL=0 THEN 1390
1380 MAT INPUT #1:FTNOGN(STRL)
1390 MAT INPUT #1:WTDIST1(STRL)
1400 1 OFFSET FOR OPTION 5
1410 200=0
1420 1 LOOP FOR EACH ENTRY IN STRING
1430 FOR I=1 TO STRL
1440 1 COMPUTE DISTANCE
1450 330=WTDIST1(I):WEIGHT
1460 ON OPTION GO TO 1550, 1560, 1580, 1620, 1640, 1700
1470 1 COME HERE FOR OPTION 1
1480 IF DIST(I)=WEIGHT THEN WTDIST1(I)=0
1490 IF DISTR(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1500 1 COME HERE FOR OPTION 2
1510 GO TO 1750
1520 IF TONGUE(I) THEN WTDIST1(I)=0
1530 IF TONGUE(I) THEN WTDIST1(I)=WEIGHT
1540 IF DIST(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1550 IF DISTR(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1560 GO TO 1750
1570 1 COME HERE FOR OPTION 3
1580 IF DIST(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1590 IF DISTR(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1600 GO TO 1750
1610 1 COME HERE FOR OPTION 4
1620 IF DISTR(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1630 IF DIST(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1640 IF DISTR(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1650 GO TO 1750
1660 1 COME HERE FOR OPTION 5
1670 IF DISTR(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1680 GO TO 1750
1690 1 COME HERE FOR OPTION 6
1700 IF DISTR(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1710 1 COME HERE FOR OPTION 7
1720 IF DISTR(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1730 IF DISTR(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1740 IF DISTR(I)=WEIGHT THEN WTDIST1(I)=WEIGHT
1750 STOP

BY

MICHAEL F. GOODCHILD
VALERIAN T. NORONHA

MONOGRAPH 8
DEPARTMENT OF GEOGRAPHY
THE UNIVERSITY OF IOWA
IOWA CITY, IOWA 52242
SEPTEMBER, 1983
Post Clarke 1990, things got dark

- It was a long decade, but things have improved since Y2K in math and CS
- Wood, 2002 (Java intro, nicely done)
- Shekhar and Chawla, 2003 (Spatial Databases)
- O’Sullivan and Unwin, 2003 (Spatial Analysis)
- Worboys and Duckham, 2004 (CS GIS)
- De Smith, Goodchild and Longley, 2007
And there was always LCGSA
What about GRASS?

- Yes, there is code there...
- But, it is *probably* too complex for educational use, as is
- Possibly “mineable”
Another “Source”

- *Computers and Geoscience* doesn’t seem like “us”?
- Open source code? Do we need a GIScience code portal?
- We had a version of one long ago called Geography Program Exchange
- A brief historical digression...
Geography Program Exchange

- To assist with interchange of computer software which relates to problems of a geographic nature.

- Operates on a self-sustaining basis with programs, documentation and test-data sets available at cost. Distribution of programs is in the form of listings, punched cards, or magnetic tapes.

- Inquiries to Dr. R. I. Wittick, Computer Institute for Social Science Research, Michigan State

Source: Mathematical Geology, Vol. 5, No. 3, 1973
Program to construct a contour map from a series of data points whose locations are defined in terms of grid squares.

**Calling Name:** CONTOUR

**Contributors and Affiliations:**
- Duane F. Marble
- Department of Geography
- Northwestern University
- Evanston, Illinois

**Installation Name:** CRAY X-MP/28

**Program Abstract:**

The program establishes a map grid from the data read in, the row and column positions (i.e., the data points) are located on the map output by a series of maps each one from the margins and across the top and bottom edges of the map. The contour bands are then located between the output of the two-dimensional autocorrelation function, which is discussed by Heiskanen and Moritz (1967), and is written in the form of equations in two-dimensions should be recognized.

**More specifically:** The program establishes a lattice and estimates a value at each matrix point by using a weighted average of the six nearest data points. The weights are the inverses of the squares of the distances between the data points from the lattice point (i.e., linear interpolation). These weighted average is then averaged with the value at the nearest observation point implying an autocorrelation which has a large negative slope in the vicinity of the origin to give the final estimate.

**Program **GRID** differs from **GRID** (an earlier version, also available from the program exchange) by allowing variables, all recorded at the same locations, to be interpolated sequentially.

**References:**

**Definition of Options:**
- **ROWS:** The number of rows in the data grid matrix. There is no restriction on the number of rows which may be used.
- **COLS:** The number of columns in the data grid matrix. The number of columns in any one map must not exceed 40. If a larger grid is used, the map must be made in sections by using only a part of the data in each pass. Each section is treated as a separate map. The number of columns in each pass should be identical so that all the maps will be at the same scale. To do this, it may be necessary, and it is advisable, to repeat some columns.
- **CONT:** The number of contour bands desired on the map output. CONT may take on any integer value between 2 and 19. However, if CONT is 0, the program automatically uses 6, except in the case where **TOUR** = 1. Then the program must be set to CONT = 0, and the program automatically sets CONT = 17.
- **TOUR:** The contour interval desired. If **TOUR** = 0, the program uses the maximum and the minimum of the data. The data range (i.e., the maximum minus the minimum) is then divided by the value specified for CONT to give the contour interval. If **TOUR** = 1, the desired values for the top and bottom contours are read in by the user. Then the contour interval is calculated by dividing the range between these specified values by the value given for CONT. If **TOUR** = 2, variable contour intervals are read.
I’ve talked about some materials
Methods are different
Language choice is a difficult problem since it boils down to a matter of faith
But underneath that is more basic algorithmic knowledge
Drawing a parallel with computer graphics...
What gets taught in graphics?

- Basic stuff like how to draw a line between 2 points and how to draw a circle (Bresenham)
- More advanced stuff like efficiently clipping a line to a rectangle (Liang-Barsky) and projection from 3D → 2D
- Really advanced stuff like reflectance models and ray tracing
Hence we have an iterative way to calculate \( d_i + 1 \) from the previous \( d_i \) and to make the selection between \( S_i \) and \( T_i \). The initial starting value \( d_1 \) is found by evaluating (11.2) for \( i = 1 \), knowing that \((x_0, y_0) = (0, 0)\). Then

\[
d_1 = 2dy - dx. \quad (11.5)
\]

The arithmetic needed to evaluate (11.3), (11.4), and (11.5) is minimal; it involves addition, subtraction and left shift (to multiply by 2). This is important, because time-consuming multiplication is avoided. Further, the actual inner loop is quite simple, as seen in the following Bresenham's algorithm (note that this version works only for lines with slope between 0 and 1; generalizing the algorithm is left as an exercise for the reader):

```plaintext
procedure BRESENHAM(x1, y1, x2, y2, value: integer);

var dx, dy, incr1, incr2, d, x, y, xend: integer;

begin
  dx := ABB(x2 - x1);
  dy := ABB(y2 - y1);
  d := 2 * dy - dx; \{ initial value for d from (11.5) \}
  incr1 := 2 * dy; \{ constant used for increment if \( d < 0 \) \}
  incr2 := 2 * (dy - dx); \{ constant used for increment if \( d > 0 \) \}
  if x1 > x2 then begin \{ start at point with smaller x \}
    x := x2;
    y := y2;
    xend := x1
  end
  else begin
    x := x1;
    y := y1;
    xend := x2
  end

while x < xend do begin \{ first point on line \}
  x := x + 1;
  if d < 0 then d := d + incr1 \{ choose \( S_i \) — no change in \( y \) \}
  else begin
    y := y + 1;
    d := d + incr2 \{ choose \( T_i \) — \( y \) is incremented \}
  end
  WRITE_PIXEL(x, y, value); \{ the selected point near the line \}
end \{ while \}
end \{ BRESENHAM \}
```

For a line from point \((5, 8)\) to point \((9, 11)\), the successive values of \( d \) are 2, 0, -2, 4, and 2. Figure 11.3 shows which pixels are set and the ideal path of the line.

It would be convenient to make the translation "add" into a form for single step operations.

Borrow a "trick" from homogenous coordinates.
P(x, y) is represented as:

\[
P(W \cdot x, W \cdot y, W)
\]

with scale factor \( W \neq 0 \) with \( W \) always = 1 ...

Translation

\[
\begin{pmatrix}
  x' \\
  y' \\
  1
\end{pmatrix} = \begin{pmatrix}
  x \\
  y \\
  1
\end{pmatrix} \begin{pmatrix}
  1 & 0 & 0 \\
  0 & 1 & 0 \\
  0 & 0 & 1
\end{pmatrix}
\]

concatenating translate then,

\[
\begin{pmatrix}
  x' \\
  y' \\
  1
\end{pmatrix} = \begin{pmatrix}
  x \\
  y \\
  1
\end{pmatrix} \begin{pmatrix}
  \cos \theta & \sin \theta & 0 \\
  -\sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
\end{pmatrix} \begin{pmatrix}
  3 & 0 & 0 \\
  0 & 3 & 0 \\
  0 & 0 & 1
\end{pmatrix}
\]

3x3 but 3 ops are 0 or 1...
What do CS students do?

- They take quizzes & tests to demonstrate their knowledge of algorithms
- They write “machine problems” in which they design, implement and demonstrate that they can accomplish assigned programming tasks
What are our students doing?

- Not that.
- Some students are hand-crafted, artisanal
- Is that any way to run a scientific enterprise?
What to do?

- Interdisciplinary approach to provide foundational instruction in CS fundamentals; GISci is basic science and application domain.
- Computational geometry, linear algebra, data structures, sort/search, optimization.
- Focus on GIS algorithms, code examples (projections, shortest path, interpolation, polygon fill, line and point symbols, overlay, buffer).
- At higher level, parallel (MPI, OpenMP), software engineering, SOA...
What to do?

- Can a renewed focus on algorithms & code play a role in GIScience Knowledge Web?
Concluding Comments

- We started out stumbling along a more-or-less correct path
- We lost that path, relying on software and texts that pander to “users”
- Need to reclaim and reinvent what passes for high-level education—this involves basic research and development rather than use...
That’s all for now...