


Geographic Information Systems
Prof. A. K. Saraf
Department of Earth Sciences
Indian Institute of Technology – Roorkee

Lecture - 11
Raster Data Compression Techniques - 01

Hello everyone! and welcome to new discussion. Today we are going to discuss raster data compression techniques. This is in 2 parts. So, first we will take Part 1. Please recall the discussion which we had related with vector data compression techniques. Though, vector data does not provide that much compression because of less redundancy. But in case of raster data, we get lot of redundancy.

(Refer Slide Time: 00:53)



- Organizing the space in the car could be compared to the data compression problem!
- Large amounts of data can create enormous problems in **storage space** and **transmission time**.

http://www.gita.info/DataCompression/html/unt_01/compintro.html

IT Roorkee | NPTEL ONLINE CERTIFICATION COURSE | 2

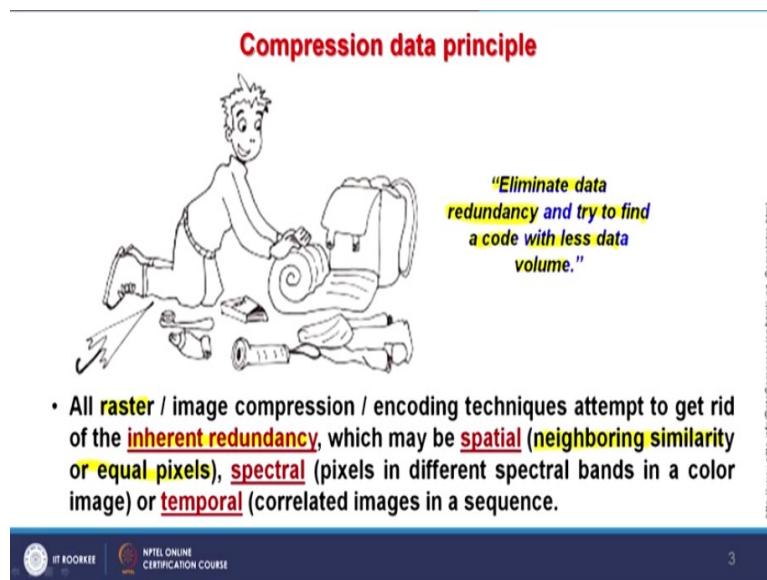
This is just one example that how you know, basically what we are trying. We are trying to accommodate as much as data possible within whatever the space is available or in minimum space. Not optimum space, but in minimum space. Like example here in a small car so much of goods are being you know planned to keep. That means things would be compressed there.

And the data compression is the same problem as problem with the small car and lot of goods are there or stuff is there. Now, you know that why we want to compress. The 2 reasons, first is this storage space issue is there because we want to reduce the storage space so that lot of space is available for other storage or other works.

And secondly also during the transmission from one end to another end through internet, we would like to reduce the size of the data and that is why the data compression is required. Also, there are some compression techniques especially associated with raster data compression which you know compress the data in a manner that even in the compressed form, you can use them.

And we will see some examples of that also. So, this is a big field now about the data compression and lot of options are available and lot of you know copyright protected options are also available. So details about those may not be available with us but to some extent, we can try to understand what is that algorithm or approach and what kind of compression these tools provide to us.

(Refer Slide Time: 02:57)



Compression data principle

"Eliminate data redundancy and try to find a code with less data volume."

- All raster / image compression / encoding techniques attempt to get rid of the **inherent redundancy**, which may be **spatial** (neighboring similarity or equal pixels), **spectral** (pixels in different spectral bands in a color image) or **temporal** (correlated images in a sequence).

IT ROOKEE | NPTEL ONLINE CERTIFICATION COURSE | 3

So, the same thing here that one is trying to you know keep as much as stuff inside a bag here. Basically, the purpose here is to eliminate data redundancy. The data redundancy; there is you know if I talk about the raster data, if there are lot of pixels or cells are having almost same value or same value then by some means if we can code them in a manner that we do not have to store for each cell or each pixel rather than a group of cells or pixels can be stored with the one code then we can achieve good compression.

So, basically data compression eliminates redundancy and try to find a code. And there are various ways of doing this thing with less data volume. Ultimate aim is to reduce the size of the file so that it can be stored in the system and as well as it can be transmitted through internet by either email or some other ways. So, that is the basic purpose here.

Now about this point which we will have, whether all data compression techniques are you know constructive or lossless or there are some techniques which can create a destructive effect on the quality of the data because nobody would like to compromise with the quality of the data. But there are techniques and very popular techniques are also there.

The data compression which can be achieved as we have been discussing in a higher way by applying data compression to the raster data. Vector data inherently do not have redundancy. Whereas raster data inherently having redundancy. Especially if I talk about satellite images or if I talk about digital elevation models that is the raster in grid form for flat area or less undulated area then this issue or this advantage of data compression can be achieved.

So basically, what it is done is that in the surroundings, in neighboring similarity is searched. And equal pixels of cells mean they are having the same value and then they are stored in a manner giving a code and they are stored in a manner that they will require less space. So obviously, we are discussing now the raster data compression techniques as we know that raster data is stored in terms of rows and columns.

(Refer Slide Time: 05:52)

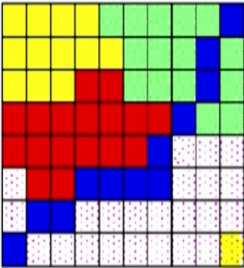
Raster



Stores images as rows and columns of numbers with a Digital Value/Number (DN) for each cell

Units are usually represented as square grid cells that are uniform in size

Data is classified as "continuous" (such as in an image), or "thematic" (where each cell denotes a feature type).

Numerous data formats (TIFF, GIF, ERDAS.img etc)



  4

And each cell here represents a either pixel or cell depending on I am using a grid or image respectively. And as you also know that units in the raster are always a square and all are uniform in size throughout the data file. And there are various image formats are also there. And also, we know that raster is a continuous data. And that is why there are high chances of getting redundancy.

Various formats which are known to us like Tagged Information File Format or in short, we write TIF or TIFF file we also called. This GIF file which is a Graphic Interchange Format; file format and in short, we write GIF. And then some you know, software's specific like ERDAS is a digital image processing software. So, they are having their own format and that is called .img format or imaging format.

Though most of the good software's; GIS or digital image processing software's because the raster data or data compression technique is common in both the fields; in digital image processing as well as in GIS. So, most of these software will allow you to handle such data either TIFF, GIF, img or many other raster data formats and will also allow you to save as from one format to another.

But whenever you do this exercise or a step or this operation, make sure that the whatever the new format which you are choosing should not be you know lossy format or destructive format. It should be always lossless format. Because we should know generally in a real serious scientific application, we do not want to compromise on the quality of the data.

(Refer Slide Time: 08:03)

- Raster data tends to be "*spatially auto correlated*", meaning that objects which are close to each other tend to have similar pixel / cell values.
- "*All things are related, but nearby things are more related than distant things*" (Tobler,1970).
- Because of this principle, we expect neighboring pixels to have similar values.
- Therefore, instead of repeating pixel values, we can code the raster as pairs of numbers.

http://www.gita.info/DataCompression/html/rastercomp_chain.html

As you know that raster data are spatially auto correlated. That means many neighboring pixels or cells can have same or similar pixel or cell values. And all things as we know are related. But nearby things are more related than distant things. This is a Tobler theory. In 1970, he has given that there are chances that whoever is nearest get influenced maximum than a far distance.

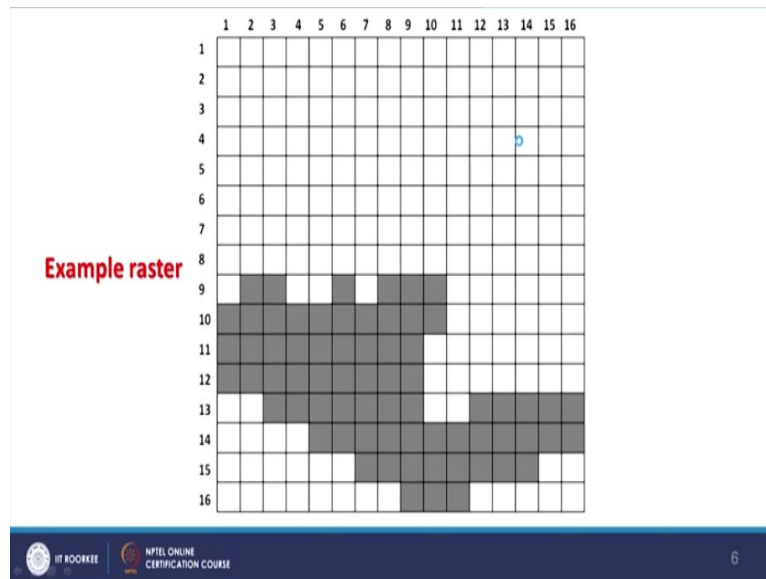
And this is now also seen in the corona pandemic that we are advised that we should keep at least you know 6 feet or 2-meter distance between 2 persons. So that we do not get infection. So, if we keep the distance, the influence would be little less or really less in case of corona. So, the same thing here that all things are related but nearby things or neighboring things are more related than distance things.

And keeping this principle in mind, we expect that neighboring pixels to have similar values. And especially as I have said if we are handling satellite images of a snow covered or a desert terrain then these are the areas where sometimes you know featureless; we say featureless because featureless why because the values of pixels are having almost same value.

And therefore, we can compress these images or digital elevation models which are having less variations in the elevation values or no variations. And we can compress them very quite significantly. So repeating pixel values and then code is given and a pair of numbers. Now I am in order to explain different basic techniques. I am not going for some other techniques which of some are even have copyright. So, we do not know much details.

So, 4 techniques, I am going to talk. 4 basic techniques which have been implemented in GIS or digital image processing software's. And a variant of these techniques is also seen here and there in different software's. So, for simplicity what I am going to do? I am going to take a very binary image. And just for our understanding, the grid or these rows and columns are shown as a you know mesh.

(Refer Slide Time: 10:44)

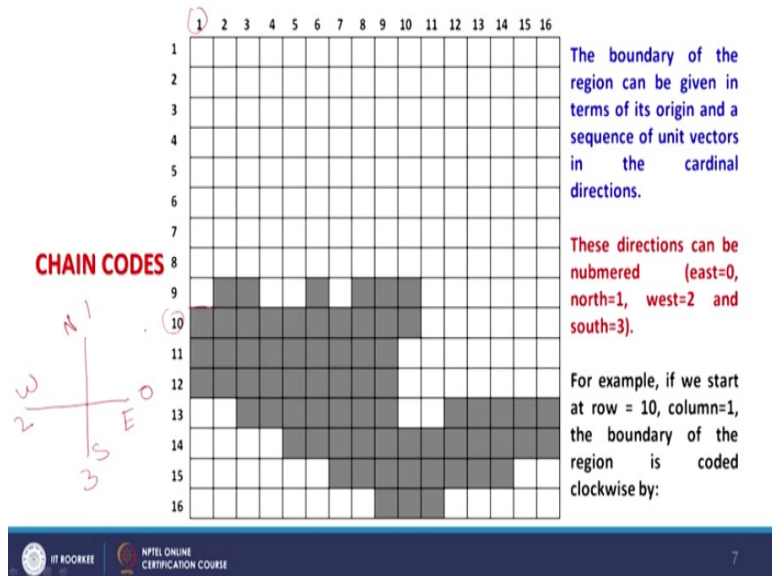


But you know really that raster data is stored just in a two-dimensional matrix and no such boundaries are there or cell boundaries are there but for just to develop understanding about raster data compression, I am using this grid to show or this wire mesh to show how compression can be done using different concepts.

2 areas because a binary image so, one is white areas and one is the dark area. And we say the dark area or black area is having 0 value, white area having value 1. And this is since binary so, only 2 values are possible. So, when we observe this image, what we see that there are 2 distinct areas. And therefore, rather than coding the large area, our example compression techniques will try to code the smaller area.

And just reemphasize that point that to develop our understanding about the data compression specially raster data compression. I am taking the simplest possible example. There may be complications later on or in real data compression techniques. But first we have to understand the fundamentals.

(Refer Slide Time: 12:14)



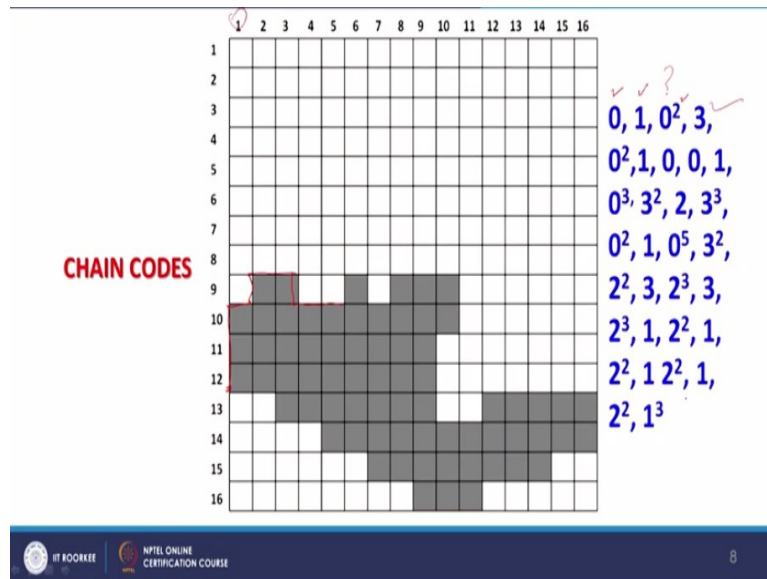
So, if I talk the first technique which is the chain codes technique in which the same dark area will be coded. And how it is coded that let us discuss this part. So, in this one, the boundary of the region; this boundary which is marked here, the black boundary will be coded and can be given in terms of its origin from where it starts and then unit of vectors and a sequence and that to in cardinal direction.

Cardinal direction means here in north, south, east, west. So, what these directions can be, here different approach has been followed. This is just code, do not take as a number. It is a code basically. So, for east direction, the code is 0. For north direction it is anti clockwise. So, if I draw here then for east is 0, this is 1 and for west 2 and for south you know 3. So likewise, these codes have been assigned here.

Now, if somebody is writing their own program, we can give code say for north 1 or for north 0 does not matter. You know accordingly things will be done. Just for you know, we have to code. So, these values have been given. And I have said that these are just not numbers but these are codes though they are being represented in form of numbers. So, if we start for coding this black area or dark gray area then we start at row number 10.

Why we start at row number 10 because that is the cell of this binary image from where we are going to code this one like this. So, column 1 and this is column 1, this is row 10. And then region is coded clockwise. Then we will keep moving like this and accordingly. So, let us see how we can code this.

(Refer Slide Time: 14:39)



So, like this exercise has been done for this image and as you can see that these are the code, not a number on you cannot perform numeric value. Otherwise, one would say that how you can have 0 power 2. So, this is code. 0 is the code. So, we start with a 0. That means we are moving now towards the east direction; 0 for east then 1 cell towards the north, so 0.

Then 1 cell towards the north then 2 cells towards the east. Then 2 cells towards the east then 1 cell towards the south, so, 1 cell towards the south and then again 2 cells towards the east. And likewise, we come at the end and then we say 1 cell here and then this row number 1 here and then we are going 3 cells. And therefore, the entire polygon in which the black cells are having 0 values are coded.

And automatically the other region is also coded and by which instead of storing you know values for each cell and large area is having uniformity and therefore, redundancy in this particular image which can be coded by only giving these values. So, this way we can do the chain codes. Now, there is another coding which is little variation from this chain code which is called run length coding or RLE; It is also called RLE.

(Refer Slide Time: 16:24)

Run length coding / encoding / RLE (lossless)

- The run length coding is a widely used compression technique for raster data.
- The primary data elements are pairs of values or tuples, consisting of a pixel value and a repetition count which specifies the number of pixels in the run.

http://www.gta.info/Data-Compression.html#rastercomp_ch4an.html

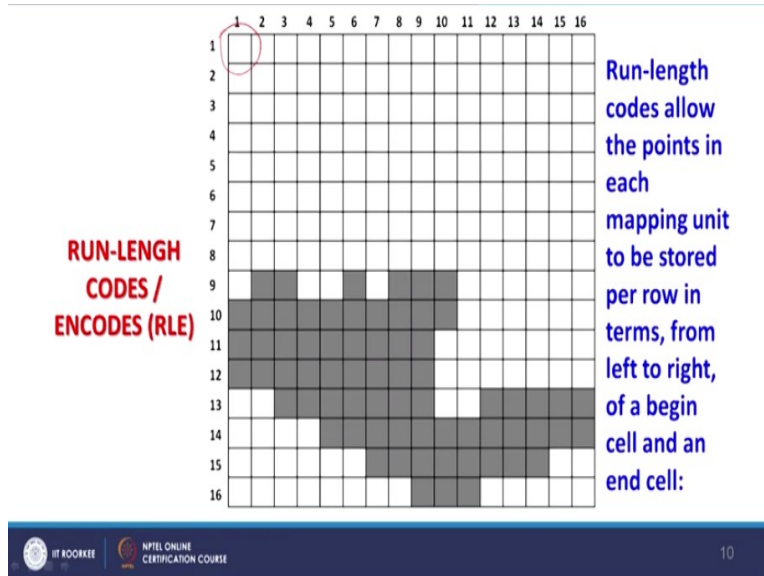


And this is lossless; nondestructive coding technique. And sometimes people write in literature either coding or encoding, is almost same thing. So, I have mentioned both things here. Now what is in run length coding is a widely used compression techniques and many software's like ERDAS or ArcGIS or many software's, you would see they will mention in this RLE way.

And they will mention in some form RLE rather than run length encoding. So, does not matter. And then the primary data elements are pairs of values or tuples, consisting of a pixel value and a repetition count which is specify the number of pixels in the run. Because here we are running along the boundary that is why it is called run length and counting the length along the boundary.

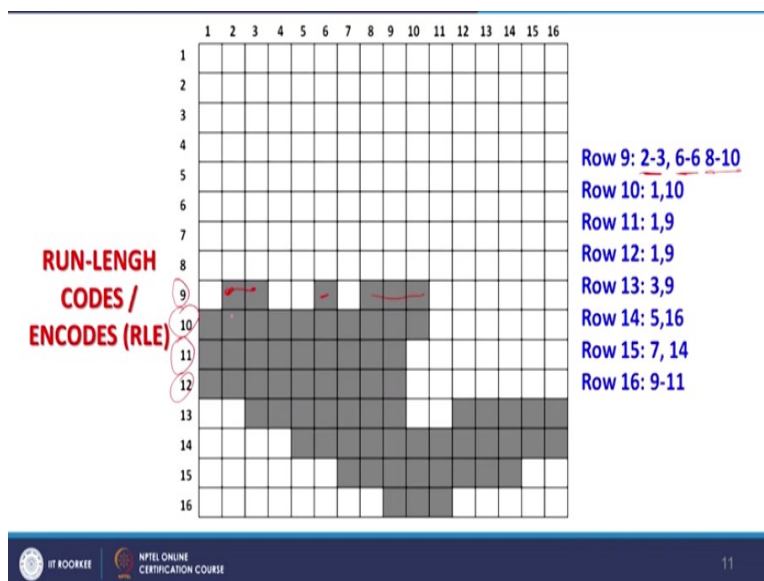
So, data are built by reading successfully rows by row through the raster data input image or input data creating a new tuple every time the pixel values changes or the end of the row is reached by which we can encode using this method. So, this very good again run length coding depending on the redundancy either in the grid or in an image. So, we will see.

(Refer Slide Time: 18:09)



So run length codes, this RLE for the same image if we run and this is how it is going to have that it allows the points in each mapping unit and mapping units are over here; the cell this one like this individual cell mapping units and these are stored per row. So, row by row that is why run length. So, running along the length of the row and from left of course, to right of a begin cell and then end cell and likewise coding will be performed.

(Refer Slide Time: 18:44)



So, if I have to code so first row which is the 9 which is having these black cells. So, for a 2, 3 that means run between column number 2 to 3 then there is a gap because of white pixels. So, no coding will be done. Then 6-6, it is here; the length is just 1 cell. Then 8 to 10; 8 to 10 is here. And then it will go for the next row. Then another row till it reached to the last row of the data.

So, for 10; it is just simply 1 to 10. It is all you know black. And likewise, we can code the entire black area of this particular image.

(Refer Slide Time: 19:39)

Run length coding

Row 3: 2,3; 6,6; 8,10
Row 4: 1,10
Row 5: 1,9

Describes the interior of an area by run-lengths, instead of the boundary.

Codes - I:
 Row 1: 4X; 2W; 3R; 1X
 Row 2: 3X; 4W; 2R; 1Y
 Row 3: 2X; 6W; 2Y
 Row 4: 5W; 5Y

In multiple attribute case there are more options available.

Codes - II:
 X4,W6,R9,X10
 X3,W7,R9,Y10
 X2,W8,Y10
 W5,Y10

Codes - II: Recording end cell position within a row.

http://www.gdta.info/Data/Compression/html/rlecomp_ch04an.html

IIT ROORKEE NPTEL ONLINE CERTIFICATION COURSE 12

Now run length code for little complicated area. You know instead of having just binary image, if we are having some you know, a theme which is going to be coded like here the example is given in the bottom. Then in a simple run length code, it will be done like this. But here instead of 2 categories, we are having 4 categories. So, for each category, we have to provide one more code here.

So, the X, Y, R, W; these are you know, the different categories in this map and that information will also be stored. So, if I take the row 1 then 4X. So, first 4 are there and they will assign this category which is yellow color. Then 2W then 2W like this. Then 3R means 3 blue cells and so on so forth. So, it is not that only the binary but of course you know, complicated things can also be stored without much problem using this concept of run length encoding or RLE.

So, what we see basically the search is done first to find out the smaller area. So that larger areas should not be coded; smaller area should be coded. And that is what it is done. And if you are having multiple attributes within a map then also it is possible to do it. So, this recording of cells within each row, there is a code I and then there is a code II also that it can be done something like that also.

So, it is already whatever is written first it is row 1 then row 2 and this is last row. So, little different variations which you would find in all these compression techniques, when people implement they apply their own mind and innovations and sometimes may give better results. (Refer Slide Time: 21:56)

Run length coding

VALUE	LENGTH	ROW
A	16	0
A	16	1
A	14	2
D	2	2
A	10	3
D	4	3
A	10	4
D	4	4
A	10	5
D	4	5
A	10	6
D	4	6
A	10	7
D	4	7
A	10	8
D	4	8
A	10	9
D	4	9
A	10	10
D	4	10
A	10	11
D	4	11
A	10	12
D	4	12
A	10	13
D	4	13
A	10	14
D	4	14
A	10	15
D	4	15

Full raster coding 256 values

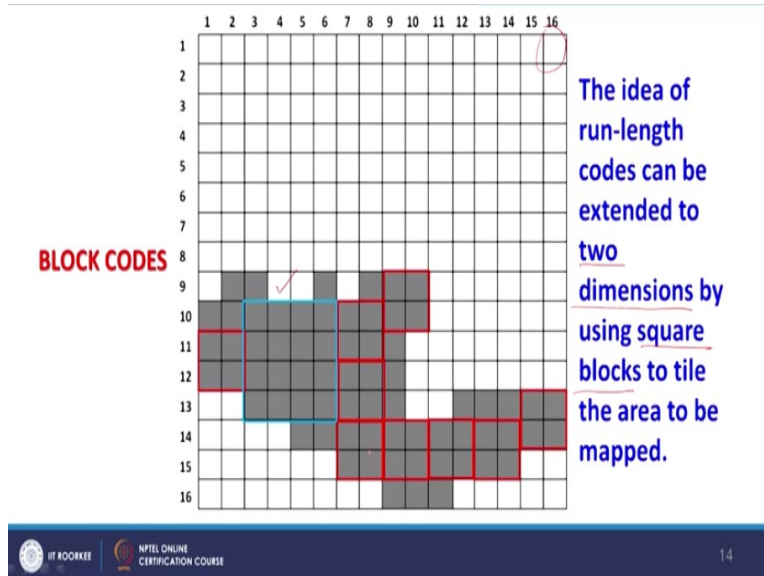
Run length coding 129 values

http://www.gita.info/DataCompression/html/rastercomp_chain.html

Another example of run length coding is given here that you can have you know, full coding for this entire and the table will store all that information also. So, here run length coding values are 129 values are there which instead of 256 values, if we store in this form raster. So, instead of that if we code in run length coding then we will end up only 129. And that means, we are getting rid of redundancy.

And further that also means that we are achieving compression. But this is a very small size image. Many times, we handle really huge size image. And therefore, you know reducing from 256 to 129, we can achieve a very high compression almost half. Now, the 3rd type of compression technique is called the block codes.

(Refer Slide Time: 22:57)

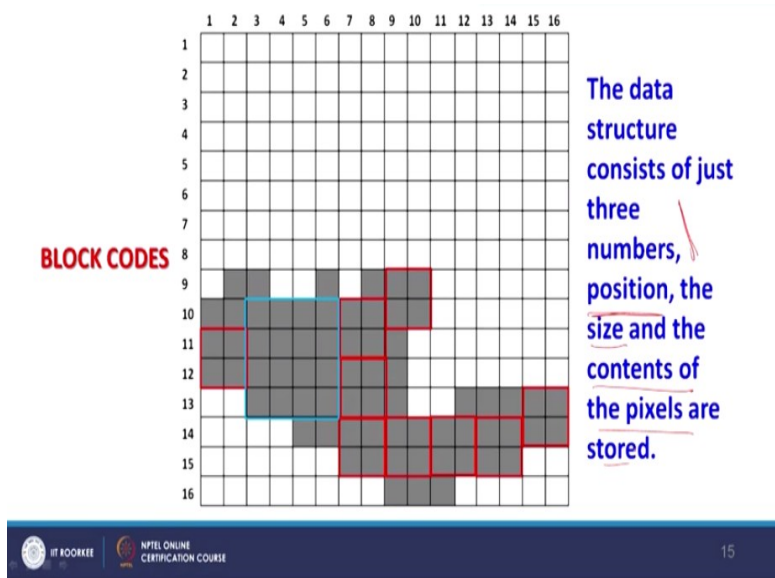


Block codes as name implies the blocks are first constructed which will contain values of cells which are uniform. And first largest block is coded and then second largest and till you reach to the unit level and unit here is our of course the cell. So, let us see that how it works. This the idea of run length codes as can be said instead of just you know these black codes that instead of going through only the in one direction that is along rows.

We can go in the now two dimension or two-dimension things by using a square block to area to be mapped. And as you can see here like this blue square which is in almost here in this part; blue square is 4 by 4 cells. That means 16 cells in one go can be coded. But in this binary image example that is the only you know block which can be created of 16 or 4 by 4 cells.

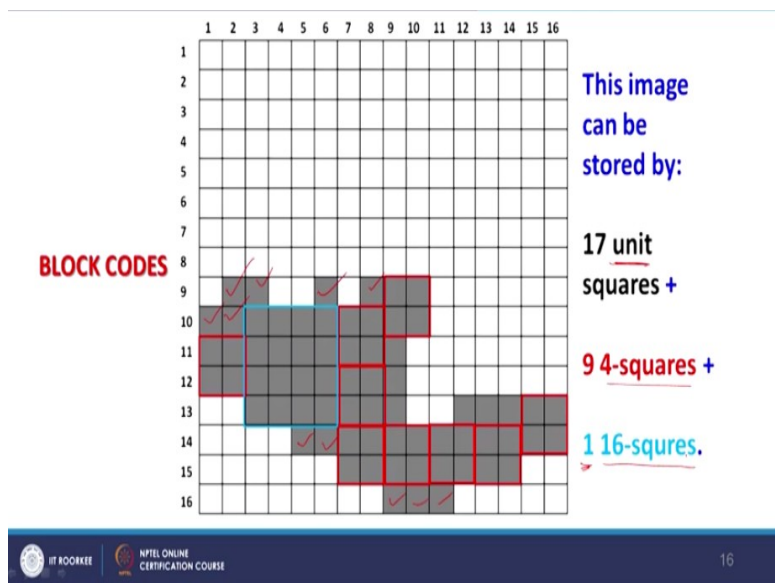
Then we have to go for the next size. And next size here is because it's a block, is square shape. That means next size can be 2 by 2. So, then you are seeing many red boxes are there and the next one would be of course unit level. So, then there are many boxes which are still remaining out of blue box and out of red blocks.

(Refer Slide Time: 24:42)



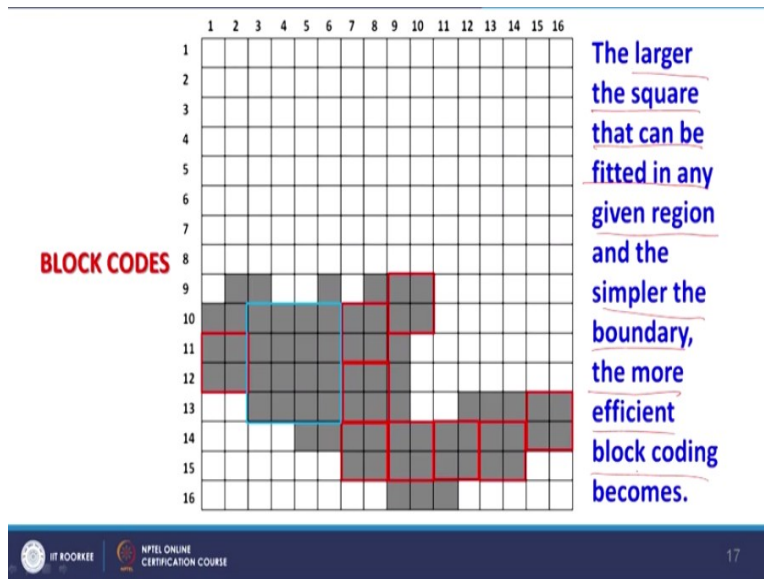
So, let us see how these will be coded. That the data structure consists just 3 numbers. What are those numbers? The position, the size and the content of the pixel are stored. Here in this case, all cells are having values say 0. This is binary. All white cells are having value 1.

(Refer Slide Time: 25:04)



So, in this one the image can be stored by 7 units which are unit level means single square. So, there are you know, sorry, 17. So, there are 17 such cells which can be stored here like this. And then there are nine 4 squares. 9 these red cells, there are total 9 of having 4 cells in each and then of course, 1. 1 which is having 16 squares or consists of 16 cells. And that is only 1 by which we can achieve quite good compression implying this technique which is called block codes.

(Refer Slide Time: 25:52)



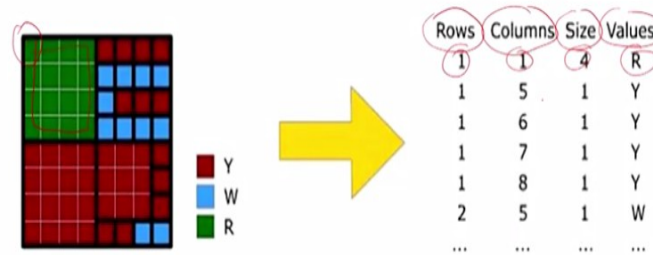
Here how you will achieve better efficiency or better compression if your input data is having lot of redundancy. That is almost true with everyone. So, the larger the square which can be created or fitted in any given region and the simpler boundary; the boundary of this shaded area in this one or dark area, more efficient block coding becomes. So, if we want to achieve better compression then 2 conditions are there.

That we should be able or this coding should be able to you know create large blocks and the area which are being coded should be simpler. And this can only happen like I have given the example in case of say satellite images representing you know like this ice sheets like Antarctica or other places where a blanket snow cover is there. And almost all pixels are having same value.

And therefore, you can construct or this coding will construct large blocks having same value and high compression. Also, high compression can be achieved in desert areas or high densely forest covered areas where again high compression can be achieved because the variation in the neighboring pixels is going to be hardly anything or little. And that way we can do it. And also, the area would not be complicated which can be coded.

(Refer Slide Time: 27:33)

Example of Block Codes having Multiple Attributes



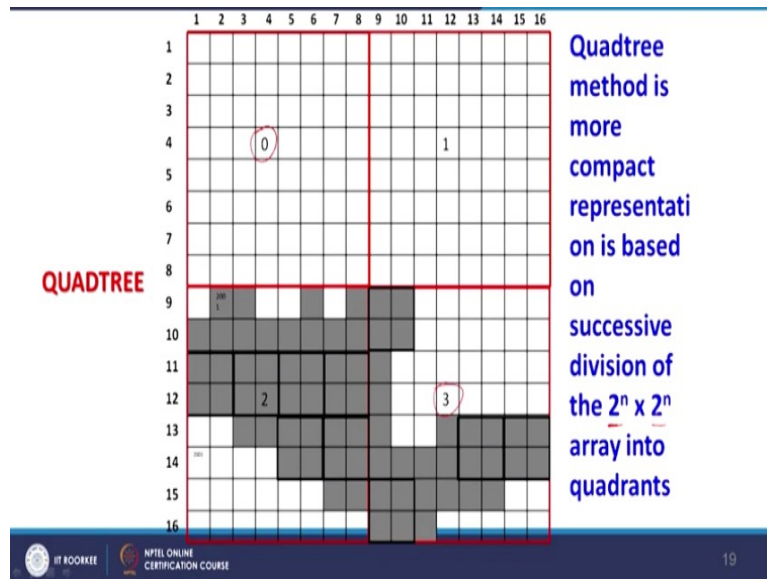
However, if we are having multiple attributes instead of single attribute in the previous example, then of course that can also be recorded but that kind of compression will not achieve as in case of binary or does not matter. So, here if I have to store then we have to store in a form of table as in previous examples also and then size and the values.

So, if I take that row 1 column 1, this is R which is given the green color. And what are the size, there are 4 by 4 size. So, the origin is given through this row and columns. The size is given and the value is given. That is attribute value. And this way, you can store the entire image though it may have multiple attributes. Multiple attributes mean individual cells or pixels are having different values.

But still, you may find redundancy in block coding by constructing larger squares. Now, this last one which we will discuss but there are many other. But based on this binary. the 4th technique which we are going to discuss and quite popular and the variants are also there. That is called quadtree. It is again name implies that quadtrees means there are always 4 blocks.

And then you are having search and next step is taken. So, let me explain the quadtree first here and then we will see other details. So, what is done?

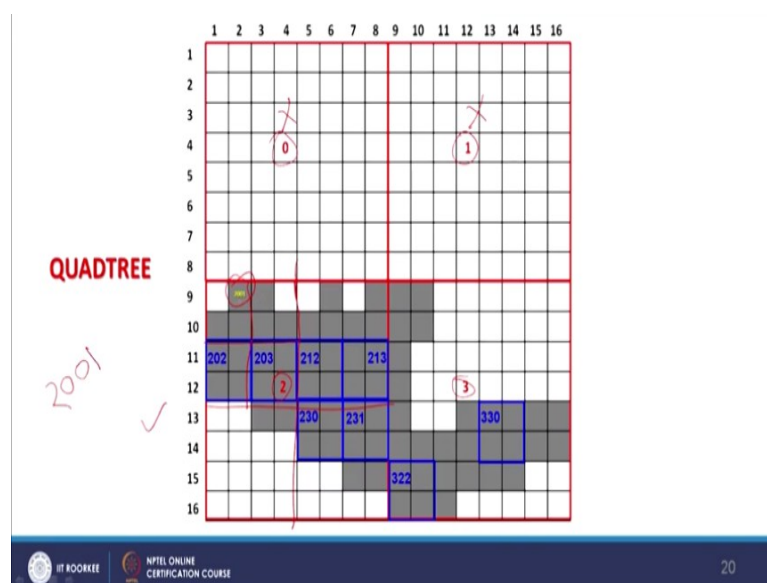
(Refer Slide Time: 29:25)



Once this image is given for this kind of compression using quadtree then you know, this is divided in 4 quadrants. And then each quadrant search is made whether there is heterogeneity present or not. If homogeneity is present like in case of quadrant 0 then no further divisions will take place. However, if heterogeneity present like in case of quadrant 3 then further divisions would be done till you reach to the unit level.

And unit level here is our cell. So, let us see that quadtree method is more compact representation which is based on successive division of 2^n by 2^n . And that is array into quadrants. So, this quadrant always it is divided into 4 parts till you reach to the unit level.

(Refer Slide Time: 30:25)



Like here, this has been done. And that entire input binary image is divided in 4 quadrants; 0, 1 and 2, 3. Again though these are numbers but these are codes. So, one can argue that why not 0, 1, 2, 3. Okay you can do it likewise, it does not matter. So, first divisions are made. Now, no further divisions are required for quadrant 0 and quadrant 1. However, the further divisions are required for quadrant 2 and quadrant 3.

So, we will concentrate for time being on quadrant 2. So then further this quadrant 2 is divided into 4 quadrants. So likewise, it is divided into 4 quadrants. Now, again search is made and still we have not reached to the unit level and still heterogeneity is present then further each this quadrant will be divided in 4 quadrants. So, if we concentrate on this northwest quadrant then again it is divided in 4 quadrants like this.

And the same way again search is made. So, quadrant this one, southwest quadrant and this southeast quadrant; both have achieved the homogeneity. And therefore, there will not be any further divisions. But this northwest quadrant and this northeast quadrant still are having heterogeneity. And therefore, further divisions would we made.

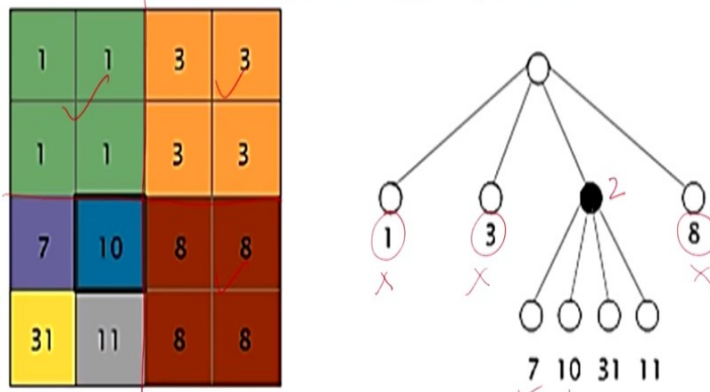
And if we made 4 more divisions then we are achieving the unit level. And then how coding is done. So, coding as you can realize that initially like here for this, we are having for this block that is block 2, we are having first that is written say 2 then further divisions are made. So, again the same numbering process will be followed that 0, 1, 2, 3. So, this becomes our block or quadrant 0.

And therefore, for this cell if I go then it is 0. Then within this once again divisions are being formed. So, I have achieved here for this 0, again divisions are made. So, another 0 because again I am going to handle northwest quadrant and within northwest quadrant, again 4 divisions are made and finally, I come to the unit level and the last one is my northeast quadrant which is 1. So, that is why it is called 2001.

And by this, we can code the entire image using quadtree concept. So quadtree method as we have discussed is more compact representation or data compression, more efficient by dividing this 2^n by 2^n into different quadrants. And this is tree; quadtree. So, basically say inverted tree.

(Refer Slide Time: 33:42)

Example-1 of Quatree having Multiple Attributes



Example: Position code of cell having value 10: 3,2

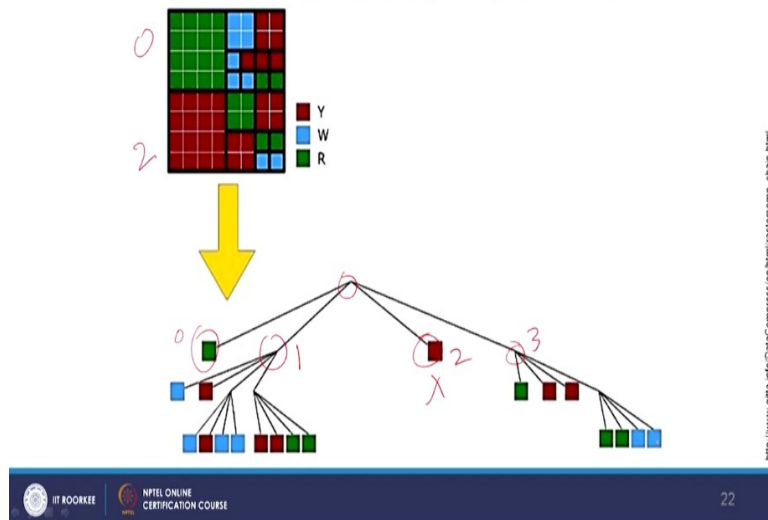
So, first we start for the entire image and then 4 divisions are made like this would be 2 and then there are further divisions till you reach to the unit level. So, example is given here. And each cell here is having some value which is 1, 3 and here lot of heterogeneity is there but this southeast quadrant, you do not have much heterogeneity. So, what is done here? 1, now homogeneity has been received; no further divisions.

In case of 3, homogeneity has been received; no further division. So initially, this image was divided into 4 quadrants, search is made, whether homogeneity has been achieved or not. If yes, like in case of 1; no further division. In case of 3, yes and in case of 8 also homogeneity has been achieved; no further division are required. Only further divisions are required in case of quadrant 2.

And quadrant 2 here is further divided into 4 quadrants. And each quadrant is having a cell value which is given here. So likewise, even instead of having just binary image also you can do on an image which are having various pixel values, not only with just 2 types; 0 or 1. So, this way we can compress very significantly.

(Refer Slide Time: 35:16)

Example-2 of Quatree having Multiple Attributes

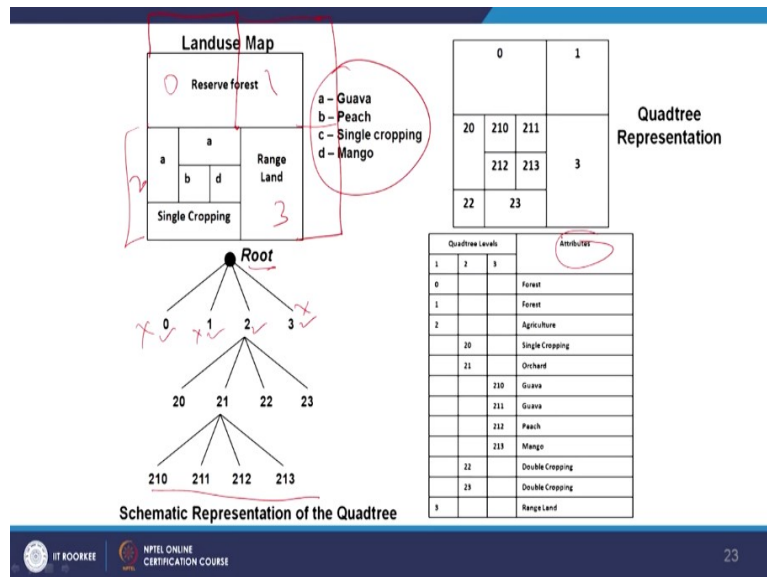


Now, this quadtree having multiple attributes. Example, I have already given. More complicated things can be done. Here also again, original image is divided in 4 quadrants 1, 2, 3, 4. And then here, no further divisions are required like in this is 2. So, this is 0, this is 1, this is 2 and this is 3. So, for 2; no further divisions are required. For 0; no further divisions are required. But for 1; divisions are required. And for 3; divisions are required till you reach to the unit level.

So, quad that means every time you keep dividing your image into 4 quadrants and you do it till you reach to the unit level. And since, in other way we can understand it's an inverted tree. So, these 2 terms; quad and tree are used to form. Now, there may be an argument. So, for the examples which we have taken are all image shape was square and this implying or you know doing coding using quadtree becomes much easier.

But in case, suppose the input image is not square in shape that means number of rows and columns are not same and it is having rectangular kind of thing then what would happen. So, no problem. First the system, what it will do? The algorithm, first it will go and make the image square by bringing some you know value here.

(Refer Slide Time: 36:58)

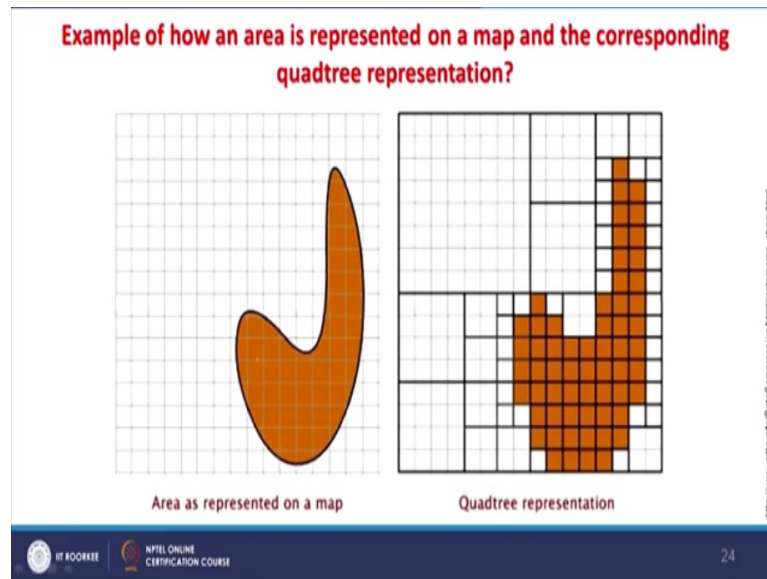


So, here this is the center of image in this particular example. So, in order to make like square this one, extension will be done on this side so that this also becomes 1 square here and then on this side also. So, then this block; all blocks become square in shape. So, 1 row and 1 column; top row and right column will be created with say no data values. But still these will be coded. And therefore, you can achieve still quadrant.

Example is given here that the root is divided. That means overall image is divided into 4 quadrants 0, 1, 2, 3. Then the quadrant 2 is further divided for 0 and 1 and 3. Their homogeneity has been achieved. So, 0, 1 and 3 homogeneity has been achieved. So, no further division for 0, 1 and 3 whereas for 2 further divisions are there. So, this is our quadrant 2. And then again here also keep doing till you reach to the unit level.

So, here these 3 layers representations; 3 layers divisions will allow us to achieve you know the coding. And your attribute table can store all related information especially about the attributes here and by which we can achieve. But these are very you know, schematic and simple examples. But as I have shown that multiple attributes can be handled quite easily by all these techniques also.

(Refer Slide Time: 38:47)



Now, how a quadtree has been represented. So, there will be some issues about the boundary. If you are having a scenario something like this. In left image, what you are seeing this you know orange area or brown area is having you know a very smooth boundary. But when we will code it, it will become a stair steps case; staggered kind of thing.

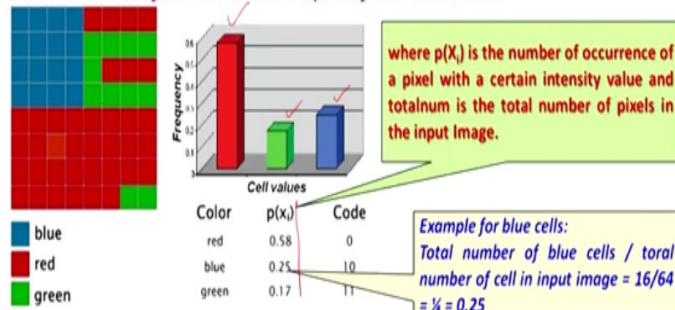
And that is the only problem which will arise because when we decompress it or uncompressed it, at that time we cannot achieve this smoothness. Because basically what we are doing, we are converting vector to raster. And as we have discussed earlier that whenever we do like this, there will be always error. Both transformations from vector to raster and raster to vector are not complete transparent.

That means there will be introduction of some errors from one conversion to another. This is what you are seeing. And this error basically is highlighted in the boundary part. This is what you are seeing in this example. Now, there is also a technique.

(Refer Slide Time: 40:12)

Huffman coding (lossless compression)

- Huffman coding compression technique involves preliminary analysis of the frequency of occurrence of cell values.
- The technique creates, for each symbol, a binary data code, the length of which is inversely related to the frequency of occurrence.



And most of the techniques which we are discussing are lossless, you know constructive techniques. That means that whenever I want, I can come back to my original image quality without losing anything. This is what means is lossless. And lossy techniques or destructive techniques like JPEG is one of the examples which will not allow you to reach to your original image. So, the quality of image is damaged.

And that is why they are kept in the category of lossy or destructive technique. So, in this Huffman coding compression technique involves preliminary analysis of frequency of occurrence of cell values. Every technique first will analyze the image as soon as it is given to the system. It will analyze. And accordingly, if whatever the option you choose, it will do the next step.

So here also, it is analyzing the frequency of occurrence of cell. How frequently the same value cell is appearing. So, you can say a frequency-based kind of technique. And this technique creates for each symbol or for example each category or each value of a pixel or cell a binary data code. And then length of which is in inversely related to the frequency of occurrence and why which you can achieve this compression.

So, here input image is given on the left side. There are 3 types of pixels are there. They have been given code blue, red and green. So, what we see the frequency of red pixels is the maximum compared to what you are having blue and green. So, this will further add the understanding about the technique. So, where this $p X i$ is the number of occurrence of certain intensity value and the total num is this.

Total num is the total number of pixels which are in the input. So, how many pixels are having same value that will be also given there. For example, for blue cells; the total number of blue cells in this input image are having 0.25 means say just a quarter frequency means say one fourth of the total and whole image is 1. So, that is why it is 0.25.

(Refer Slide Time: 43:12)

Run length coding: Exercise

Compress the following raster file with the compression techniques that you have learnt in this unit. How much space is needed for storage, if we assume that bytes (8 bits) can be used to store the data values?

$8 \times 8 = 64$ bytes

Cell value	Length	Row
1	4	1
2	4	1
1	4	2
2	4	2
1	4	3
3	4	3
1	4	4
3	4	4
4	2	5
5	2	5
6	4	5
4	2	6
5	2	6
6	4	6
7	2	7
8	2	7
6	4	7
7	2	8
4	1	8
5	1	8
6	3	8
4	1	8

Code-I:
 $22 \times 3 = 66$ bytes

IT ROORKEE NPTEL ONLINE CERTIFICATION COURSE 26

Now I am leaving these 2 images. Whatever we have discussed, these 2 exercises I am leaving for you to do it at your end. One is this one and another one is also for run length.

(Refer Slide Time: 43:32)

Run length coding: Exercise

Compress the following raster file with the compression techniques that you have learnt in this unit. How much space is needed for storage, if we assume that bytes (8 bits) can be used to store the data values?

$8 \times 8 = 64$ bytes

S4,T8
S4,T8
S4,U8
S4,U8
V2,W4,X8
V2,W4,X8
Y2,Z4,X8
Y2,V3,W4,X7,V8

Code-II:
 $22 \times 2 = 44$ bytes

IT ROORKEE NPTEL ONLINE CERTIFICATION COURSE 27

So, the first one where you know, you would do it that compress the following raster file with the compression techniques. And the compression techniques here is run length which you have learnt in this discussion and how much space is needed for storage. So, you have to

calculate. If we assume that the bytes that 8 bits; 1 byte equal to 8 bits can be used to store data values.

So, if this is the situation, how this will be stored that you can do it. This is the exercise, though the results are already given. But nonetheless, you use to try by yourself that how you would achieve. And then another way you know that how you would write this thing. So, please go through. I am not discussing in much detail about these 2 last slides. Thank you very much.