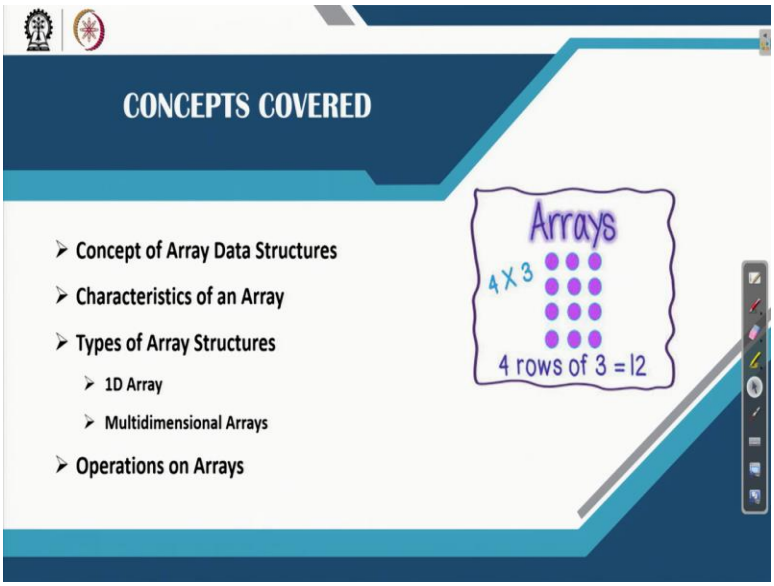
Data Structures and Algorithms Using Java
Professor. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture 11
Array Data Structures

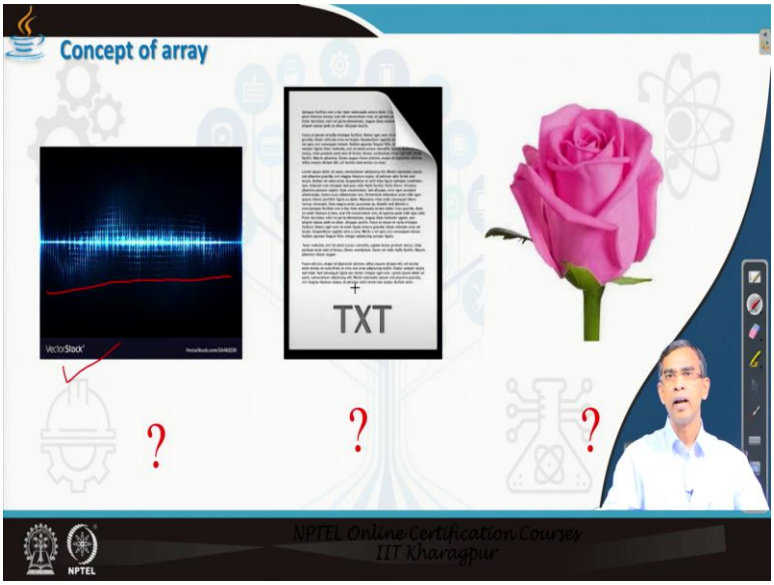
So, now it is the time to come into the core of this course. After having prerequisites about Genetic Programming and Java Collections Framework, that we have discussed in the last two modules. So, this is the third module in our course and this module totally discussed about the array structures. Array is a very important data structures known so far, it has many use.

(Refer Slide Time: 00:57)



So, in today's lecture, we will try to cover the basic concept of arrays and then arrays always should follow or obey certain characteristics, so we will see exactly what is the main characteristics that an array must satisfy. And then I will discuss about the different types of array known so far, mainly 1D array and 2D arrays, and then meaningful operations, useful operations on arrays. So, this is basically the agenda of this lectures. And now let us first start with the concept of Array structure.

(Refer Slide Time: 01:34)

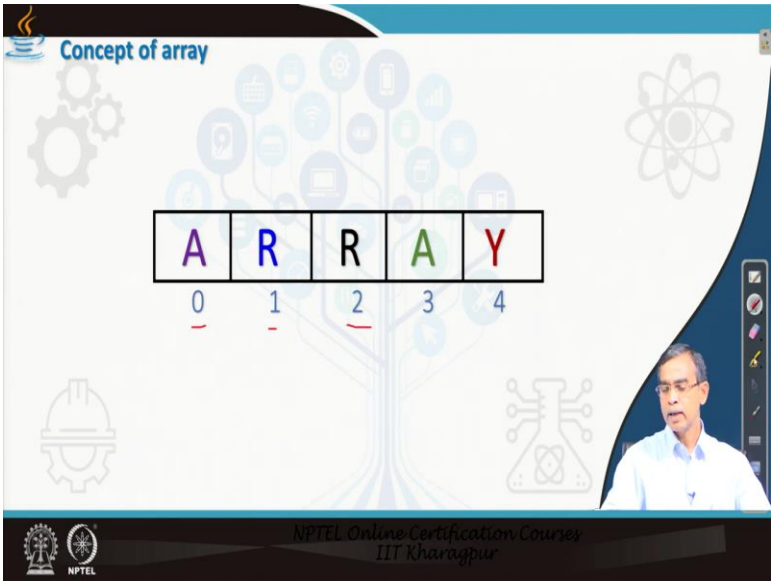


Now we know, so far, our current data manipulation is concerned, data usually not of that innocent type like a set of numbers or set of names, but data may be as complex as it is. Here I have mentioned few example of datas. The first example as we see here, this is the one data that data you can think about is the data that is there in a from the speech signal or the data that is there in the form of a message that is transmitted through network.

So it is basically the signal, that is the network is through or is a speech that we have. Now, this is another example of data it is a text data, it is basically conventional data that can word editor or word processor document produce. And this is an another example, is a, is an image.

Now my question here is that, how all these data stack that data can be stored? What form of structure that can be used? We will see exactly after learning the array structure, we will learn about whether all those data can be stored in the structure like array and if it is there, then how we can store it.

(Refer Slide Time: 03:06)



Now let us proceed, for the concept of array. What exactly an array is, actually array is called a linear data structure, more specifically it is called the indexed data structure. What is the meaning of indexed? The idea is that, in an array we can store any element, for example, in this array we store 5 different elements, of different colours we can say or different font or different style whatever it is there.

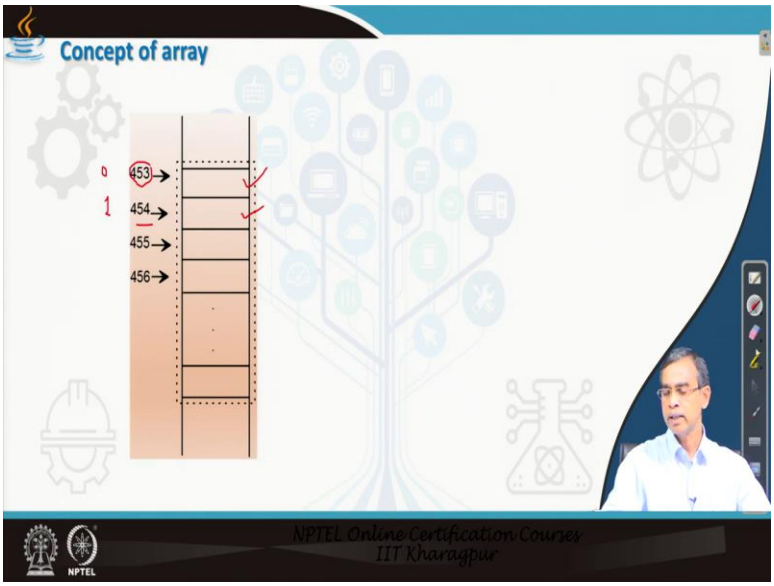
And the indexed because here the indexed is, this is called the zeroth index, this is the first index, this is the oneth index, second index, and so on, so on. Now, index starts usually in C, C++, and Java programming language, index starts from zeroth level, but in older programming language like, Fortran, Cobol, Pascal or other programming language the index starts from level 1. So, starting with 1, the first element is stored, second element is stored, and so on, so on. And, so this is the concept it is there indexed.

Now, exactly what is a index basically, an index is, an index is basically to axis in the memory where the elements should be placed or from where the element needs to be retrieved, that is your index concept. But this indexed is not that physical location of the memory, you should know that it is a logical, from the programmer's address it is. So, programmers give the name that zeroth or this one, or there are index concept we will discuss in details about it.

So, an array is basically is a collection where the elements, elements can be number, elements can be a character, elements can be a floating point values, element can be any object or defined

by user can be stored one by one. And starting location of the array is 0, so for C, C++, Java is concern. In Java, we will always consider the starting location is 0, otherwise you can specify any other starting location there.

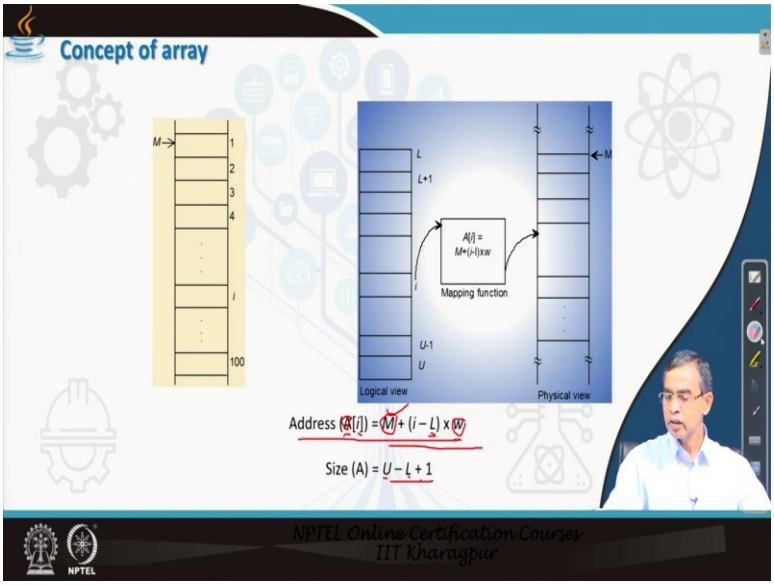
(Refer Slide Time: 05:24)

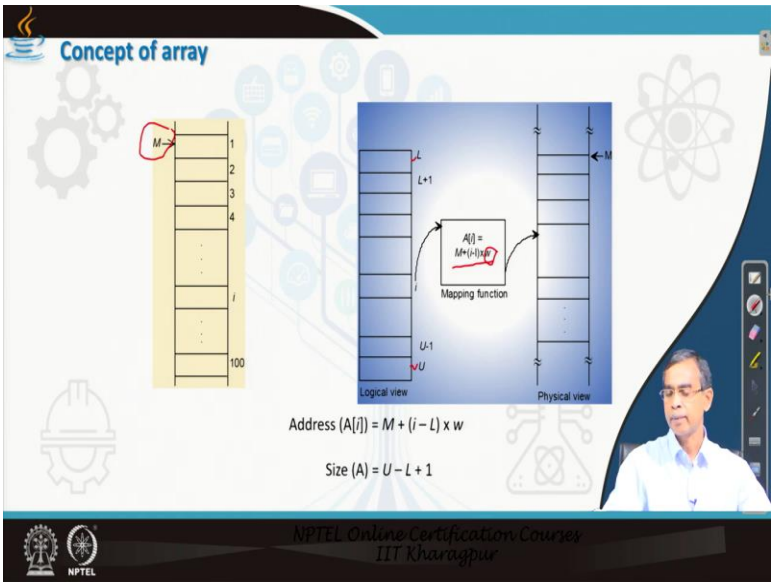


Now, so this is the concept of array and the array index is basically as I told you, is a programmer's location or programmer's address. But actually, whenever the elements are stored it needs to be stored into the memory, computer memory, primary memory. So, memory address is different. For example here, as we see the array, the memory location is 453 start onwards and each store 1 word in the memory, 1 word maybe 1 byte or 2 bytes or 4 bytes depending on the architecture of the computer, how the memory they can manage, let it be 1 byte.

Then the first byte is stored in this location that means the first character say, A is stored here. Then the second character which needs again another 1 byte will be stored in the next location stored here. So, actually the 0 which is basically the programmers address, 1 which is the programmer address but the physically they are stored in a 453, 455, 454, and so on, this is basically the actual or physical location of the elements in the memory. Anyway, so these are the concept there is for so array is concerned.

(Refer Slide Time: 06:51)





Now, now array is basically is a, is a index location again it is also, one by one conjugative location the in the memory is concerned. And there is a mapping actually, mapping in the sense that the location defined by a user or used by a programmer and the actual location that is used by the system in memory are to be mapped somehow. So for this mapping, so this formula you can use for that, the address of any ith element in the memory, so Ai, here A is basically is the name of the array. Every array should be uniquely given a name by the programmer.

So, usually capital letter is used or small later is used whatever it is there, so for example here A is the name of the array and within the square bracket i, indicates that it is according to the

programmer's notation, intuition is basically ith element. Now, in Java it is i is basically 0 starting point and then anyone. So, Ath location, if i is Ath, actually it is the nineth element.

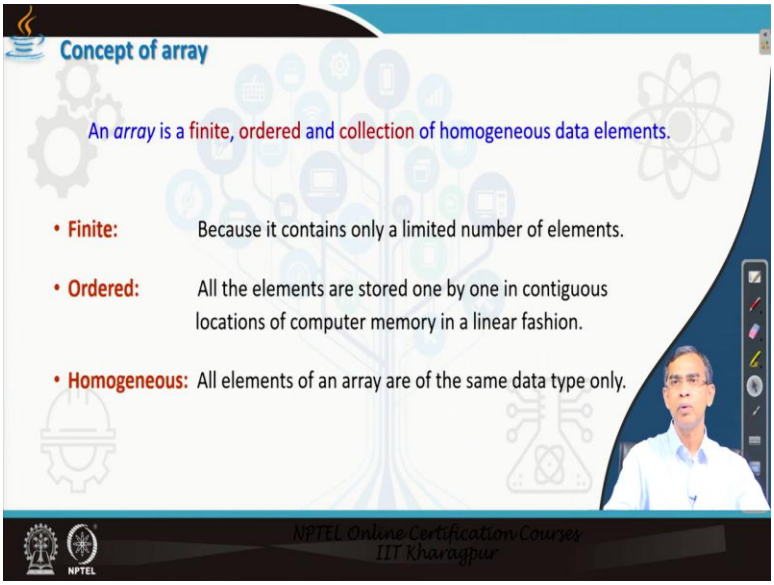
Anyway, so if M is your starting location, that means whether 0 in Java or 1 is a Pascal, whatever it is there and L is the lower bound, and w is a word size, here word size is important, because if you want to store character, it require only 1 byte, 1 word. But if you want to store some user defined type who is equal say 4 words, then it multiplied by 4.

So, M is basically the starting location of the memory that is the memory address, for example in the last example this 453 and L is a lower bound of the address or index, so for example, 0 and then we will be able to find the address of the ith element using this formula. So, if is a 453 plus i minus L, L can be 0, if it is Java L can be 1 if it is Pascal, multiplied therefore, so this one is there the address. And size of an array element can be obtained by this formula, where U is the upper bound of your array index so 0 to 9, so it is basically it is 9 minus 0 plus 1 so size of the array if the index starts from 0 to 9, it will basically 10.

So, this is the concept logically it is followed by all programmers and for the better manipulation of the array elements this concept needs to be little bit learned about it carefully. Again, I repeat, M is the memory location where the array will start and L and U are the lower and upper bound of the array bound, that means say 0 to 9, if we want to store 10 element.

And Ai indicates the location of the ith element in the memory that can be obtained by this formula, where w is the amount of bytes or words rather that is required to store 1 element in the array. So, that concept needs to be very clear and this way we can store an array in a memory or actually from the user view to the actual view of the array actual.

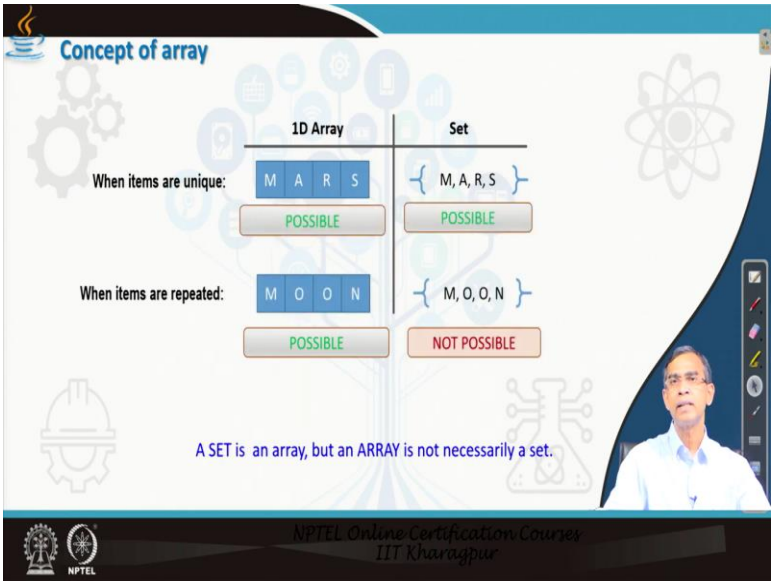
(Refer Slide Time: 10:36)



Now, there are certain characteristics or criteria that an array always satisfy. So, this criteria is basically from the definition point of view, the computer scientist gives a very formal and clear a precise definition what exactly an array is. So, according to the computer scientist and array is called a finite, ordered and collection of homogeneous data element. Now, here you note few important points, it is a finite that mean array should include not that infinite element, it will always consider finite element, it may be 100, it may be 10, it may be 1000, it may be 100000, but it is always finite size should be known.

Unlike other linked list and others, you can store theoretically infinite elements. Now, ordered is the concept is that all the elements should be stored one by one in the contiguous memory location. And homogeneous means that, in an array it always store only a particular type of data, if you store an integer value, then all integer values should be there in an array. So, if an array of integers means all values are of type integer, but it is not true that an array can store at the same time together an integer value, a floating point value or string whatever, it is not possible. So in that sense, array is homogeneous.

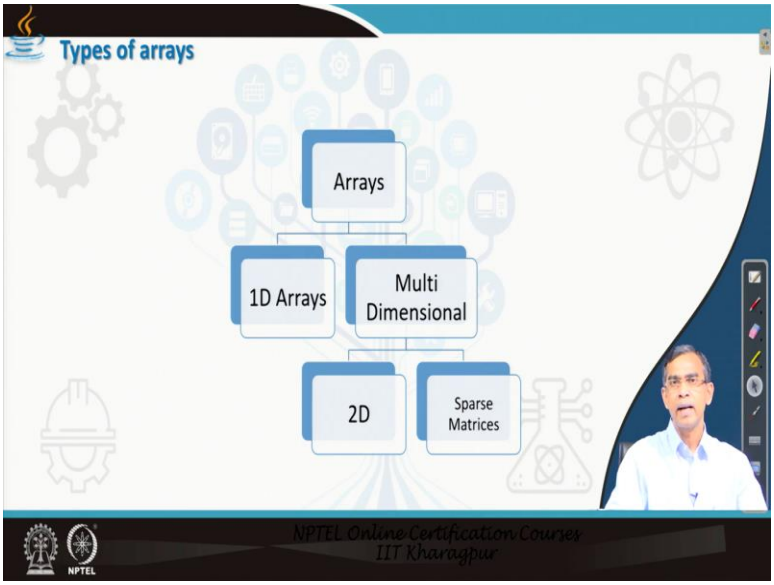
(Refer Slide Time: 12:07)



And another one difference, actually array is a group of elements we can say, more precise we can say set of elements. But the difference is that an array can contains any number of elements whatever it is there and any number of times a particular elements may occur, this means that array allows delicacies. For example, here, an array can contains M A R S all are distinct, no problem.

Now it is same as set, but here the difference between the array and set is that where repetition is allowed in array, but repetition is not allowed in set. For example, if we want to store MOON, we can store it in an array, but the MOON cannot be store in a set, where O occur twice. So, set does not allow duplicate element to be stored.

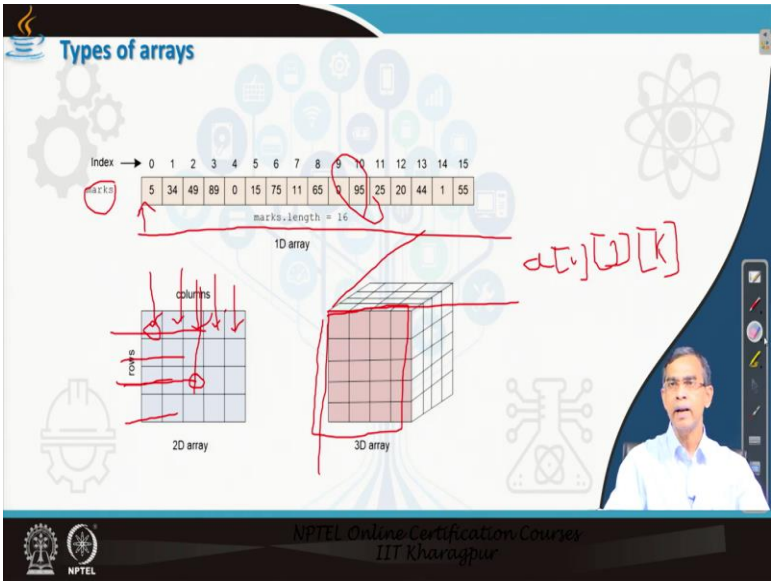
(Refer Slide Time: 13:06)



Now, let us see what are the different type of data structures, those are known. All the array structures can be broadly classified into two categories; 1D array this is the simple most array structure, and little bit complex array structure is called the multi dimensional. As the name 1D means one dimension, that means only rules are there, but whereas multi dimension it can be either 2D or more than 2D's also.

If it is there, whether 2D or 3D the array can be better classified as a sparse matrices. Now, we will discuss about when an array or multi dimensional array is called the sparse matrix and others, usually 2D array is also alternatively termed as matrix.

(Refer Slide Time: 13:48)



Now here, is an example as you can say, the array as a 1D array the first example that we can see here is basically is a 1D array. Now, here as we see the index starts from 0 to 15, so lower bound is L0, upper bound is 15 here. And the elements as we see the homogeneous elements all are of integer type stored there, 0 may occurs many time for example and total size of the array as we see is 16. And this is basically is the one location in the memory which is basically starting point the array store and this array is named as marks.

So, every array should be uniquely by name. So, if I say marks then within square bracket 10, so basically it hits the 95 element, that means the tenth element in the array this one. Now, so this is about 1D array, 1D array in order to access only one index is required, that means here i this index is there. Now, 2D array, here this is an example of 2D array, here actually it see these are number of rows are there, this is the first row, this is the second row, and third row, and fourth row, it includes four rows are there.

In addition to four rows, there is the columns also there, one column the first column, second column, third column, then fourth column, and the fifth column. Now, so this is one array we can say with 4 rows and 5 columns all together we can say it store 20 elements, 4 cross 5, 20 elements are there. Now if you want to access a particular element, then unlike 1D array we have to mention two index, indices; one is that, what is the row number, so it is basically row 3, so row, and this is the what is a column number, it is basically 3.

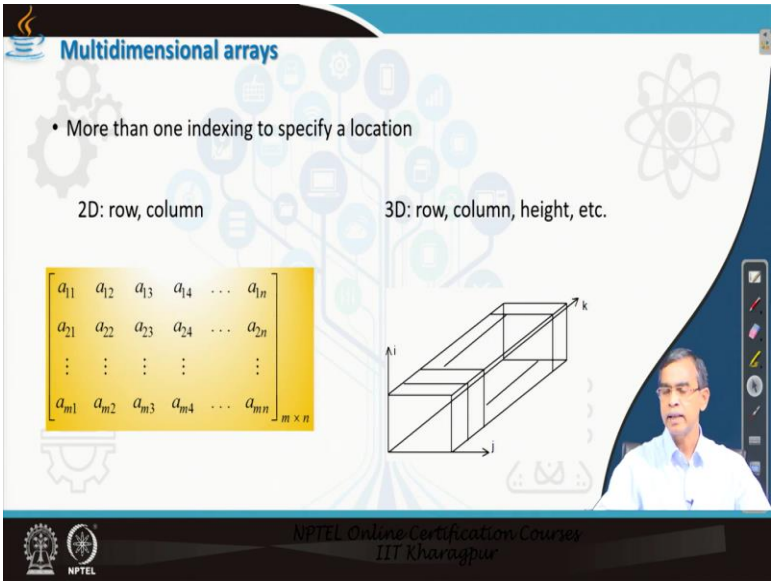
So, this is a 3 3, this is basically 3 4 and this is also 4 5, whatever it is there. But again, index like in Java start from 0 to the upper values. So the row, this is the zeroth row actually and this is, this is basically row 3, this means that fourth row actually. So, 0 because index starts from 0, similarly, column index also start from 0 U. This means that this is the element which is according the users notion it is a 0 0 element, and this is basically 3 4 element, because the row index is 3 and column index is 4.

This is the 2D array. Now here, the same concept can be explained in terms of other multi dimensional array. So there in case of 2D array only 2 index is required, here in case of 3D array like the 2D array this is say, I, and this is say j, and this is say k, so it is called the 3 dimension. So, i, j, k and any element inside this can be index by Ai, Aj, k. So, it is basically if A is the array then its index is basically i and then the first is basically row and then j, and then k.

So, there are 3 indices are required in order to access the element. So it basically says that, this basically also alternatively call in 3D it is one call page, so i, j, the k indicates that which is the page, so zeroth page or tenth page depending on, so this is a k, this is a page.

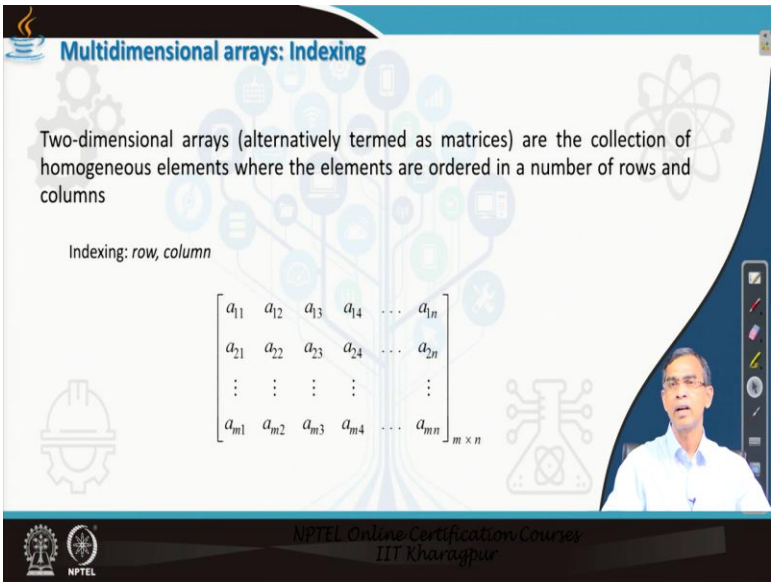
And then in each page row and column needs to be indicates, so that it basically pinpoint the actual location of the array elements there. So, this way the array of different dimensions and the concept can be, concept can be extended likewise to the higher dimension, nth dimension, then A i, j, k, l, and so on. So, this is a concept of the different arrays. Now, there are few more type of arrays, they are called the sparse, sparse matrix like.

(Refer Slide Time: 19:02)



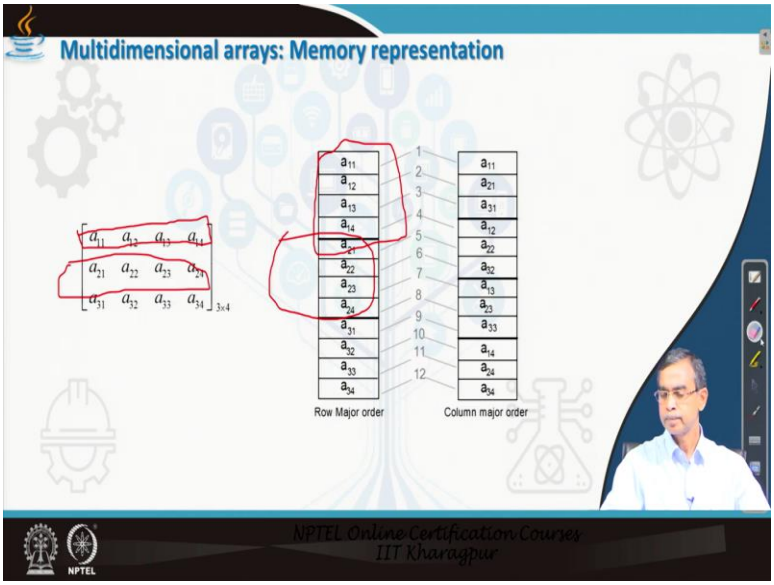
Now, multidimensional array we are discussing about 2D's array and 3D's array, we have discussed about row and column are two index and then here row, column, page or height are the three indices for the 3 dimensionality array. And usually they are called matrixes.

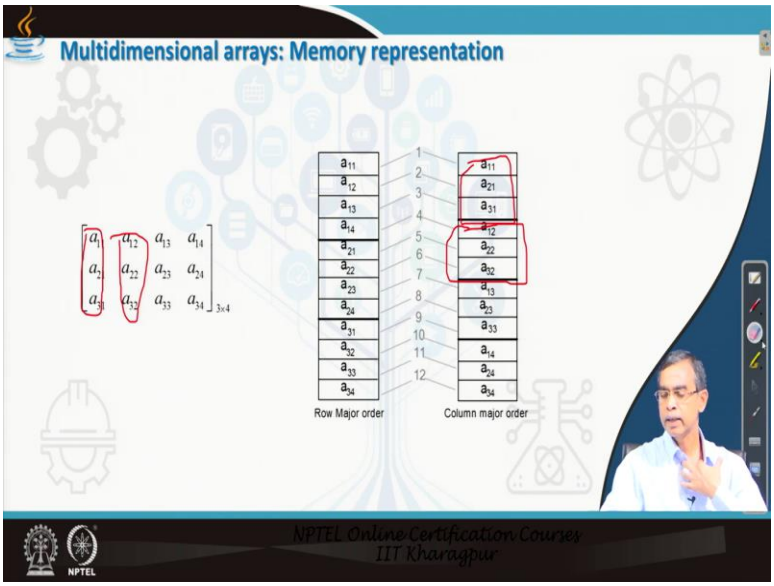
(Refer Slide Time: 19:10)

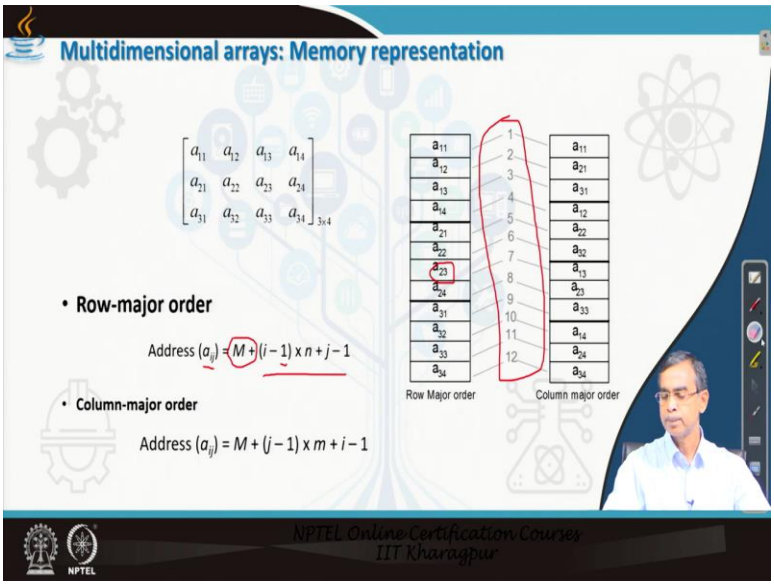


And in the regular concept of matrix a m cross n, m indicates the number of rows, n indicate the number of columns.

(Refer Slide Time: 19:20)







And here, the array whether it is a 2D or nth dimensional array, absolute no problem how they can be stored, because our memory is always is a 1 dimensional in nature. So, here the array can be stored using two what is called the format, one is called the row major order, another is the column major order. So, basically 2D array can be stored in a 1D memory actually, so here basically, this is the first row. So, row major order means, here we see first rows of all the elements should be stored first, then the second row of the element will be stored in the second row and here is a second row and so on.

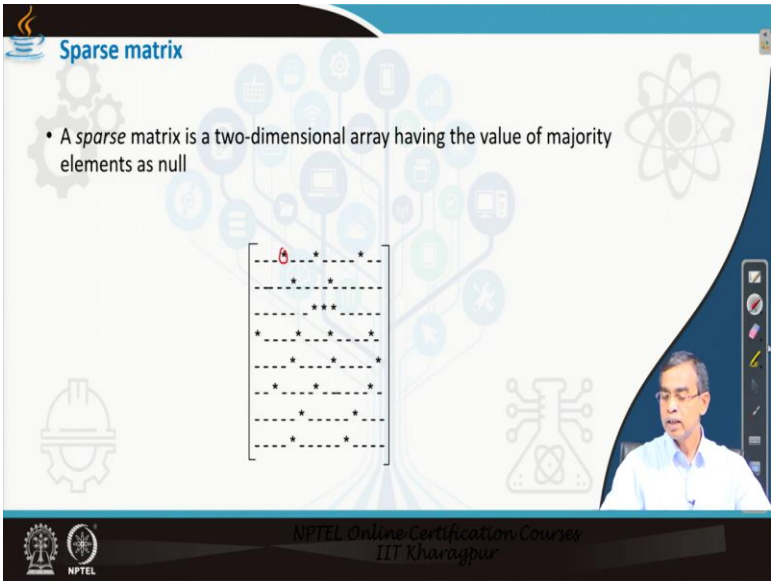
Now this is the row major order, on the other hand if it is a column major order, then it is in the column major order, then here the first column will be stored here in the first column, as you see here and the second column will be stored here in the second that position like this one. So, it is basically column by column. So, these are the two ways that the element can be stored either row major or column major order. Again, there is a again problem arises that if is a row major or column major how a particular element can be map from user point of views to the memory point of view.

So they, likewise the 1 dimensional array that we have already discussed here also the address can be computed. So, here if M is the location of the, starting location of the memory then any ijth element its address in the memory can be calculated using this formula, you can check it and if it is basically okay, so here we assume that the starting location is start from what is call the 1, but if it is 0 then this 1 can be, so i minus l n minus j minus l also can be, so starting location is there.

So, here in this index formula we assume that starting location is 1 here for example, the starting index is basically 1 to 12, if it is 0 to 12 then it can be just instead of 1, we can replace by 0, that is all, that is the only thing that you can do otherwise everything remains same. And here for example, so this is a location say 2, 23, so here for example, i is 2 and 3 if we put the formula for i equals to 2 and j equals to 3 and n in this case the number of columns so 4 and you can find this is a formula, 7 can be obtained.

So, this is a formula that we can be used to directly convert from the ijth location to the memory location, if you know the memory location say is a 411 then adding these values it basically tell in which memory the element is stored. So, this is the concept that the array can be stored in memory.

(Refer Slide Time: 22:50)

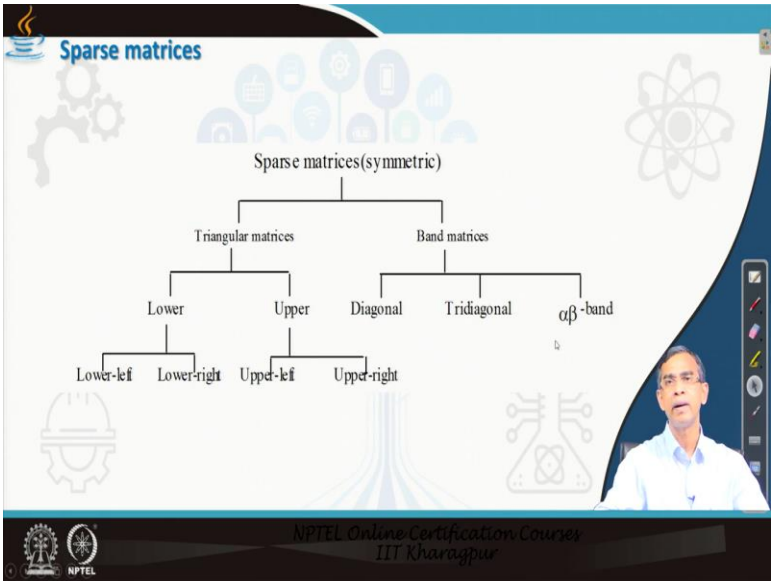


And sparse matrix again, I just wanted to mention what exact the sparse matrix is. Sparse matrix is a two or higher dimensional matrix actually, where that all elements are not present in all location only few elements are there, so here in this example those are the star is basically the actual element present but majority of the elements are not present there.

Now, sparse matrix is many, in the many situation we use to follow it where we can have a little bit compact way to access it and store it because majority of the elements are not present unnecessary storing all the null values or no values is basically useless of the memory requirement. So, there is point that scientist has decided how a sparse matrix can be represented

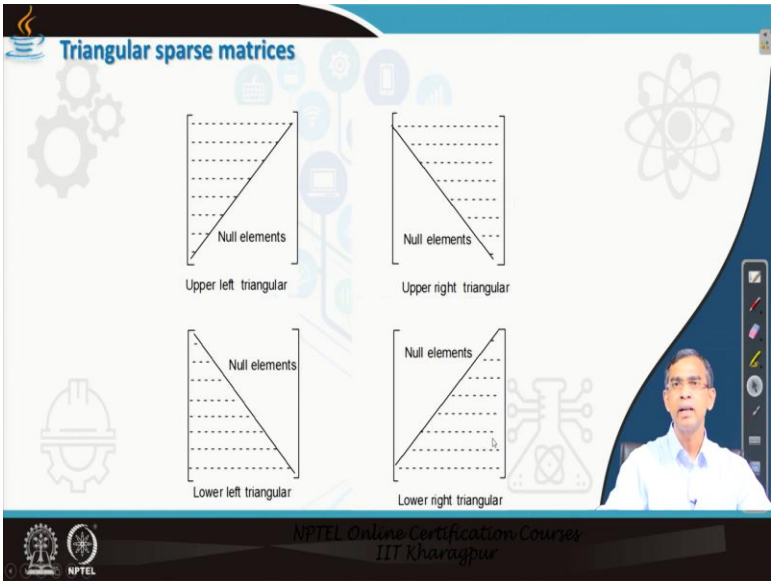
in an efficient so far the time of (())(23:41) is concern, storage is concern, and memory compaction or memory utilization is concern.

(Refer Slide Time: 23:46)



Now, there are few more different type of sparse matrix are there, I have given what is the different sparse matrix is possible, one by one I can discuss about.

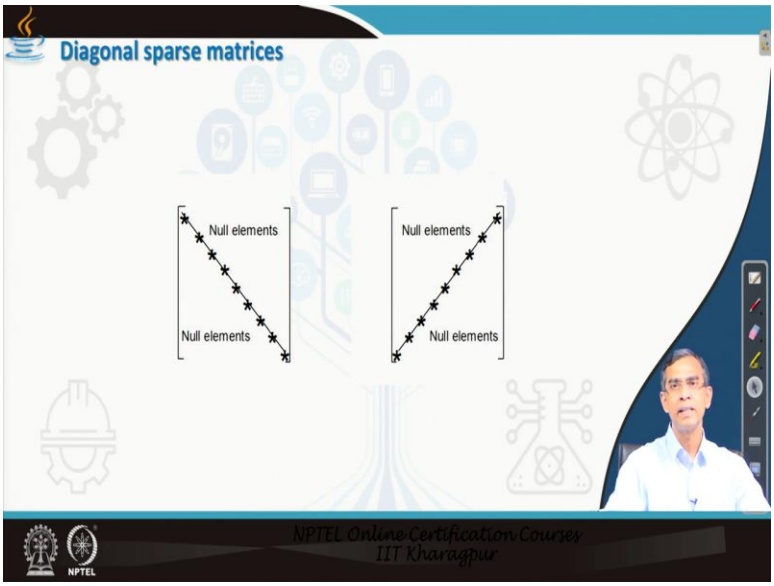
(Refer Slide Time: 23:53)



These are the some type of sparse matrix, where the elements occurs in a particular one half of the arrays, they are call the diagonal half. So, it is basically upper left diagonal, it is upper right diagonal, lower left triangle or lower right triangle and everything. This basically the elements

stored in a triangular form. Now, here again, if we want to store such a sparse matrix in memory, so better not to store all the elements which are basically null elements or no elements. So, there is again the clever way that the element can be stored and then accordingly index can be formulated. So, this idea is basically

(Refer Slide Time: 24:38)

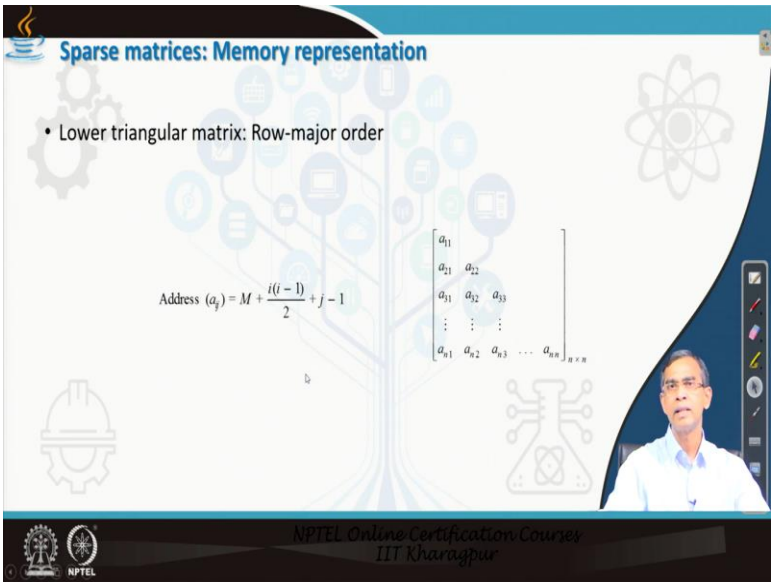


This is a sparse matrix, there is another type of sparse matrix it is called the diagonal sparse matrix, where the elements, all elements which are other than the diagonal are null. Now diagonal can be like here, top to bottom or bottom to top these kind of things are there, these are the two different diagonals is possible.

(Refer Slide Time: 24:57)

And there is again tri diagonal, the elements are stored just one above the diagonal or just below, but not all. So, this is also one sort of band matrix or band sparse matrix it is just like a band. Now two type of bands that we have discussed here.

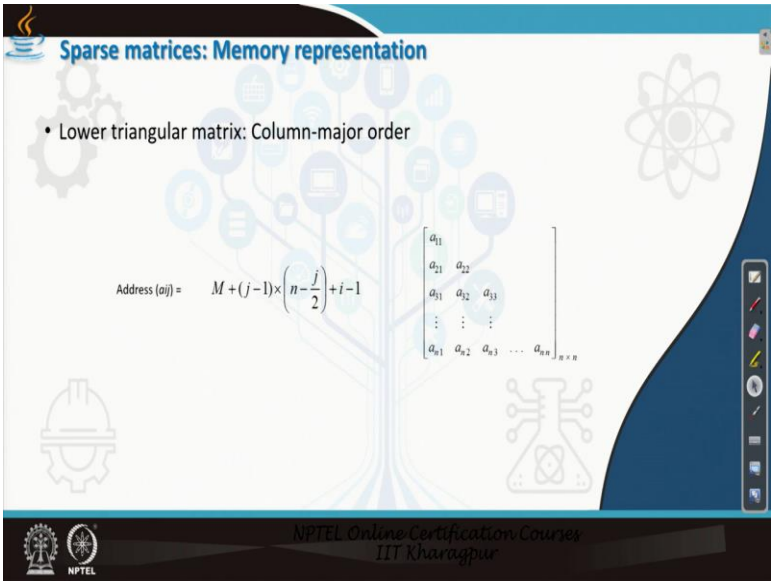
(Refer Slide Time: 25:14)



Now, here is the formula by which you can efficiently store the array elements starting from the Mth memory location the sparse elements to be there. So, if we follow row major order then this is a formula that you can include, but here we assume that the index start from, index start from

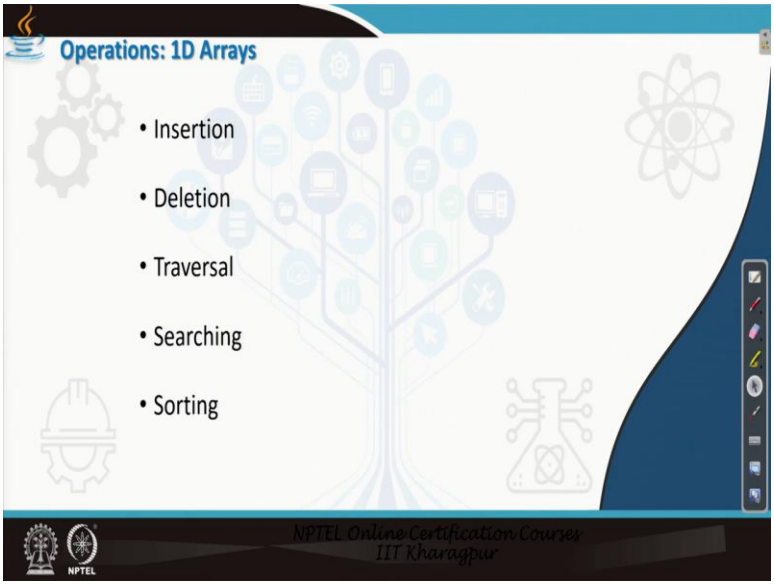
1, if index start from 0, then we just simply this will be 0 or 1 in place of 1 it will be 0 like. So, this is the row major ordering.

(Refer Slide Time: 25:46)



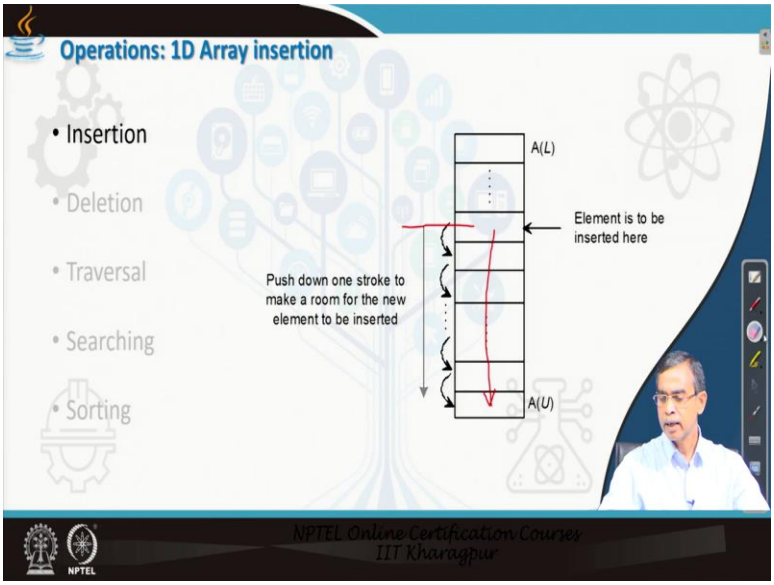
Similarly, the column major ordering which will be, which will take this form, you can verify the formula of your own.

(Refer Slide Time: 25:56)



Now, let us come to the discussion about operation with arrays. The several operations like Insertion, Deletion, Traversal, Searching, and Sorting that can be done here.

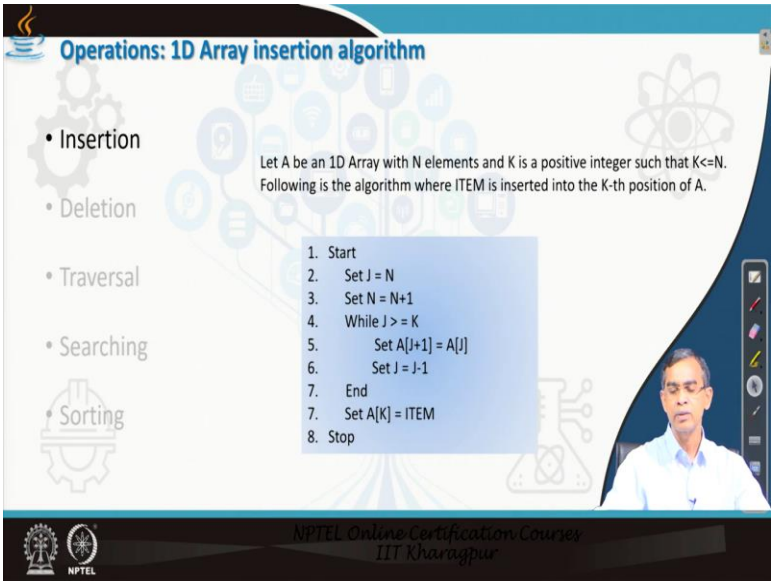
(Refer Slide Time: 26:04)



Now in insertion operation if you see, the first of all if you have that okay array can accommodate, because it is a fix size and then if we say that total size of the array that means U to L is not filled but the later, the bottom part of the some location where is free, then in order to insert a particular element at in between some elements then what we have to do is that we have to make a room for the next.

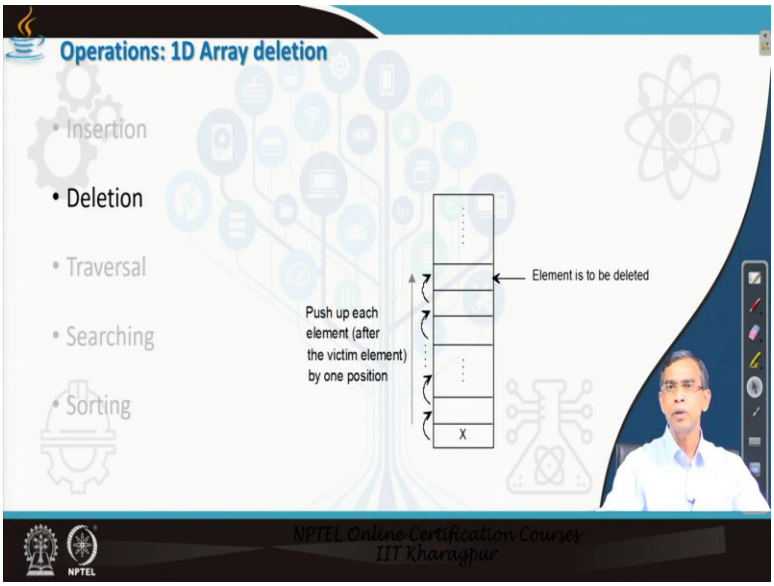
For example, if you want to store an element here, then from this element onwards we have to push little bit one here, so that we can make a room for the new entrance and at the element there. Now, here that pushing an element downwards by one location is a task to be carried out prior to insertion. So, this is basically the idea about the insertion operation.

(Refer Slide Time: 27:04)



And then there is an algorithm that you can follow also, we have written an algorithm for your understanding, you can check it how it is basically pushed one element towards down and then make the room ready for the new entrant and this one. So, this is algorithm you can go of your own and you can understand how it works.

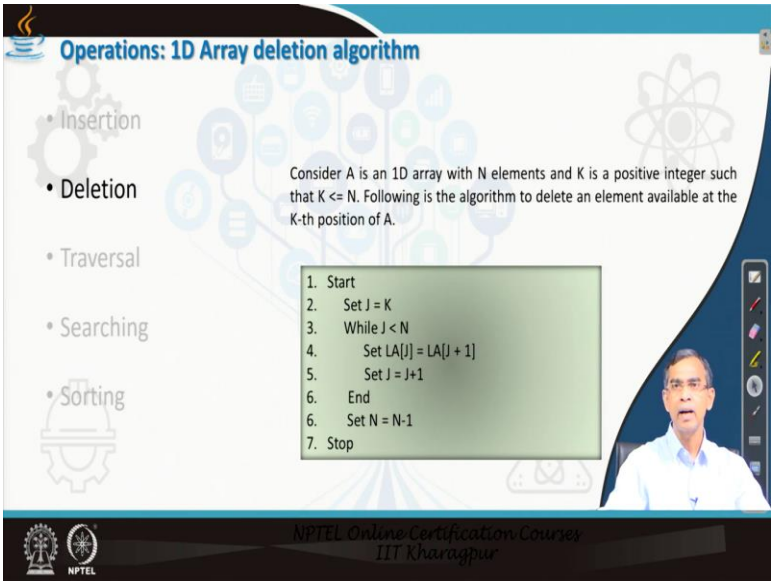
(Refer Slide Time: 27:16)



Now, deletion just opposite to insertion. If you just delete one element that position should be make void like, we have to fill that position by the next element onwards, so that that elements

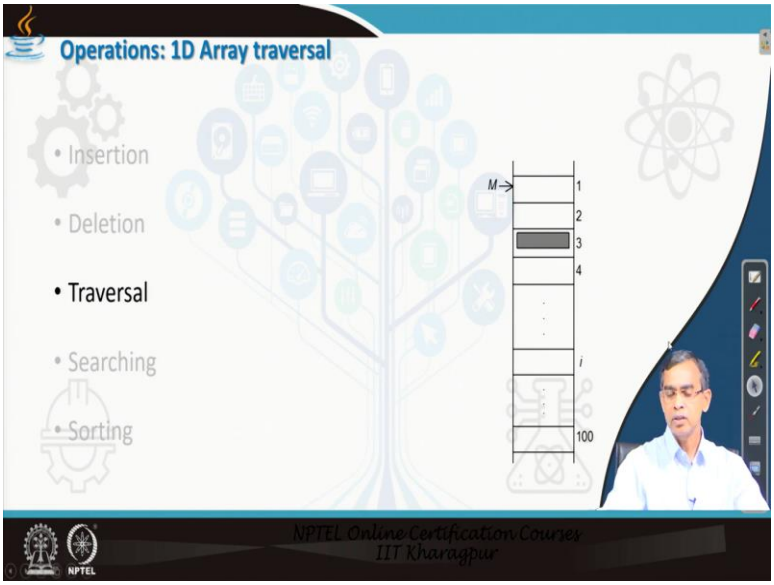
after deletion occupied by the valid element actually. So, in case of insertion push down, but here in case of deletion you have to push up one location as there.

(Refer Slide Time: 27:42)



And there is a, I mean program, the algorithm that I have mentioned you can go through the algorithm how this operation it is there. But we will write program easily in Java concept also very simple programming that it is required.

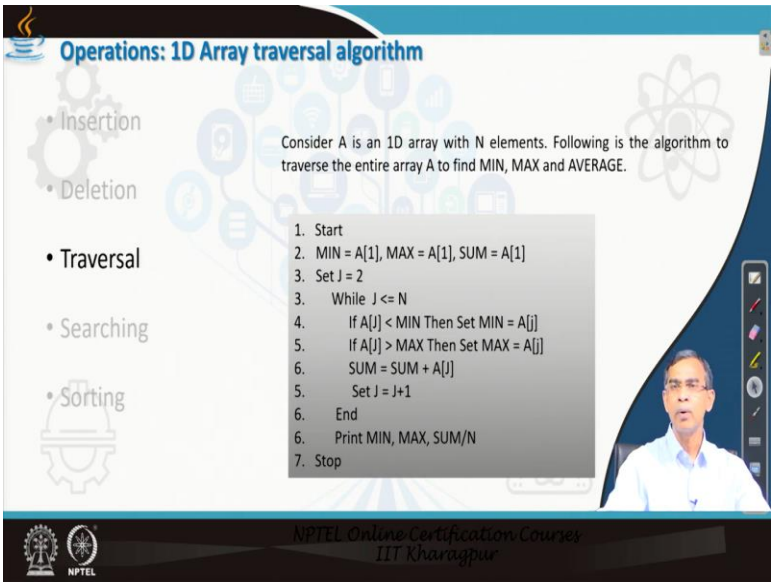
(Refer Slide Time: 27:53)



Now, the traversal, traversal is basically visiting all the elements to perform certain operation. For example, you want to search for the largest number, so what exactly the thing is that we have

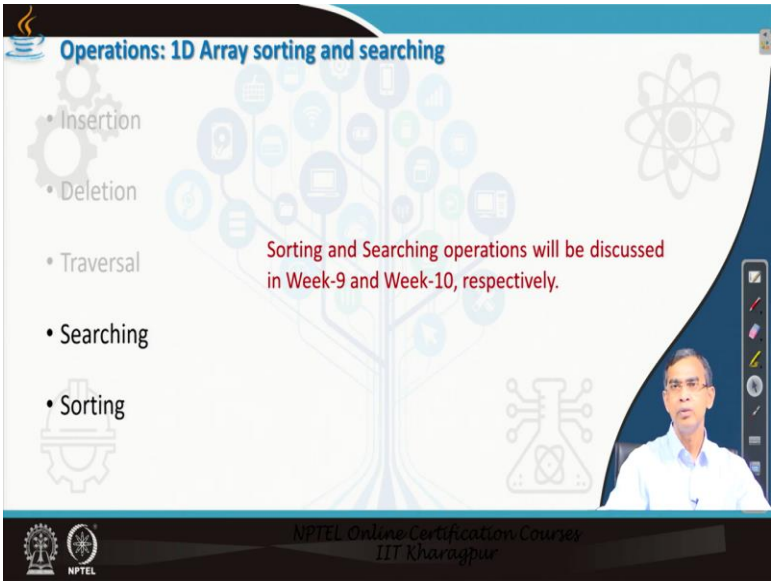
to start from the starting location of the array elements and then go one by one and check that whether elements are available or not, once we find it then you can stop it. So, searching is the one simple way the traversal is there.

(Refer Slide Time: 28:17)



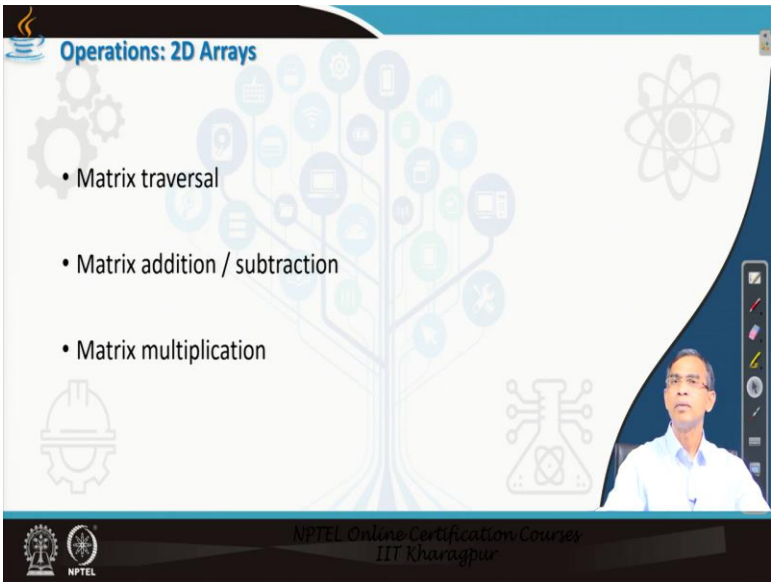
And also sometimes we have to traverse all the elements to find either average or max or minimum, so this is an algorithm Basically tells you how you can do all the things together finding a lowest number, smallest number, the largest number, and average of all the numbers assuming that array store integer values or floating point values.

(Refer Slide Time: 28:38)



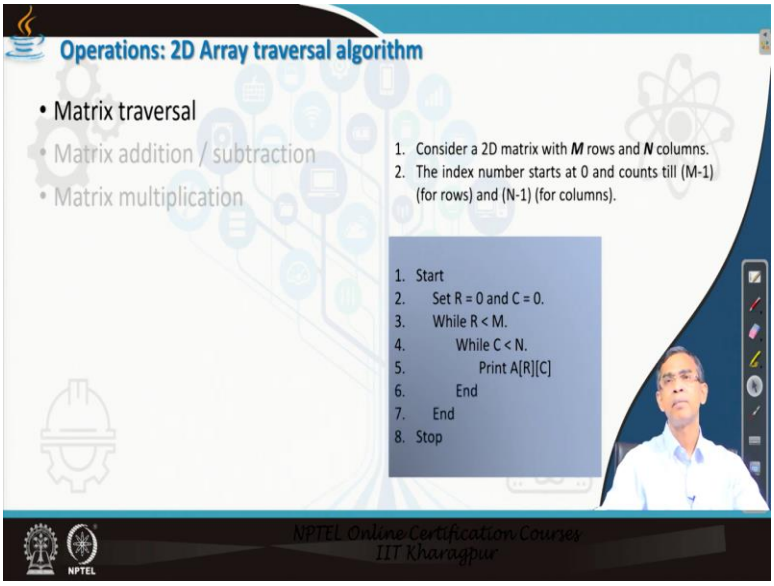
Now, searching and sorting are the two important operations that can be accomplished on the collection which is stored in an array but sorting searching will be discussing details when we will discuss algorithms and their implementation. This is a towards the end of the course actually.

(Refer Slide Time: 28:58)



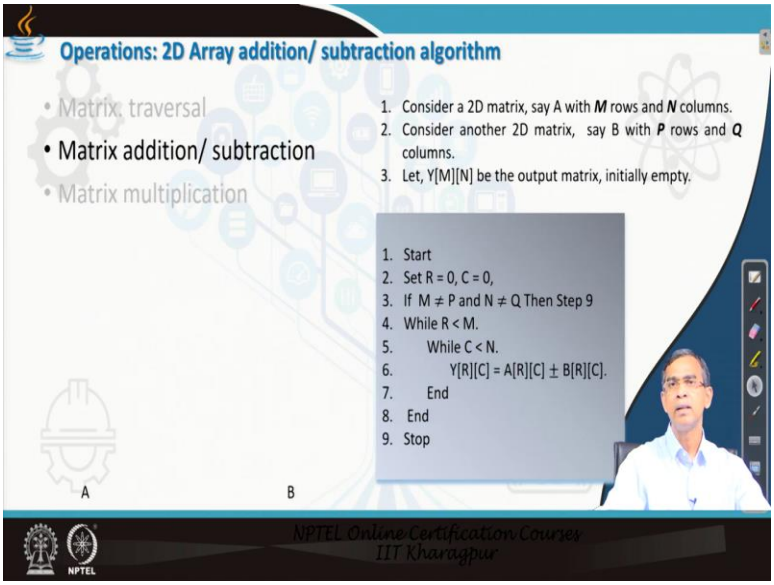
Now, operation with 2D arrays. Now, 2D arrays is extensively used, seeing this is a matrix in mathematical manipulation born in mathematics actually, for the matrix algebra and others. So, mainly if you want to create a matrix or if you want to print the matrix or if you want to add or subtract two matrixes or multiplying two matrixes to obtain this matrixes, the code is there.

(Refer Slide Time: 29:23)



Now this is a simple program, by which we can see how all the elements those are there in the matrix. So, this algorithm tell that if a M is a number of rows and N is the number of columns are there then, how a matrix can be visited. So, starting from R equals to 0 and C equals to 0 and then we can repeat the same procedure in a two loops and we can print all the elements there, and then the matrix can be traversed, is printed.

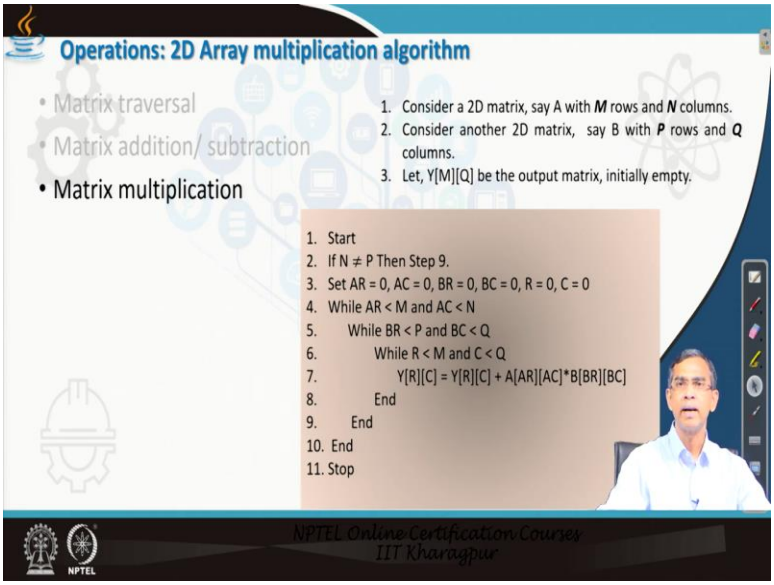
(Refer Slide Time: 29:51)



Similarly, addition and subtraction, it is also a two loops is required; one is a internally loop, and external, the outer loop. So, for the internal loop, it will visit all columns and for internal loop, it

will visit all the rows. And then this way is basically continue the step of adding, subtraction until the entire arrays is there and this program you can follow how it can work for you.

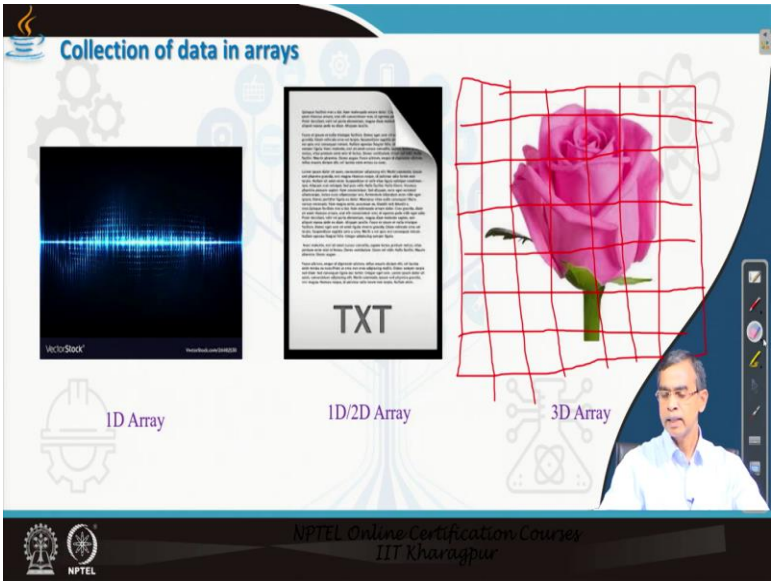
(Refer Slide Time: 30:14)



Now, multiplication is basically unlike the addition and subtraction three loops are there. I have given a logic for you, so it is basically two matrixes those are conformable for their matrix operation because all matrixes cannot be multiplicable, because it should be the number of columns in one, the one matrix should be same as the number of rows in another matrix then only we can do it.

So, you have to first check that if it is satisfying or not, if it is satisfy, then perform this loop and once the loop is completed it will result in resultant matrix is a different matrix Y. So, there are basically two matrix A and B and the resultant matrix is Y, where the number of rows of this Y matrix is same as the number of rows in the A matrix and then number of columns in the Y matrix is same as the number of column in the B matrix and then the matrix operations can be there.

(Refer Slide Time: 31:16)



Now finally let us come to the application of arrays. Actually, the first example that we have mentioned it can be used to store, it can be stored using a 1 dimensional array. Here the idea is that, the idea is basically, 1 dimensional array basically starting from zeroth element and going 1, 1, 1, so this one say it is basically say 200 elements, so zeroth element the maximum values that is there can be stored and then the next location, the next values can be store.

Here again positive and negative, if it is above and below it can be stored. So, this way 1D array can be used to store this kind of information there. Now, here this can be used either 1D array or 2D array because all 2D arrays can be basically realized as 1D array. For example, as you have seen as a row major and column major order. So, here every line can be one rows and then in each character in each line can be considered as a column entries. So, this way the total is basically one 2D matrix can be considered to store the entire page.

Now, 3D array, because if we consider the whole image by a window of this size and then this window is basically divided by a large number of rows and columns, rows and columns like, so it is basically so many columns, likewise so many rows. And so, it is basically, the whole images needs to be divided into a large number of rows and columns. So, this way, so the enter image can be considered in the form of a matrix.

If it is a matrix, then we can store the elements those are in the matrix as a numbers. Now, what is a number? Actually, if you see each entry is basically is decided by a pixel value, now if we

find ground into the high number of rows and high number of columns, then definitely each pixel can be stored very minutely. So, basically is a image has its pixel level or intensity level is has values say maybe 0 to 55, total 256 values.

So, each entry in the matrices value between 0 to 55 if it is a colour again, the colour information. Because green, pink, yellow, black, red all these things colour, so for colour information if you want to maintain then you can consider 3D arrays. That means there are three dimensions; one is that in which location the pixel is there and for each pixel the values and then corresponding the colour value. So, colour value can be many colour R, G, B values can be store the red value, green value, blue value.

If it is a simple black and white picture, 2D arrays is fine, but if is a colour you can consider to store the colour information 1 dimensions. So, this way whatever the matrix, this kind of different type of data, we can store. Now, likewise we can extend the same concept to store the higher dimensional matrices, so store a video, video basically audio as well as, audio as well as images both together, but here the video frame.

So, each frame is basically either 3D arrays, then a collection of frames is basically is just like a collecting a books sort of thing, storing the entire video for any duration. So, anyway, so array looks apparently very simple but it is from the utility point of view it has enough applications. So, in the next video, we will discuss about the different way the array be programmed.

(Refer Slide Time: 35:24)

We have learned about the different algorithms like insertion, deletion, traversal, matrix operation. Writing program in Java is a very simple actually, so in the next video lecture has been planned to cover programming for arrays. Now if you want to know many things about arrays, then I should suggest you to consult the book classic data structure, which I have given the link here and you can go through and you can learn it. Thanks for your attention.