Unit 057 - Quadtrees and Scan Orders

by Michael F. Goodchild, University of California, Santa Barbara

---

Advanced Organizer

Topics covered in this unit

- raster scan orders - Bostrophedon, Morton, Pi-order
- introduction to quadtrees, how they are constructed
- applications of quadtrees
- quadtree addressing

Learning Outcomes

- after learning the material covered in this unit, students should be able to:
  - explain the significance of the order in which a raster is scanned, and the benefits of certain alternatives
  - explain how scan order is used in GISs
  - define the definition of a quadtree and be able to construct one
  - describe the advantages of quadtrees
  - explain how quadtrees are used in indexing and retrieving objects

Full Table of Contents

Metadata and Revision History

Unit 057 - Quadtrees and Scan Orders

1. Scanning the raster

- Unit 055 introduces rasters
  - that unit assumes the raster is scanned in row-by-row order

- there are several reasons for being interested in other scan orders:
  - potential for loss-less compression
  - efficient indexing of objects in GIS
  - efficient arrangement of tiles on storage devices
  - they are the basis of quadtrees
- in all of these cases the central issue is performance
  - how to process a given amount of geographic data as rapidly as possible

- there are several other possible scan orders

1.1 Bostrophedon scan order

- reverse every other row

63	62	61	60	59	58	57	56
48	49	50	51	52	53	54	55
47	46	45	44	43	42	41	40
32	33	34	35	36	37	38	39
31	30	29	28	27	26	25	24
16	17	18	19	20	21	22	23
15	14	13	12	11	10	9	8
0	1	2	3	4	5	6	7

  - for reasons that will become clear later, the rasters in this unit will be numbered starting with 0 rather than 1
- this is the ***boustrophedon*** order, from the Greek for "how an oxen plows a field"

- every move is to an edge-neighbor, there is no flyback at the end of each row as with standard row order
  - consequently, the compression achievable with this order is greater than with row order, because adjacent cells are more likely to be the same

- following is the data from Unit 055 to see if that proposition works in this case:

1 2 1 1 1 2 3 4

1	1	1	1	2	2	3	3
1	1	1	1	1	2	2	3
4	4	1	1	2	2	3	3
4	4	4	1	2	2	2	3
4	4	4	4	2	2	3	3
4	4	4	2	2	2	2	3
4	4	4	4	2	2	2	2

    - total is now 23 runs, down from 28

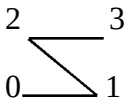
- the maximum potential reduction is (*n*-1)

1.2. Morton Order

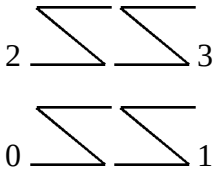
- a much more radical option is this:

42	43	46	47	58	59	62	63
40	41	44	45	56	57	60	61
34	35	38	39	50	51	54	55
32	33	36	37	48	49	52	53
10	11	14	15	26	27	30	31
8	9	12	13	24	25	28	29
2	3	6	7	18	19	22	23
0	1	4	5	16	17	20	21

- start in the lower left corner as before
- execute a basic pattern



- repeat the pattern three more times



- the set of four repeats forms the same pattern as we started with, but at a higher level
- then blocks of 16 repeat the same basic pattern
- this can be extended infinitely - the next sequence will be blocks of 64, ordering a 16 by 16 matrix
- the ordering can only be applied to a square matrix, of dimensions 2^l by 2^l, where l is an integer
    - e.g. 2 by 2, 4 by 4, 16 by 16, 64 by 64
- a raster of smaller dimensions can be filled out to the next power of 2

- this order has been rediscovered and renamed many times
    - it was first implemented in GIS by Guy Morton working for IBM Canada in 1966, so is often called the ***Morton order***
        - Morton was looking for an efficient way to order the tiles of a geographic database (each tile corresponding to one map sheet) on a magnetic tape
        - as far as possible, he wanted the order on the tape to preserve geographic adjacency
        - by coincidence the county in the southwestern corner of Kansas is Morton County
    - from the basic shape that is repeated at every scale, it is sometimes called the ***N-order*** or the ***Z-order***
    - it is also associated with the names of the mathematicians Hilbert, Peano, and Koch
    - because the shape is the same at different scales the curve is a ***fractal***

- how efficient is order as a method of compression?
    - 25 runs, compared to 28 for row order and 23 for boustrophedon
    - not a spectacular saving
    - in general, boustrophedon beats row order and Morton (Goodchild and Grandfield, 1983)

1.3. Pi-order

- finally, Pi-order is as follows:

21	22	25	26	37	38	41	42
20	23	24	27	36	39	40	43
19	18	29	28	35	34	45	44
16	17	30	31	32	33	46	47
15	12	11	10	53	52	51	48
14	13	8	9	54	55	50	49
1	2	7	6	57	56	61	62
0	3	4	5	58	59	60	63

- is the most complicated to generate

- again, various names
  - ***Pi-order*** because of the basic shape that repeats at all levels
  - often called the ***Peano-curve***

- like boustrophedon, every move is to an edge-neighbor
  - 21 runs is the best performance
  - tests have shown that this order generally gives the best results

---

2. Quadtrees

- in a raster, cells are allocated irrespective of the data
- in areas where there is less variation it might be good to use larger cells
  - this is what happens when irregular representations like polygons or TINs are used
- but it seems more difficult for rasters
  - variable-size rectangles won't fit together neatly
- the ***quadtree*** addresses this issue

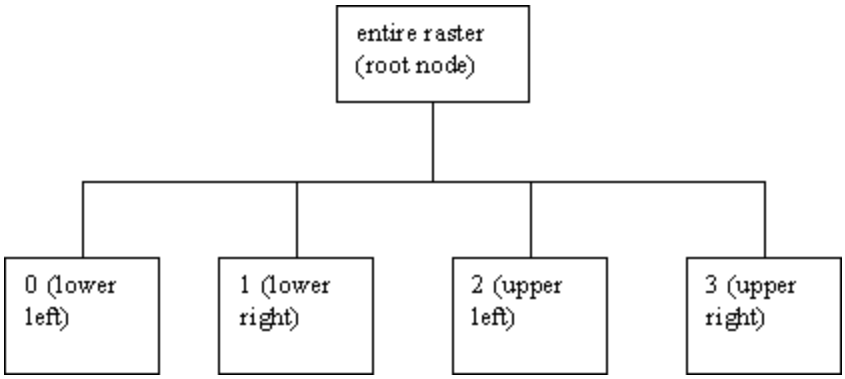
- consider the same data set

1	2	1	1	1	2	3	4
1	1	1	1	2	2	3	3
1	1	1	1	1	2	2	3
4	4	1	1	2	2	3	3
4	4	4	1	2	2	2	3
4	4	4	4	2	2	3	3
4	4	4	2	2	2	2	3
4	4	4	4	2	2	2	2

- start with the whole raster
  - is it constant valued everywhere?
    - if yes, stop
    - if no, divide into four equal quadrants

- note that this process requires a square raster with rows and columns equal to a power of two

- imagine a tree with four branches:



- notice how the nodes have been numbered
  - root node is null
  - the lower nodes have addresses consisting of one digit between 0 and 3 - one base 4 digit
  - in binary notation (base 2) the four nodes would be 00, 01, 10, 11 respectively

- in this case, none of the four quadrants is constant, so each is divided again using the same rules
  - now the nodes have addresses consisting of two base 4 digits
  - 0 divided into 00 (all 4); 01 (mixed); 02 (all 4); 03 (mixed)
  - 1 divided into 10 (all 2); 11 (mixed); 12 (all 2); 13 (mixed)
  - 2 divided into 20 (mixed); 21 (all 1); 22 (mixed); 23 (all 1)
  - 3 divided into 30 (mixed); 31 (mixed); 32 (mixed); 33 (mixed)
  - of these 16 nodes, only the mixed ones will be divided further

- the process stops at the next level when the resolution of the raster is reached

- here are the leafs at the end of the process:

```
00  4     110 2     202 1     303 2     332 3
010 4     111 2     203 1     310 3     333 4
011 4     112 2     21  1     311 3
012 4     113 3     220 1     312 2
013 2     12  2     221 1     313 3
02  4     130 3     222 1     320 2
030 4     131 3     223 2     321 2
031 4     132 2     23  1     322 1
032 4     133 3     300 2     323 2
033 1     200 4     301 2     330 3
10  2     201 4     302 1     331 3
```

- a total of 46 leafs

- note the similarity between this and the Morton order
  - the ordering of the leafs is the same
  - there are fewer Morton runs than quadtree leafs because in the quadtree there are more restrictions on the lengths of runs

- if you run through the leafs in the correct order and merge adjacent leafs with the same value you should end up with the number of Morton runs, 25

2.1 Additional quadtree example

- Figure 1 shows an additional example of a quadtree
  - Part A shows the raster, and the first level of decomposition
  - Part B shows the first three levels
  - Part C shows the full four levels
  - Part D shows the full decomposition including the tree

---

3. Applications of quadtrees

- quadtrees are tree structures, and can be accessed very quickly
  - some GIS functions can be performed much more quickly on quadtrees than on other raster structures

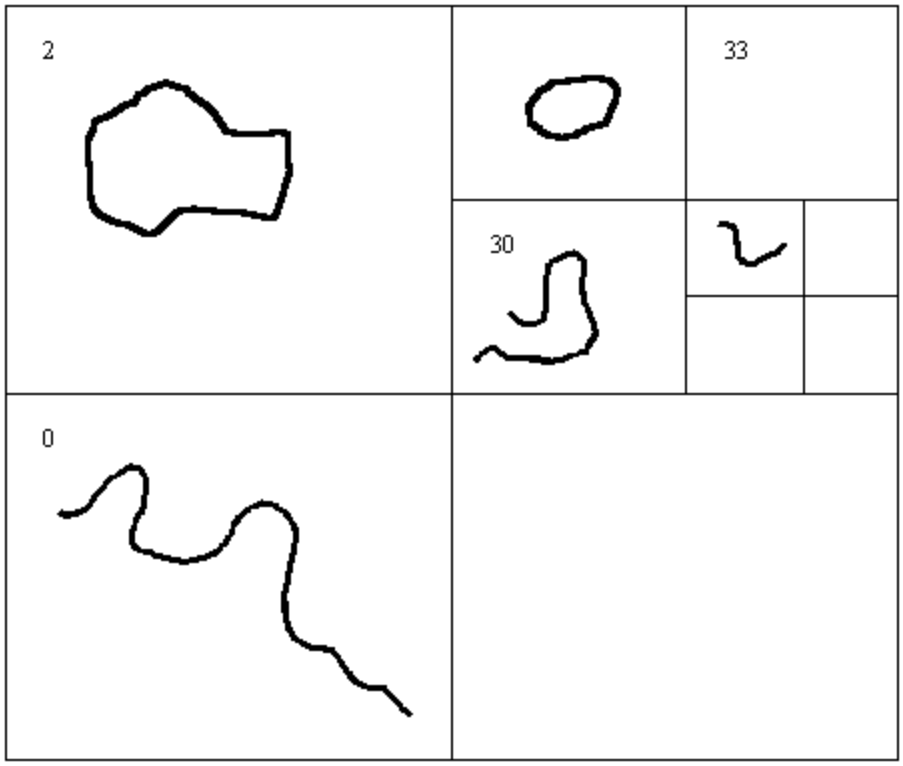
- suppose we have a raster of 2^l by 2^l cells
  - a total of 2^{2l} cells
    - this will be the number of records in a full raster structure
  - the tree has at most l levels
  - suppose there are k leafs in the tree
    - the number of records in a run length encoded (RLE) raster will be of this order for the reasons discussed discussed in Section 2, inflated by a factor of a (in the example in Section 2 this factor was 46/25 or approximately 2)
  - how many steps will be required to search for all leafs of a given value?
    - in a quadtree, k leafs must be examined
    - in a full raster, 2^{2l} records
    - in an RLE raster, ak records
  - how many steps will be required to retrieve the value in a specific cell?
    - in a quadtree, up to l levels must be accessed
    - in a full raster, the address of a given cell can be computed, so only one access is needed
    - in an RLE raster, up to ak records must be accessed

- because of these and other advantages, entire GISs have been built based on quadtrees
  - TYDAC's SPANS
  - Hanan Samet's QUILT
    - Samet's books (Samet, 1990a,b) are an excellent source on quadtrees

- quadtrees are excellent ways of indexing spatial objects in a GIS
  - suppose a large number of point, line, or area objects must be found very quickly
  - think of the nodes of a quadtree as buckets
    - for each object, find the smallest leaf that contains the object
    - store the object in the corresponding bucket

    - very large objects will be stored in buckets near the root of the tree (the top)

- here is an illustration



    - the smallest object lies in bucket 312

- for another example see Figure 2

- with this scheme it is possible to find objects very quickly based on location
    - instead of testing every object, it is necessary to test only the objects stored in the appropriate bucket plus objects in all higher-level buckets
    - e.g. to search for objects at a location in bucket 312 it would be necessary to test buckets 31 and 3 also

- this form of indexing is known as quadtree indexing or two-dimensional hashing
    - it is widely used in GIS, although the user is often unaware of the indexing scheme used by the system

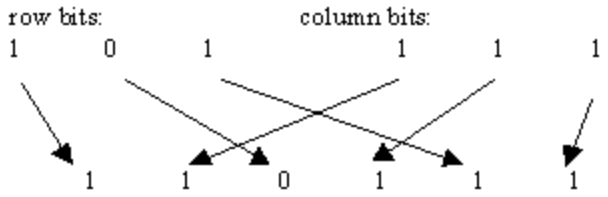
---

4. Addressing quadtree spaces

- given a coordinate (*x*,*y*), how to know where this lies in the quadtree?
    - the reverse also: given a quadtree address, how to determine where this lies in

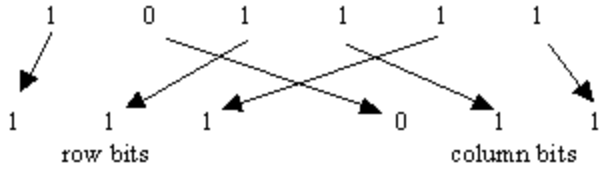
coordinate space?
    - the same techniques are used to determine where a given cell lies in Morton order, and the reverse - the row and column of a cell in Morton sequence
    - the solution is known as ***bit-interleaving***

- take a cell's row and column numbers
    - e.g. row 5 column 7
        - rows and columns and cells must be numbered starting at 0
        - in a raster of 2^l by 2^l the rows and columns run from 0 to (l-1), the cells from 0 to 2^{2l}-1
        - in this case l=3
        - row 5 column 7 is cell 55 in Morton order (that's the answer we need)
- write the row and column numbers in binary notation, using l bits
    - 5 = 101; 7 = 111
- interleave the bits, starting at the left and working to the right, and taking a row bit first
    - how to know to use a row bit first (and a column bit last)?
        - because the first move in the order was to the right, not up
        - so the *least significant bit* must be a column bit



- the result is 110111
    - converting back from binary notation, this is:
        - one 32, one 16, no 8, one 4, one 2, one 1 = 55

- reversing the process:
    - try cell 47 in Morton order
        - convert to binary, 47 = 101111
        - separate the bits, row bit first, column bit last:



    - cell 47 is in row 7, column 3

- finally, how to get a quadtree address
    - quadtree leaf addresses are in base 4
        - expand to base 2, replacing 0 by 00, 1 by 01, 2 by 10, 3 by 11
        - then proceed, or reverse, as before

5. References

Goodchild, M.F. and A.W. Grandfield (1983) Optimizing raster storage: an examination of four alternatives. *Proceedings, AutoCarto 6, Ottawa* 1: 400-407.

Mark, D.M. (1990) Neighbor-based properties of some orderings of two-dimensional space. *Geographical Analysis* 22: 145-157.

Samet, H. (1990a) *The Design and Analysis of Spatial Data Structures*. Reading, MA: Addison-Wesley.

Samet, H. (1990b) *Applications of Spatial Data Structures: Computer Graphics, Image Processing, and GIS*. Reading, MA: Addison-Wesley.

6. Exam and discussion questions

1. Why doesn't the Morton order perform better as a compression device?

2. What other potential ways are there of scanning a raster?

3. Discuss the research reported by Mark (1990).

4. Given the arguments presented above, why haven't quadtree-based raster GISs done better in the commercial marketplace (why did TYDAC add other raster structures besides quadtrees to SPANS)?

5. Try to determine the spatial indexing scheme used by your favorite GIS. How could you deduce the nature of the indexing scheme by watching the system's behavior?

Citation

To reference this material use the appropriate variation of the following format:

> Michael F. Goodchild. (1997) Quadtrees and scan orders, *NCGIA Core Curriculum in GIScience*, Unit 057, http://www.ncgia.ucsb.edu/giscc/units/u057/u057.html, posted October 23, 1997.

Created: August 8, 1997. Last revised: October 23, 1997.

Unit 057 - Quadtrees and scan orders

Table of Contents

Unit 057 - Quadtrees and Scan Orders

Metadata and Revision History

1. About the main contributors

- author
  - Michael F. Goodchild
  - Department of Geography
  - University of California
  - Santa Barbara CA
- editor
  - Karen K. Kemp
  - Deparmtent of Geography
  - University of California
  - Santa Barbara CA

2. Details about the file

- unit title
  - Quadtrees and Scan Orders
- unit key number
  - 057

3. Key words

4. Index words

5. Prerequisite units

- none

6. Subsequent units

- all

7. Other contributors to this unit

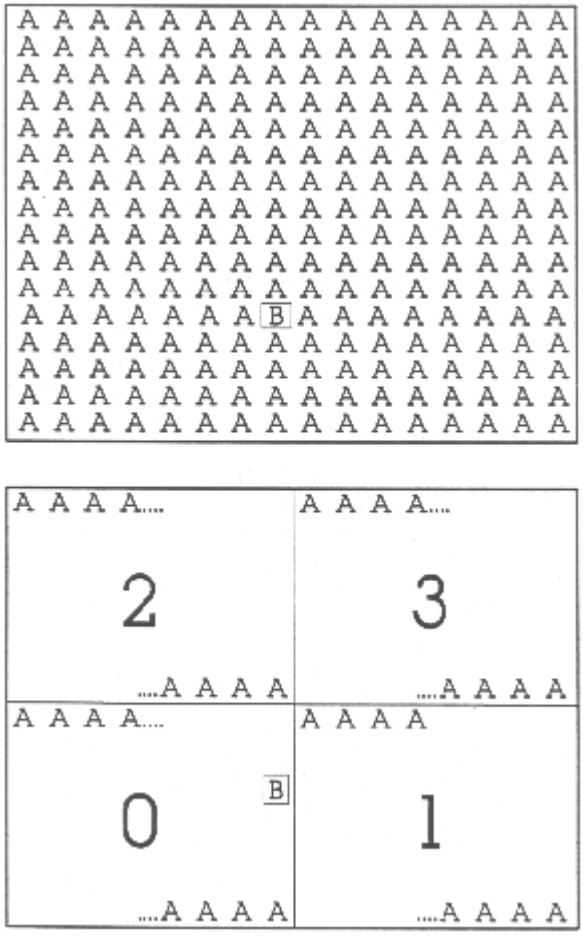
8. Revision history

- Auguest 8, 1997 - original draft created

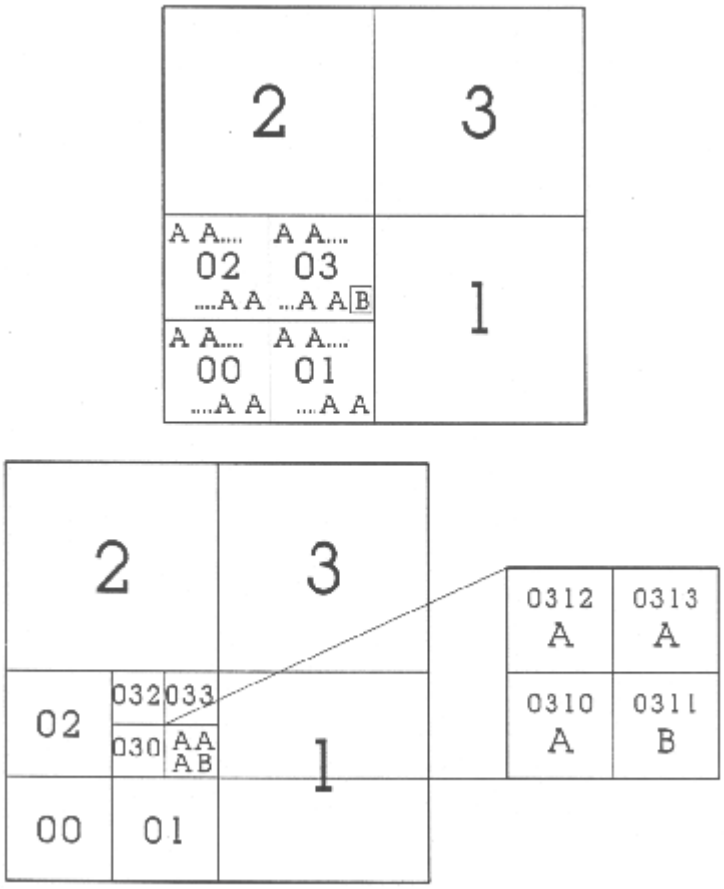
- September 16, 1997 - original draft posted to net
- October 9, 1997 - minor revisions completed
- October 23, 1997 - more minor revisions

Back to the Unit.

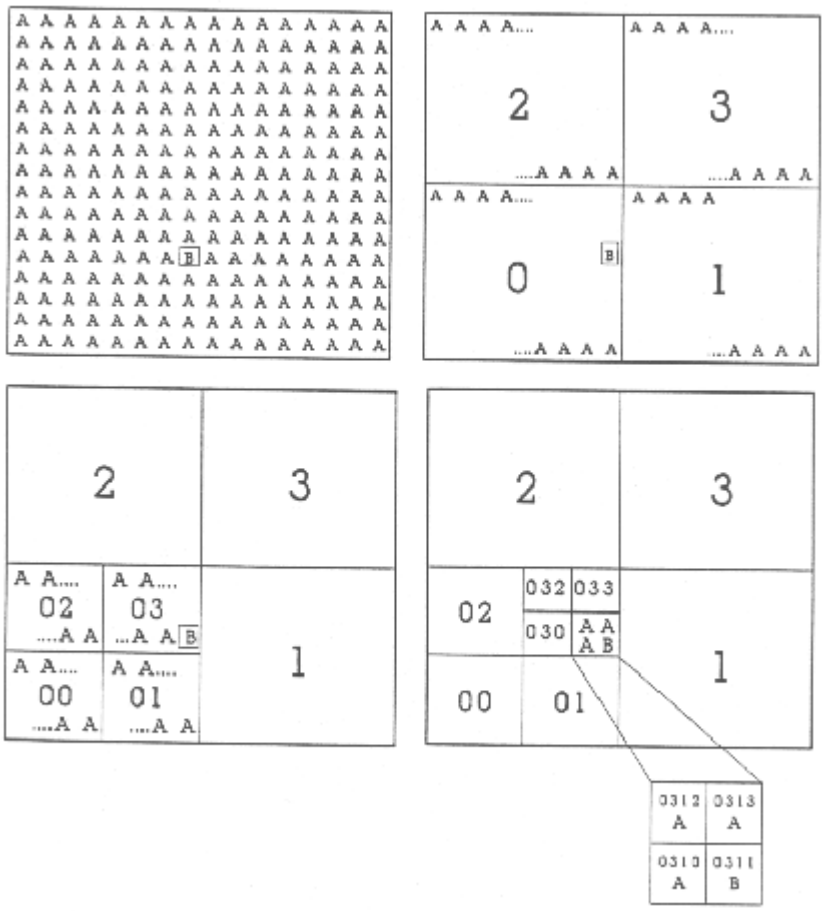
Figure 1. Additional Quadtree Example



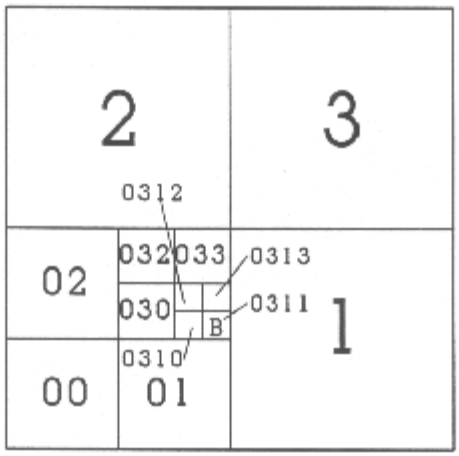
Part A.

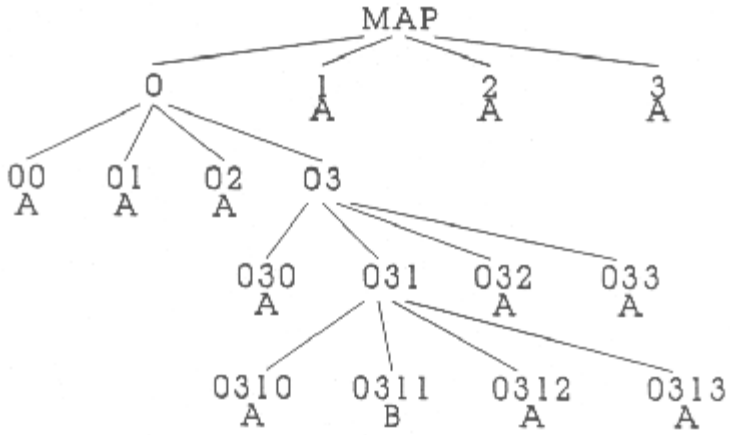

2
3
A A....
02
....A A
A A....
03
...A A B
A A....
00
....A A
A A....
01
...A A
1
2
3
02
032
033
030
AA
AB
00
01
1
0312
A
0313
A
0310
A
0311
B

Part B.

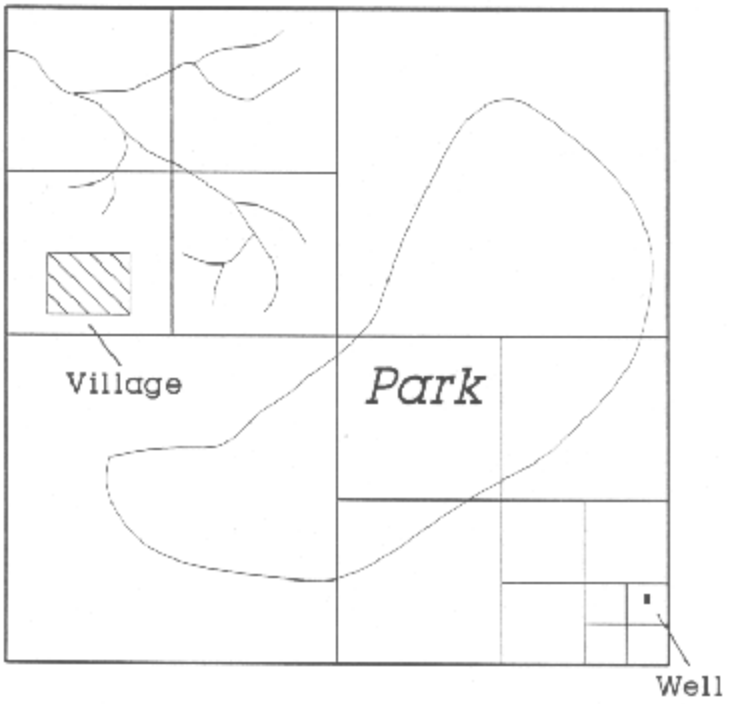


Part C.


2
3
0312
032
033
0313
02
030
0311
B
1
0310
00
01


MAP
0
1
2
3
A
A
A
00
01
02
03
A
A
A
030
031
032
033
A
A
A
0310
0311
0312
0313
A
B
A
A

Part D.



	Index
Park	NULL
River	2
Village	20
Well	1113

Figure 2.