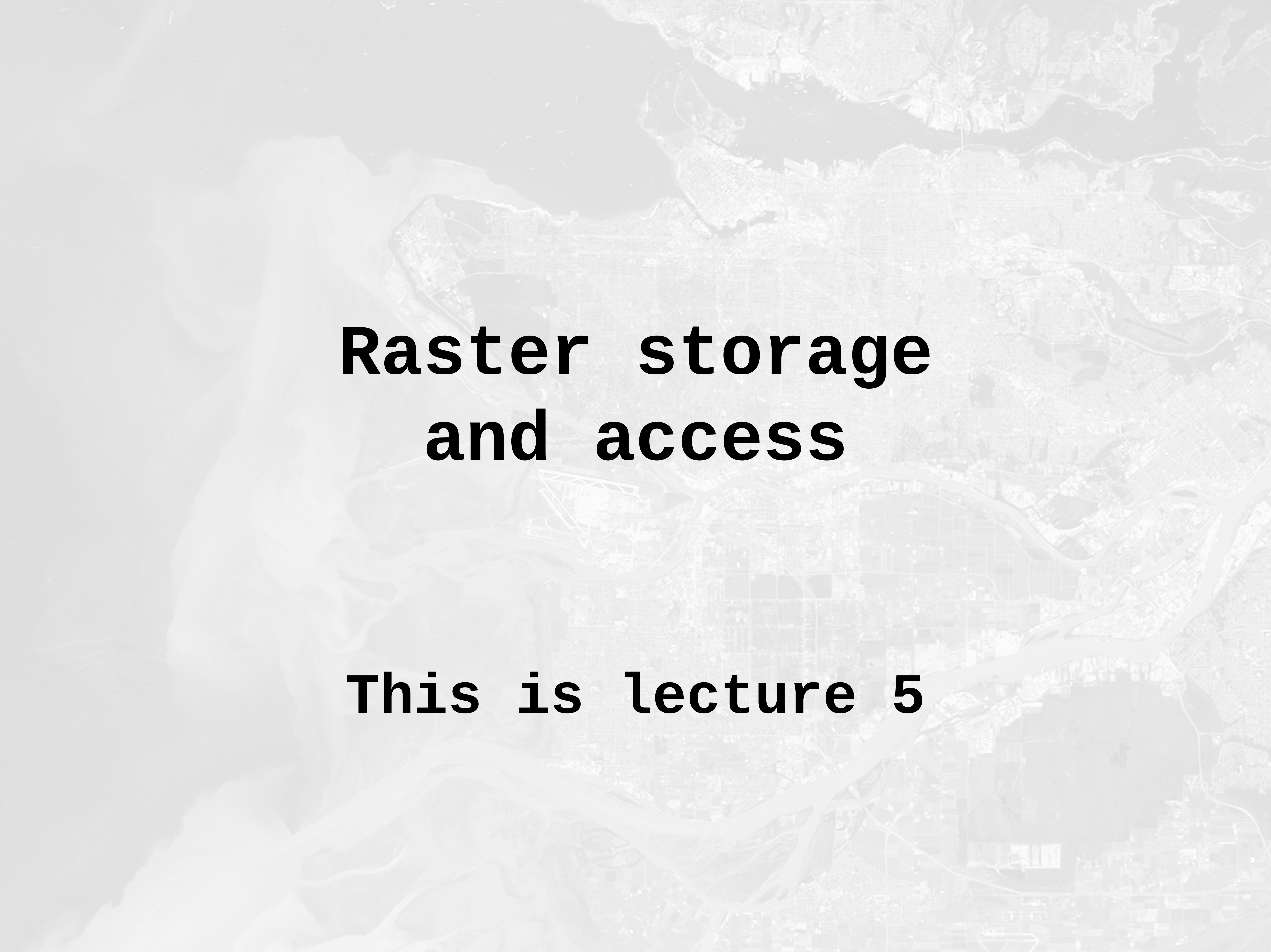


Raster storage and access

This is lecture 5

Definition of raster space

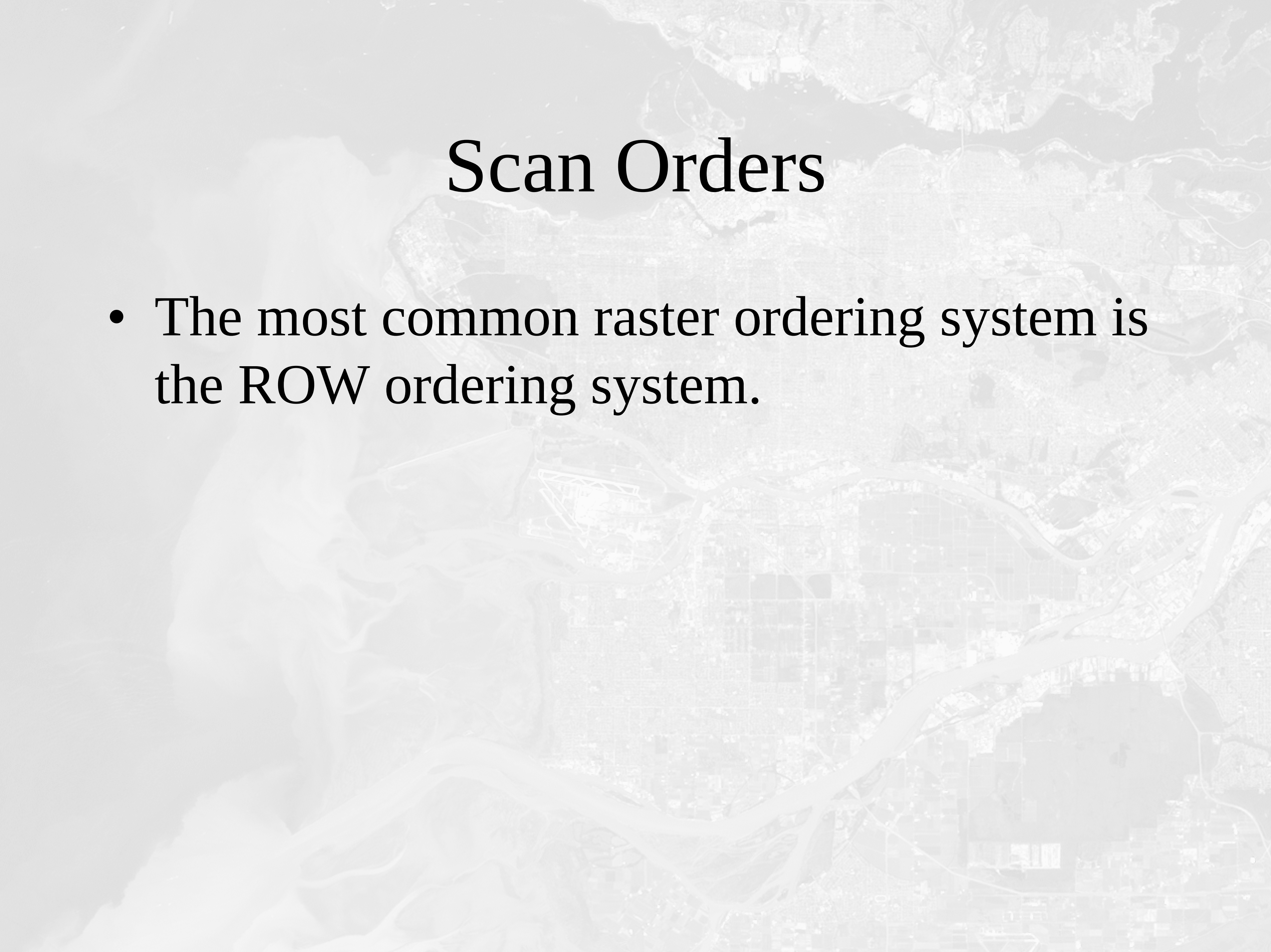
- Think of the raster plane as a set of tiles. There are many ways to navigate through them.
- These techniques can be expressed mm through a function.

Take a section of the Euclidean plane P where P is the unit square [0,1] x [0,1]. S is the real interval [0.1]. We are looking for functions *f* such that *f*: P$\rightarrow$ S has the property that , for all *x*,*y* that are $\in$ of P, *x* is near *y* in the region IFF *f*(*x*) is near *f*(*y*).

Translation: We want to find mathematical functions to apply to the Cartesian plane so that the search order or compression order stores near things, near to each other. These access route maps are often called SCAN ORDERS.

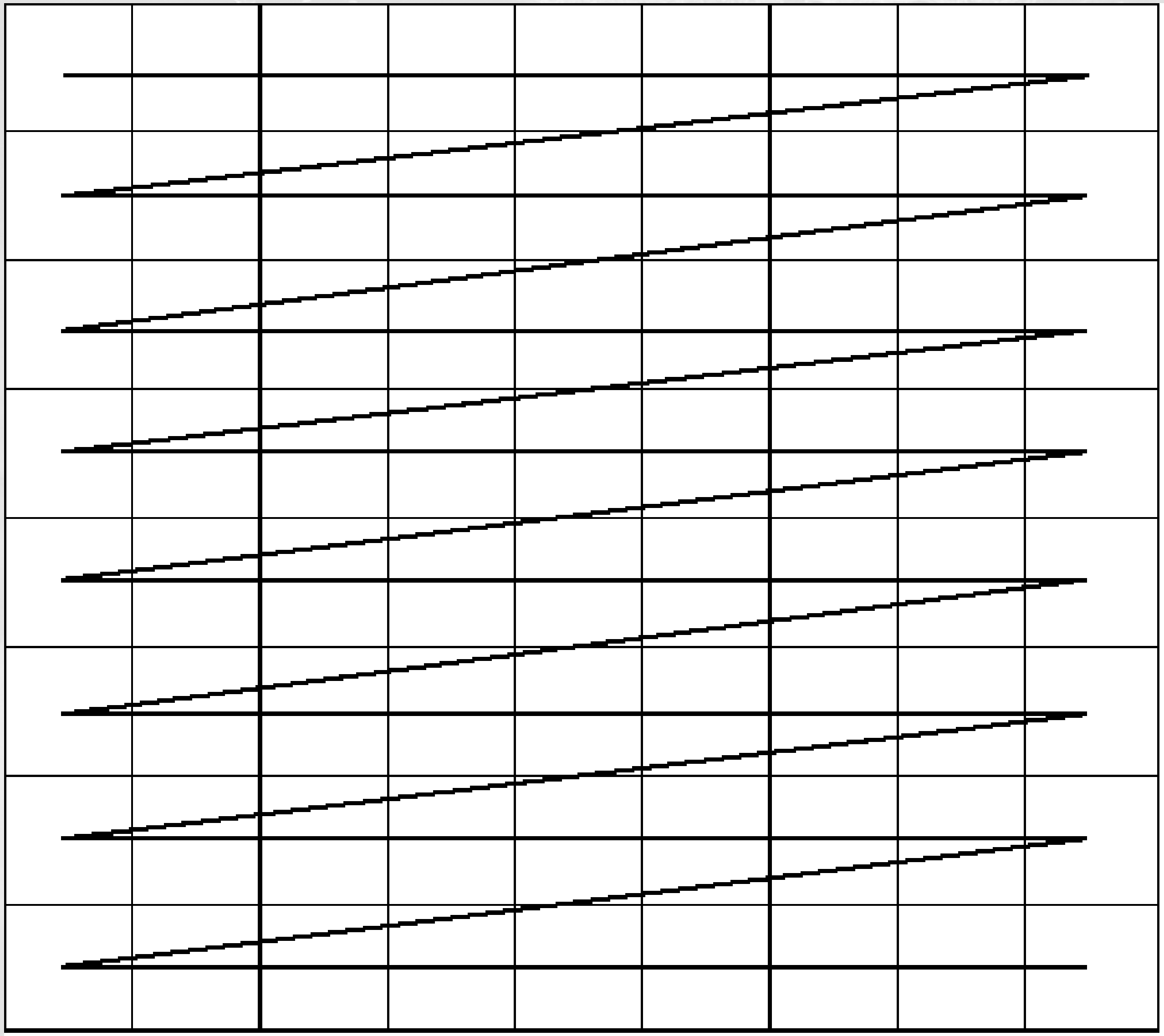
Raster access

- There are great computational inefficiencies associated with direct, uncompressed raster storage.
- These inefficiencies lead to problems with search efficiency, especially since the data is 2-D rather than 1-D.



Scan Orders

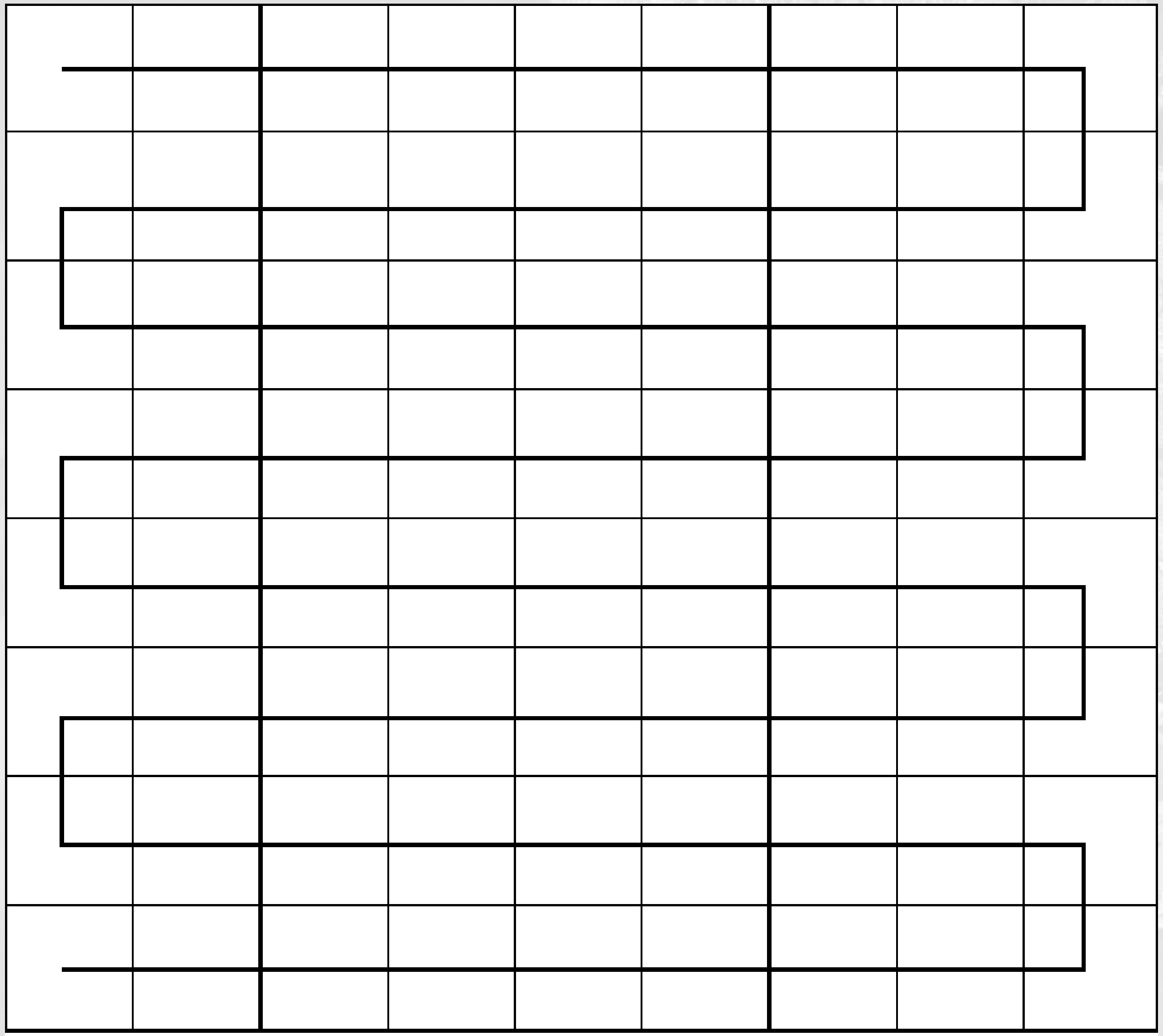
- The most common raster ordering system is the ROW ordering system.



Row ordering. This is the simplest method of ordering (and searching) for raster grids. Its weakness is that the near points at row ends are not stored close to each other

A partial fix …

- Row-prime ordering fixes half the problem



Row prime ordering. This is a modification of row ordering that solves the problem -- for half the endpoints.

More sophisticated compression

- Morton and Peano-Hilbert are a little more sophisticated. The Morton system uses a Z-pattern to scan the raster plane.

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

Morton. This is recursive system for indexing the pixels. The Z pattern is repeated for the first 4 pixels (starting from the bottom left). That pattern is repeated 3 more times to build a block of sixteen raster cells. Then the process repeats itself.

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

Fill in the rest:

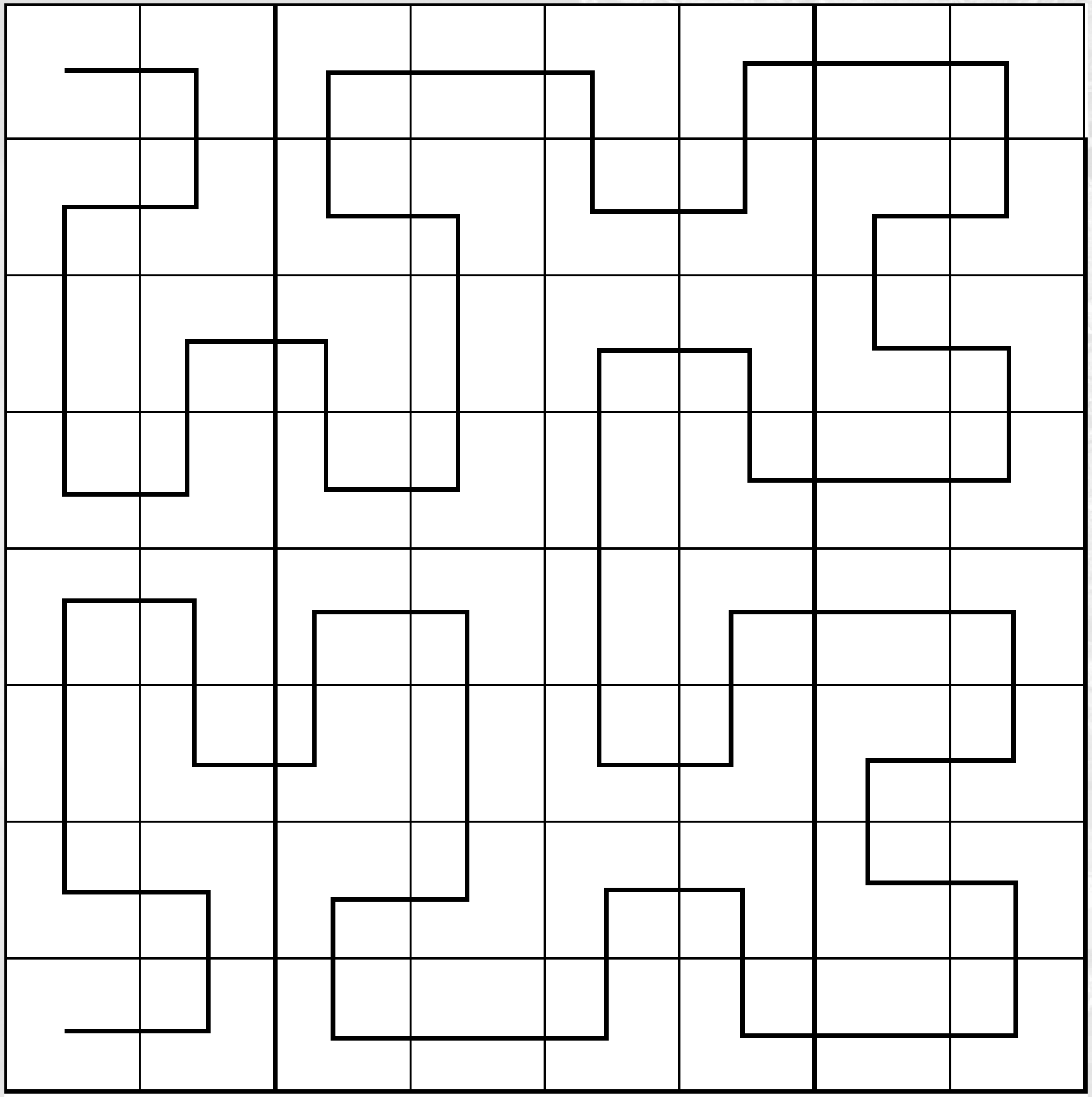
After 4 x 4 = 16 Z-shaped patterns through the raster, you start again with another 16 block.

Peano Hilbert Ordering

- Close to Morton ordering, but solves problem of switching to different 4 x 4 blocks.
- Switching is contiguous.

21	22	25	26				
20	23	24	27				
19	18	29	28				
16	17	30	31				
15	12	11	10				
14	13	8	9				
1	2	7	6				
0	3	4	5				

Peano-Hilbert. This is another recursive system for indexing raster cells. It is arguably more efficient as maximum contingity is maintained.



Peano-Hilbert. The two-dimensional scan order.

Scan ordering versus compression

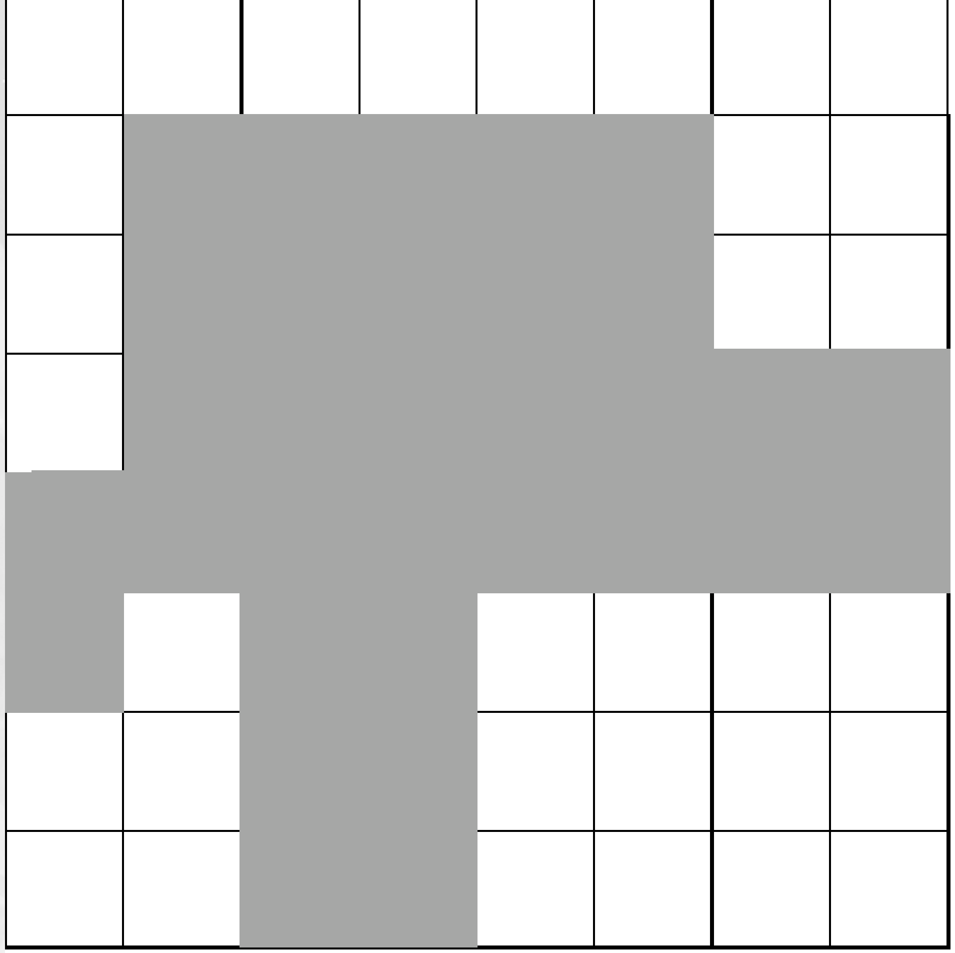
- The two are not synonymous. Do not confuse.
- 2-D ordering at the array level still requires compression to save storage space.

Raster compression structures

- Most efficient way to store raster data is to store cells with like attributes as one big area (with no differentiation between cells for that area.)
- Compression techniques are developed to encode cells as one unit.

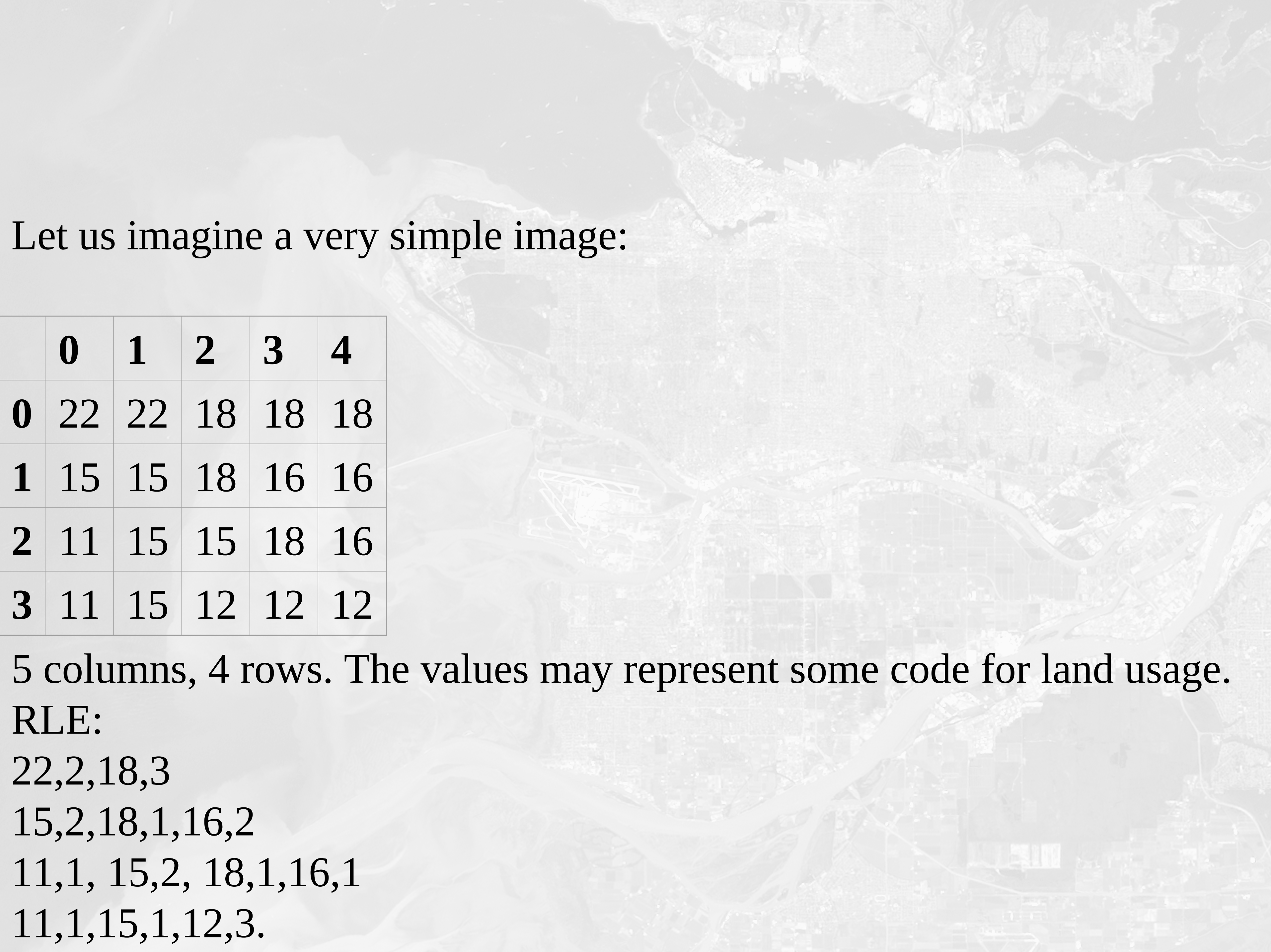
Run-length encoding

- Simplest storage method.
- Start with left-most cell in each row, and count numbers of like values.
- Several different conventions for run-length encoding.
- Simplest is for layers with one attribute per layer.



Run length encoding. The simplest compression techique but can also require significant processing power to decompress.

row 1:	row 5: 1,8
row 2: 2,5	row 6: 1,1 3,4
row 3: 2, 5	row 7: 3,4
row 4, 2,7	row 8, 3,4



Let us imagine a very simple image:

	0	**1**	**2**	**3**	**4**
0	22	22	18	18	18
1	15	15	18	16	16
2	11	15	15	18	16
3	11	15	12	12	12

5 columns, 4 rows. The values may represent some code for land usage.

RLE:

22,2,18,3

15,2,18,1,16,2

11,1, 15,2, 18,1,16,1

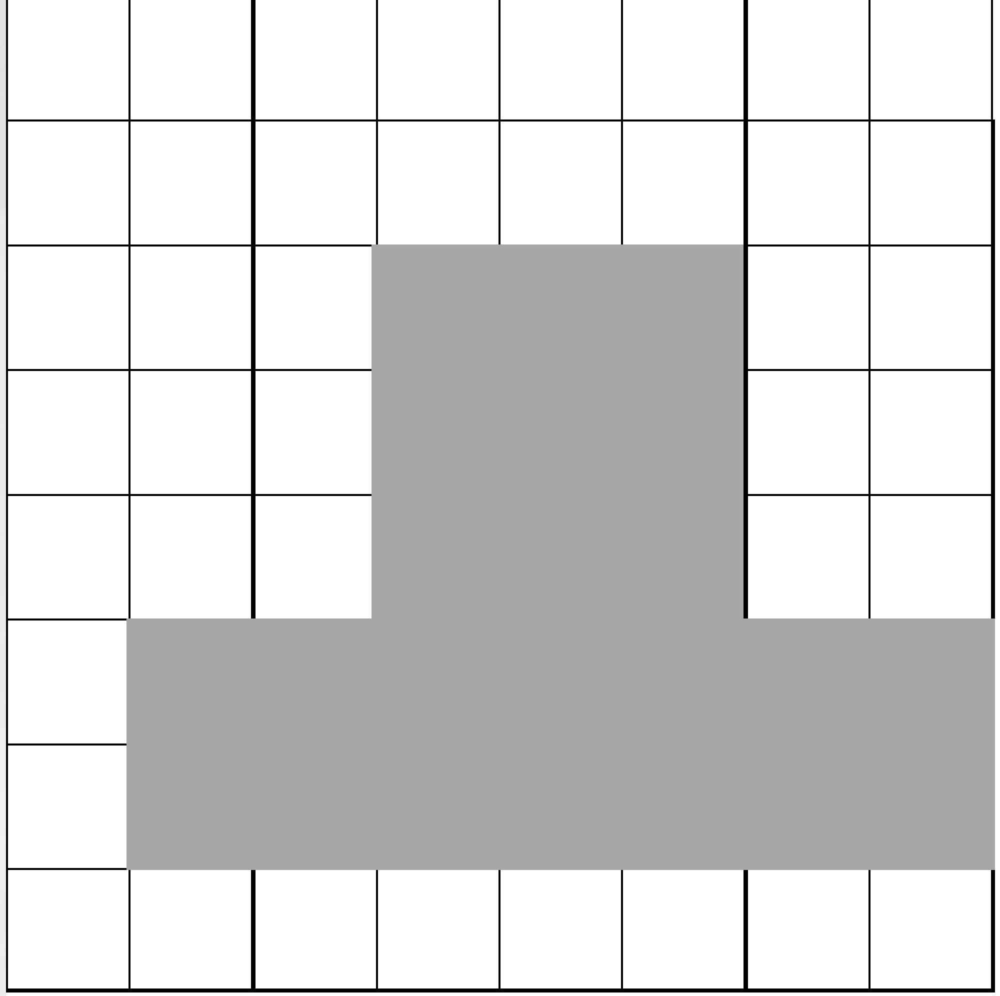
11,1,15,1,12,3.

The table shows IDRISI's datatypes

	memory required	range	compression
byte	1 byte	0 to 255	yes
integer	2 bytes	-32768 to +32767	yes
real	IEEE 4 bytes	$\pm 1*10^{38}$, 7 significant figures precision	no

Chain codes

- Chain codes compress the raster boundary of a region by starting in the SW corner, and counting the number of cells in each direction.
- When Chain coding is stored digitally, the alphabet characters for direction are assigned numbers (E = 0, N = 1, W = 2, S = 3).



Chain encoding. A compact method of storage because it relies on integer digits (i.e. no real numbers with their attendant storage requirements) but has other limitations.

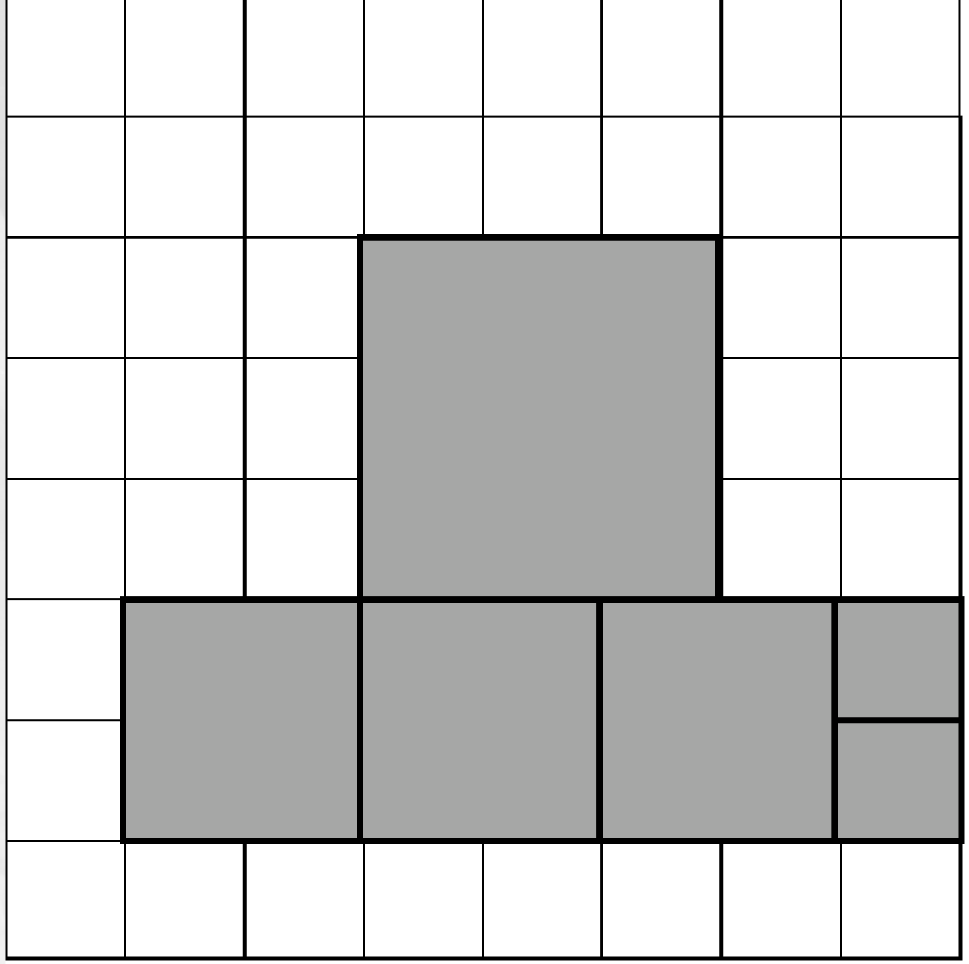
Starting from the SW corner of the shape: N2, E2 N3, E3, S3, E2, W7.

Pros of chain codes

- They are compact
- Good for area and perimeter estimates
- Good for converting raster to vector (WHY?)

Block codes

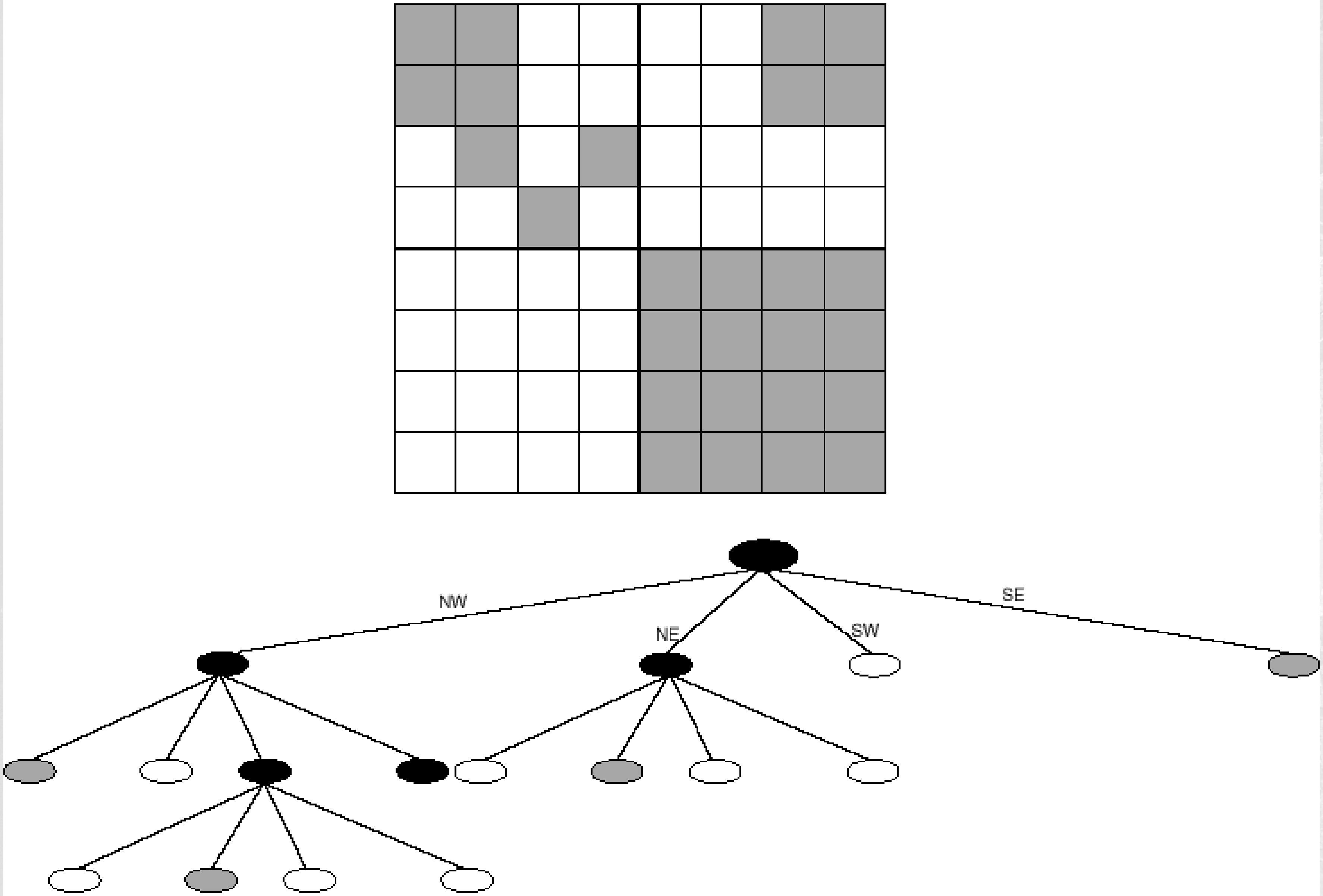
- 2-D equivalent of RLE
- Uses a technique called *medial axis transformation* (MAT) to create a data structure that stores just 3 numbers for each homogeneous block of an attribute layer.
- The 3 numbers are the origin (x,y) and the radius of each square.
- The origin can be the center cell or the bottom left cell.



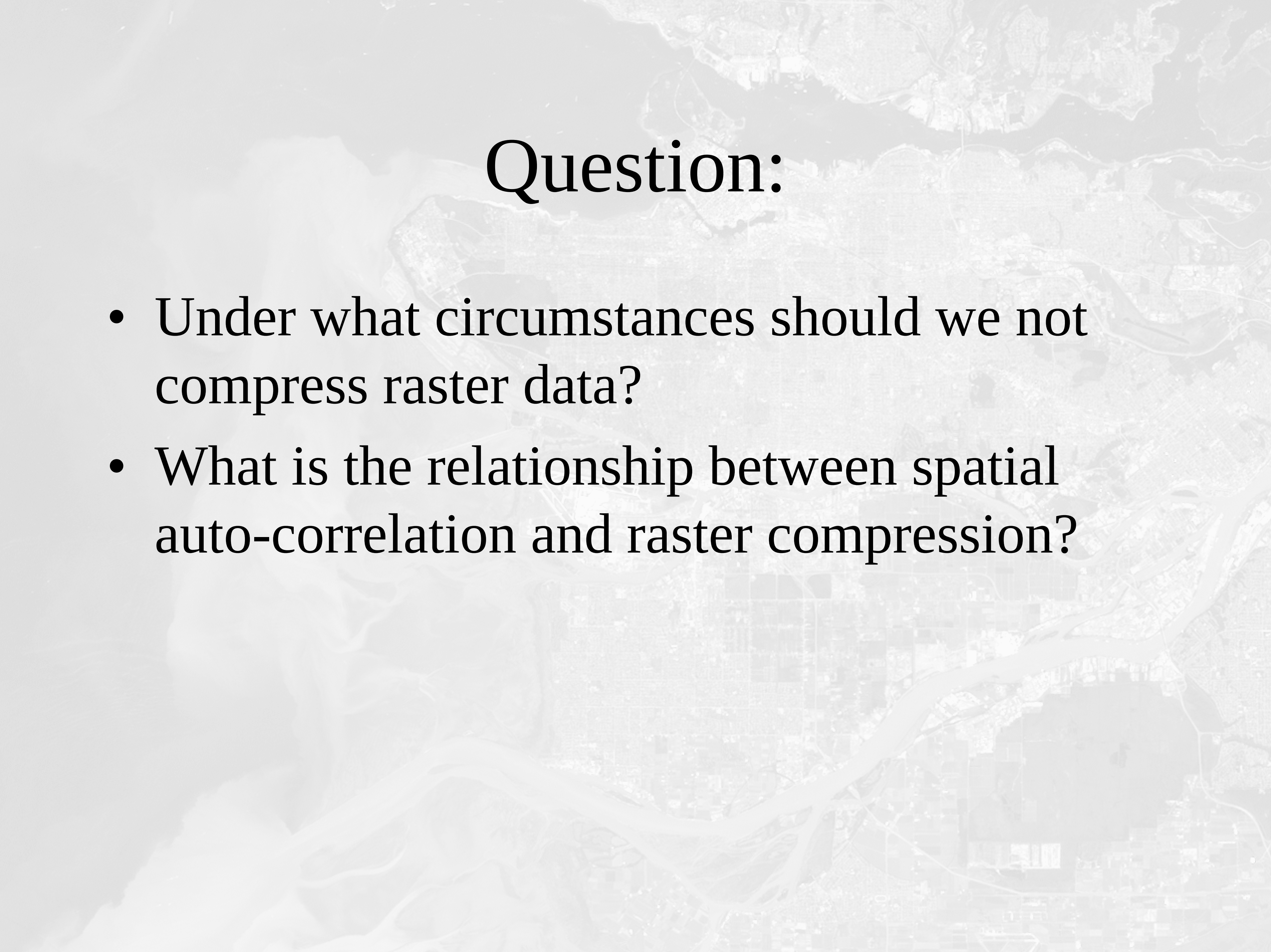
Block cells. The 2-D equivalent of run length encoding. The attribute layer is divided into squares that are stored in a data structure, using the origin and radius of each.
Here the area is divided into 5 blocks, each described by 3 numbers (total = 15). Without compression, 23 numbers would be required.

Quadtrees

- An efficient form of raster storage (compression) because they are *on demand.*
- Only areas that are not homogeneous are decomposed.



The quadtree decomposition of the raster area.



Question:

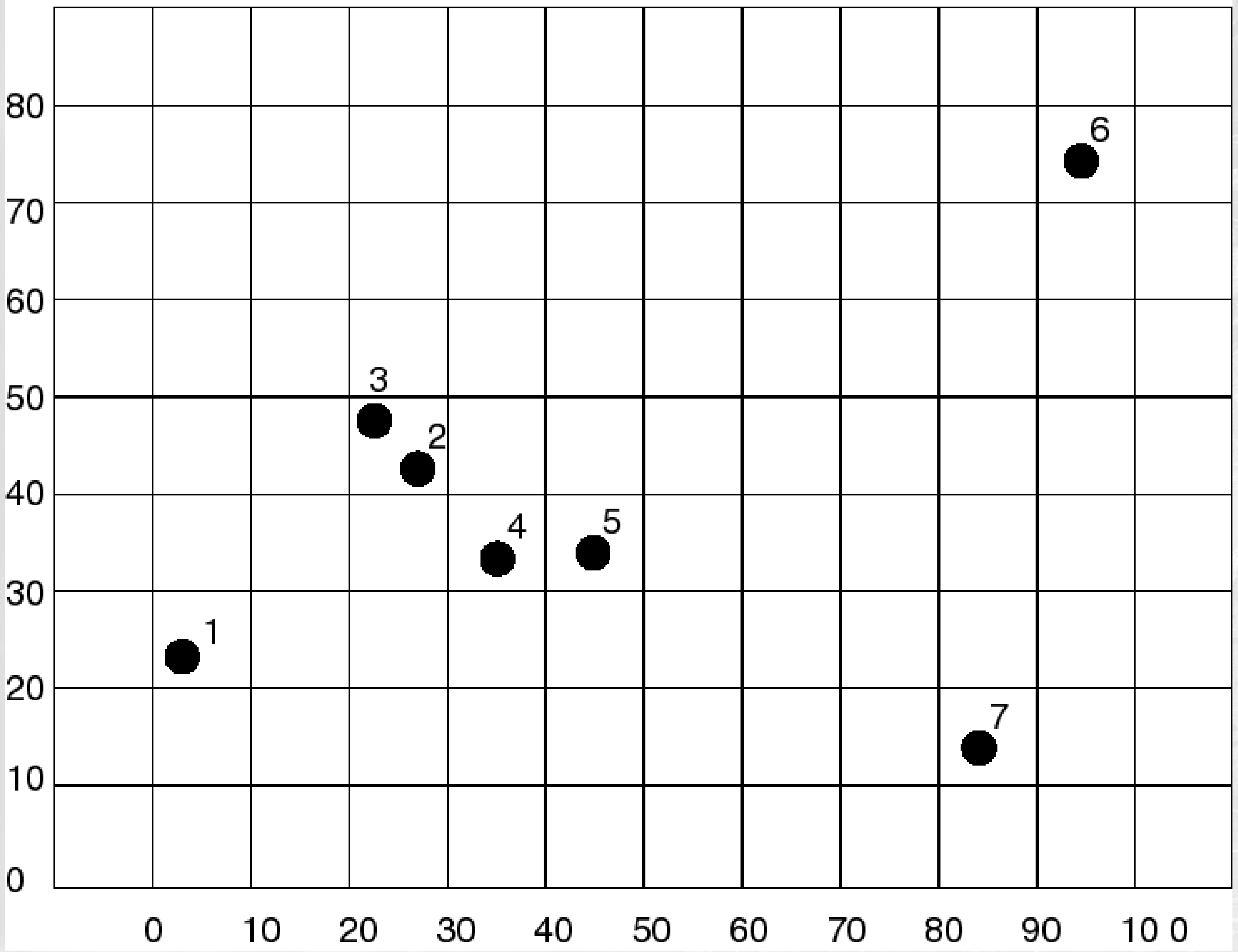
- Under what circumstances should we not compress raster data?
- What is the relationship between spatial auto-correlation and raster compression?

Three types of raster search

1. Non-spatial query: Retrieve the point location of a geographic feature.
2. Point query: Retrieve the feature (site) at a specific location.
3. Range query: retrieve any sites (features) within a bounded area.

Each type of search is executed differently depending level and type of compression and scan order.

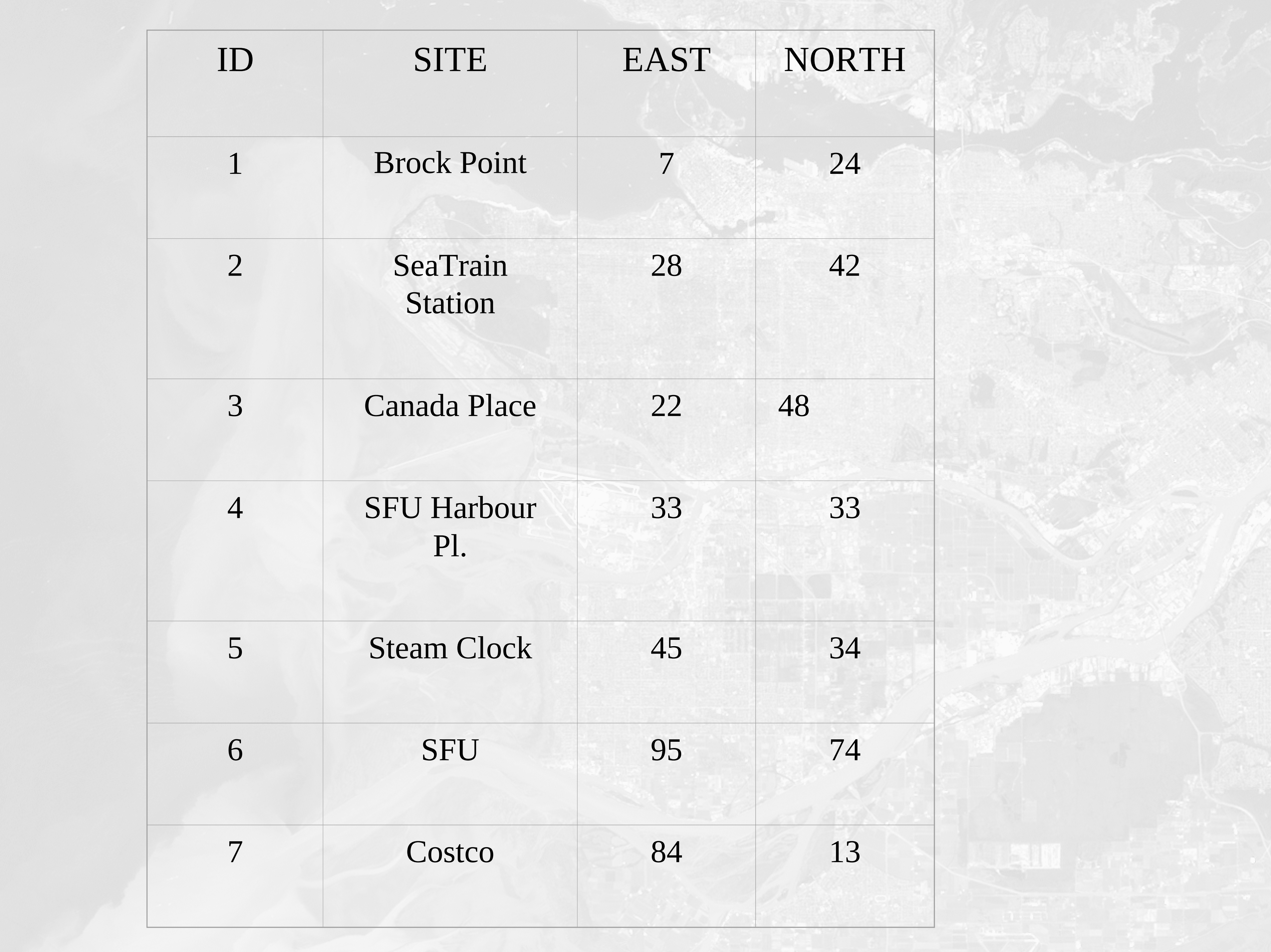
But principles remain the same from an algorithmic perspective.



0
10
20
30
40
50
60
70
80
0
10
20
30
40
50
60
70
80
90
10 0
1
2
3
4
5
6
7

Query one: (non-spatial query): Retrieve the point location of Brock Point. [Point 1 = (7,24)]

- Query two (point query): Retrieve the site at location (22,48) [Point 3=Canada Place]
- Query three (range query) Retrieve any site in the rectangle with south-west and north-east vertices (30,30) and (50,50) respectively. [points 4 and 5 = SFU downtown and steam clock]



ID	SITE	EAST	NORTH
1	Brock Point	7	24
2	SeaTrain Station	28	42
3	Canada Place	22	48
4	SFU Harbour Pl.	33	33
5	Steam Clock	45	34
6	SFU	95	74
7	Costco	84	13

- For the first query, we just use general purpose access methods.
- The application refers to the table and searches until it finds find Brock Point and return the location (7,24).
- How the search was conducted would depend on whether it was simply linear or structured.

Second query

```
STEP 1: Open SITES File
STEP 2: while there are records left to examine, do
    steps 3.1 to 3.4
        3.1 get the next record
        3.2 if the value of the first coordinate field is 22
        3.3 then                    if the value of the second
    coordinate field is 48
        3.4                 then return the site name from the
    record.
```

Third query

```
STEP 1: Open SITES File
STEP 2:  While there are records to examine, do steps 3.1 to 3.4
        3.1 get the next record
        3.2 if the value of the first coordinate field is in the range (30,30)
        3.3 then                    if the value of the second coordinate field is in the range (50,50)
        3.4                    then retrieve site name from the record.
```