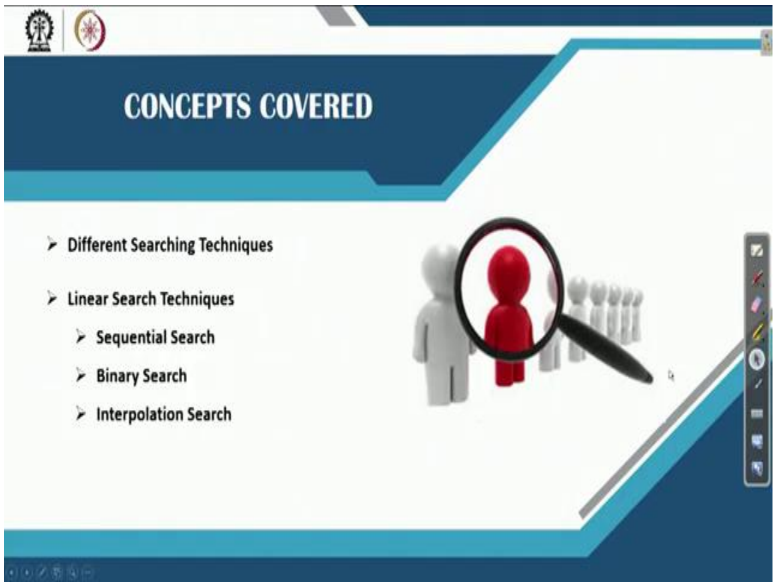
Data Structures and Algorithms Using Java
Professor. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur
Lecture No. 47
Linear Searching Algorithms

Many aspects of several data structures have been covered so far. Now, we will discuss about some algorithms which are extensively used on collections. Those collections easily can be stored in the form of an array or linked list or trees or whatever it is there. Now, we have also learned few algorithms in this regard, while we are discussing the different operations in order to manipulate the data structure or data in a given data structure.

Now, we will discuss in this module some algorithms which are related to some operations, of course, but we will give a specific algorithm those basically have received many attentions from the computer scientist to improve the best algorithm for the purpose, it is related to searching. So, our today's topic is basically searching algorithm. So, we will discuss the different concept.

(Refer Slide Time: 1:44)



CONCEPTS COVERED

- Different Searching Techniques
- Linear Search Techniques
  - Sequential Search
  - Binary Search
  - Interpolation Search

We will discuss the different concepts related to the searching, different searching techniques and there are varieties of searching techniques are known. So, in this class we shall discuss 3 important searching techniques namely sequential search, binary search and interpolation search. So, today it is our plan.

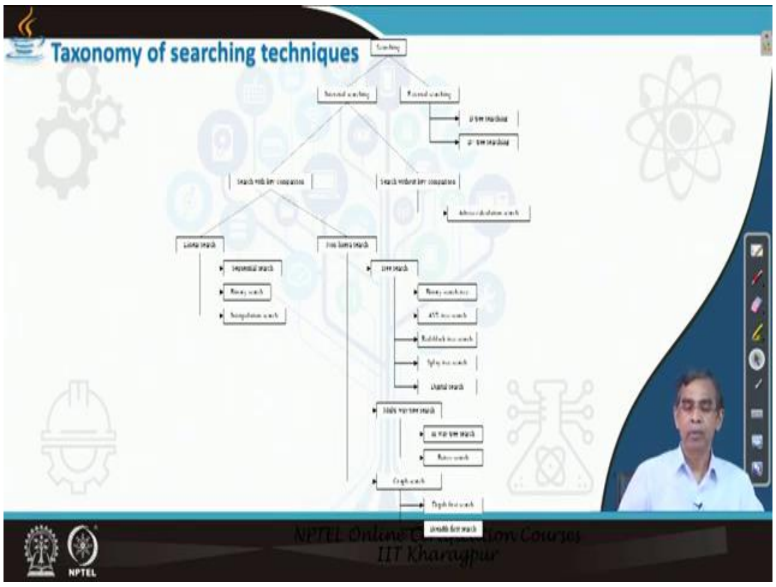
(Refer Slide Time: 2:18)

Different Searching Techniques

JAVA
NPTEL Online Certification Courses
IIT Kharagpur

Now, let us start about discussing the different searching techniques which are known till time. This topic in fact very vast, because since the inception of computer particularly the programming, there are many computer scientists have evolved with many algorithms. Every algorithms have their own benefits as well as certain limitations, but still there are process is on so that we can have better searching algorithms.

(Refer Slide Time: 2:52)



Taxonomy of searching techniques

Now, let us see the total story of several searching algorithms which have been right evolved till time and I have given a taxonomy of different searching technique, different searching technique actually follows certain principles. Anyway, whatever the sorting, searching techniques are there, they can be broadly classified into 2 categories, they are called internal searching and external searching. Now, external searching, when data is stored in memory, but it is in secondary memory. It is in hard disk like or floppy or some external storage devices like tape whatever it is.

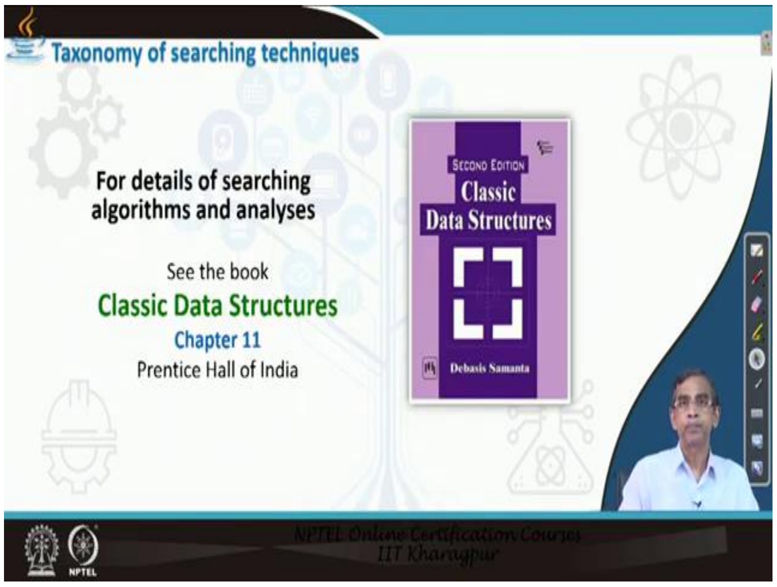
On the other hand internal searching techniques when data first loaded into your computer memory and then you apply search algorithms. Now, in this course, we will limit our discussion to only internal searching techniques, because external searching techniques needs the different treatment which is beyond the scope of this course. Now, again internal searching techniques, those are available can be broadly classified into 2 classes, they are searched with key comparison and another is searched without key comparison.

Now, so searched without key comparison means, basically we will not record to compare 2 keys while we are searching for the target key. So that is without key comparison, we will discuss how searching without key comparison can be accomplished. The next research with key compression. In this class, there are several searching techniques are known. Now, again also searching techniques belong to this category can be broadly divided into classes, one is called linear search, another is called non linear search. There are many techniques belongs to linear search, namely sequential search, binary search and interpolation search are the 3 most important searching techniques which are called searching technique belongs to that linear search.

Now, again nonlinear searching techniques there are many. In case of linear searching techniques, we usually store data into a linear data structures, linear data structure means maybe array or linked list. On the other hand, for the nonlinear data structure, the data are stored into a nonlinear data structure like tree or graph like this. Now, so if your data is stored in tree type data structure, then different searching techniques can be applied to which are for example, binary tree search or it is called the binary search, tree search, the AVL tree search, red black tree search, splay tree search, a digital search.

There are certain other searching technique where in lieu of binary it is the multi way searching. So, this is called the MA searching for example, BT search B plus T search in general they are called MA tree search. Now, again the data is typically stored in the form of a graph, some searching technique also can be applied, 2 searching techniques are very popular like depth first search and breadth first search. So, there are so many searching techniques are known.

(Refer Slide Time: 7:07)



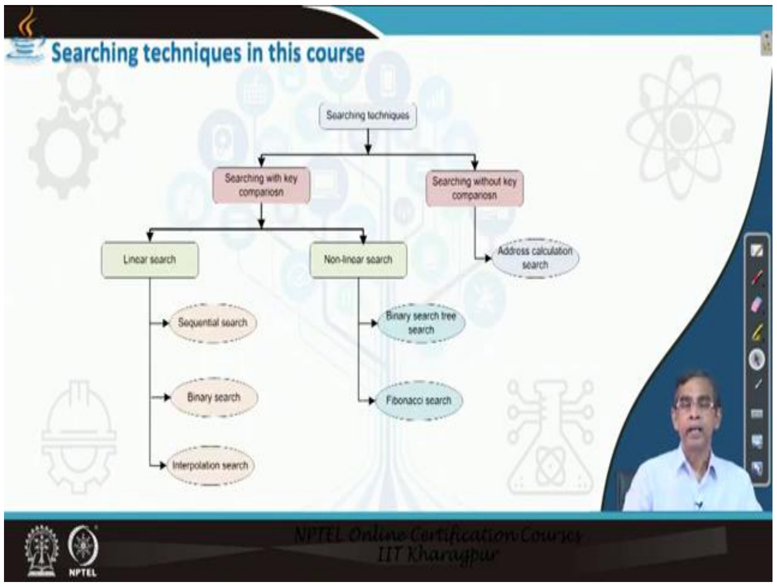
Taxonomy of searching techniques

For details of searching algorithms and analyses

See the book
Classic Data Structures
Chapter 11
Prentice Hall of India

However, in this course, it is not possible to discuss all the searching techniques. If you are interested to learn different searching techniques including external search and whatever the searching technique, then I should suggest you to follow the book Classic Data Structure chapter 11, where a thorough discussion about all searching techniques have been given. So, those are interested students, they can follow this book for their further studies.

(Refer Slide Time: 7:42)



Searching techniques in this course

Searching techniques
Searching with key comparison
Searching without key comparison
Linear search
Non-linear search
Address calculation search
Sequential search
Binary search
Interpolation search
Binary search tree search
Fibonacci search
NPTEL Online Certification Courses
IIT Kharagpur

In this course, as the time is limited, so we shall limit our discussion to only a few important searching techniques. I have mentioned what are the searching technique that I will cover in this course here. So, first of all searching technique with key comparison belong to this linear search and nonlinear search. Under the category of linear search, we will study about simple linear search, binary search and the interpolation search. Then, in the category of nonlinear searching technique, we will discuss about binary search tree search and then Fibonacci search.

Now searching without key comparison, we will discuss one technique, address calculation search. Address calculation search basically follow the hashing concept which we have already covered while we were discussing mapping. But here in the context of simple searching, we will discuss this concept. So, this discussion will complement already learned discussion while we were discussing regarding the mapping and tables theory.

So, these are the plan for this module, we will discuss different things, some portions will be discussed in today and some other parts in the next class.

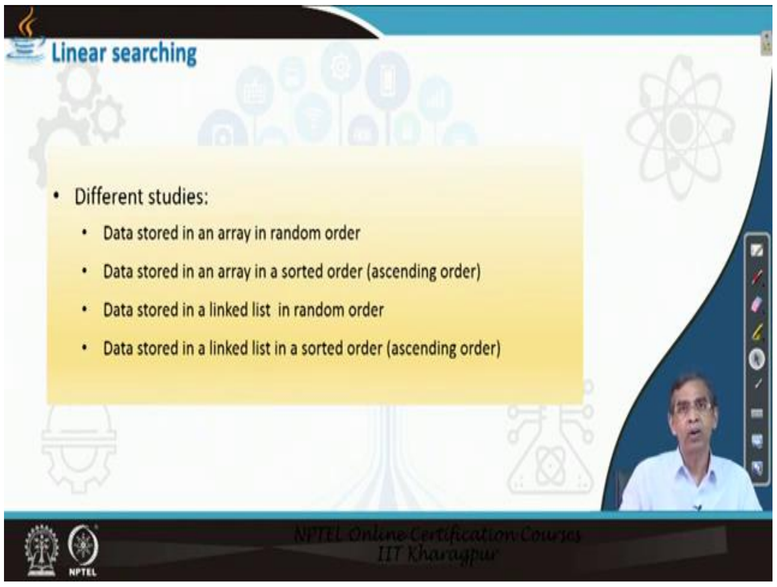
(Refer Slide Time: 9:03)

Linear Searching

So now, let us consider about linear searching. I told you linear searching is applied to data structure of type linear. So, if you store the elements, elements can be anything, it can be a number like integer or floating point number or sort, whatever it is, it can be string, it can be character, even it can be user defined objects of any type, but they are essentially stored in the form of an array. Now array we have already studied that can, that array is supported by means of different collections as we have already learned like arrays, array list, vector and so many things are there.

Even in your program also you can define your own array. So, whatever be that, the collection is basically stored in the form of an array and it is if we have to search an item stored in the array, then we can say that it is that linear searching techniques.

(Refer Slide Time: 10:13)



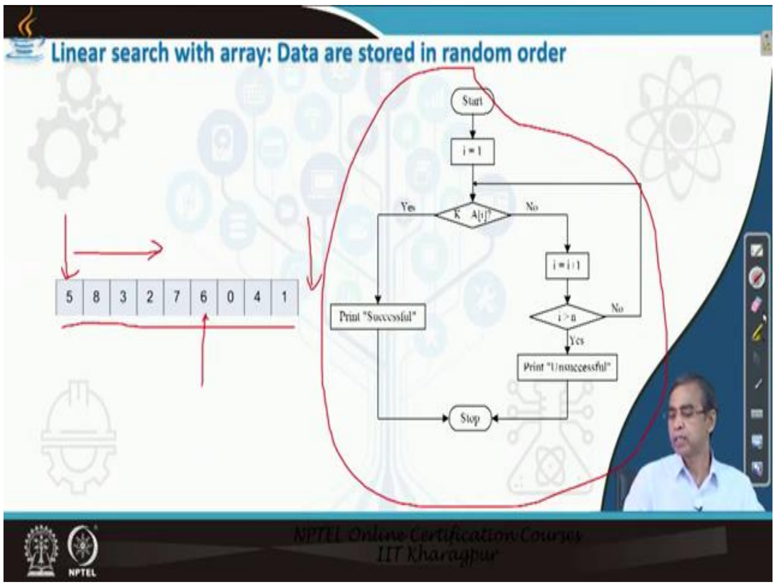
Linear searching

- Different studies:
  - Data stored in an array in random order
  - Data stored in an array in a sorted order (ascending order)
  - Data stored in a linked list in random order
  - Data stored in a linked list in a sorted order (ascending order)

So, data is stored in an array, there are now the proper question that arises is that how data are stored. Now, data can be stored in 2 different ways, one is that it is in order. Now this order may be in ascending order or descending order or some order. On the other hand, the data can be stored in a random order, no ordering. This is the one of possibilities that how data is stored. Now, that can, this kind of whether it is in random order or ascending or descending order, it can be stored in the form of a linear data structure.

Now, there are 2 versions of linear data structure, one is called array and another is also linked list. So, you can store the data into a linked list also in a random order or it can be in a particular order, maybe ascending or descending order. So, depending on the type, obviously data structure matters as well as arrangement of the data matters, different algorithms is possible.

(Refer Slide Time: 11:18)



Linear search with array: Data are stored in random order

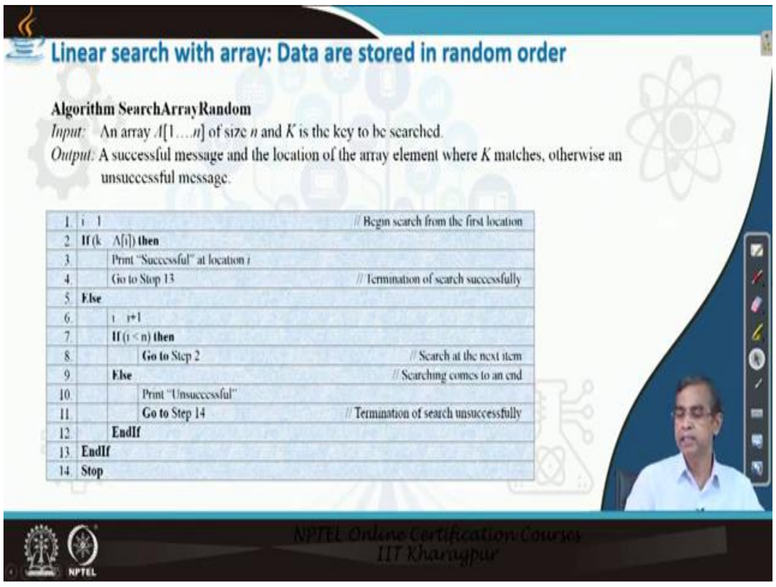
So, let us first discuss about the sorting, searching of data which is stored in a linear array and we will first consider the data stored in a random order. As in this example, we can say this is basically an array of integers and as you see the number integers are stored in a random order. Now, the algorithm that we can follow, which basically is shown here in this flowchart. Now, here the data is stored in a random array and we have to find one item or elements if it is stored or not.

So, so for example, the target key is 9. So what we should do, we should start from this position this may be 0th location or first location, whatever you can say. So, here in this flowchart it is again that first location is location I1. And now, we have to start with it and then we have to compare whether that target key that is our 9 is located here or not, if it is not here, then this pointer will move in this direction.

Now, if we traverse the entire arrays and the item is not found that when we reached here, then we can see the search is unsuccessful. Otherwise, if we find some element, which is here, then we can see the search is successful and in which location the element is available or present, it can be returned. So, this fact is depicted using this flowchart. So, this is very easy to understand if you start with you will be able to follow the logic that I told you how it can be done.

Now, so this is the concept that searching with key comparison and it is a linear data stored in the form of linear data structure like array and data are stored in random order. Now, we can extend these things writing an algorithm.

(Refer Slide Time: 13:27)



Linear search with array: Data are stored in random order

Algorithm SearchArrayRandom

Input: An array $A[1....n]$ of size n and K is the key to be searched.

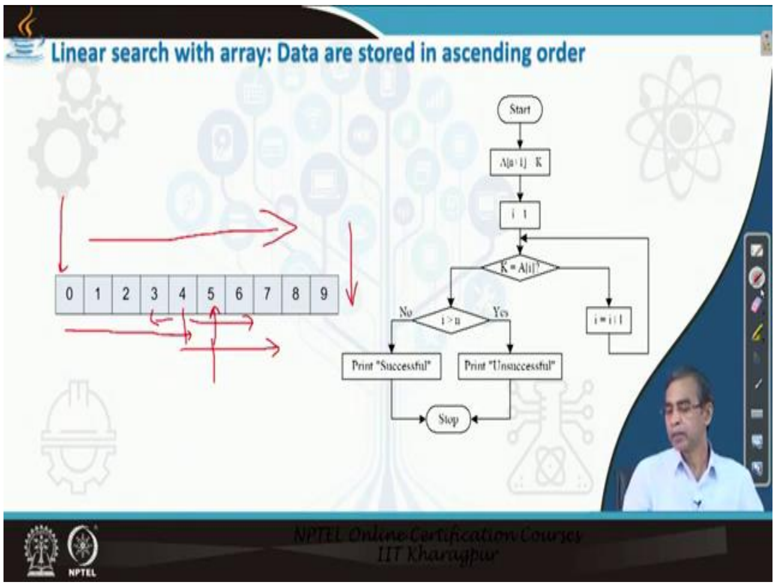
Output: A successful message and the location of the array element where K matches, otherwise an unsuccessful message.

```
1.  i   1                                    // Begin search from the first location
2.  If (k   A[i]) then
3.      Print "Successful" at location i
4.      Go to Step 13                        // Termination of search successfully
5.  Else
6.      i   i+1
7.      If (i < n) then
8.          Go to Step 2                     // Search at the next item
9.      Else                                 // Searching comes to an end
10.         Print "Unsuccessful"
11.         Go to Step 14                    // Termination of search unsuccessfully
12.     EndIf
13. EndIf
14. Stop
```

NPTEL Online Certification Courses
IIT Kharagpur

I have written an algorithm for you, if you follow this algorithm, which is given in this table, it will be easy for you to write the code. So, writing the code for this particular linear search is very trivial, anybody can having some programming aptitude they can write this. So, I should not discuss about this programming, whenever I come to the programming of different searching algorithm there I will point it out. So, now you can go through this because it is so simple that no need for further discussion.

(Refer Slide Time: 13:53)



Linear search with array: Data are stored in ascending order

NPTEL Online Certification Courses
IIT Kharagpur

Now, let us consider the another case where the data is stored in an array, some numbers for example, and it is already sorted in ascending order. So, like the previous searching algorithm, it is basically same. Now here we start from the first location. So, this is the starting location and the target key needs to be compared. If it is available here, then we will stop it here, we should not go past that. Otherwise you will scan it and if we do not find the elements here, then we can reach here at the end of the array, giving us the search is unsuccessful, otherwise search is successful.

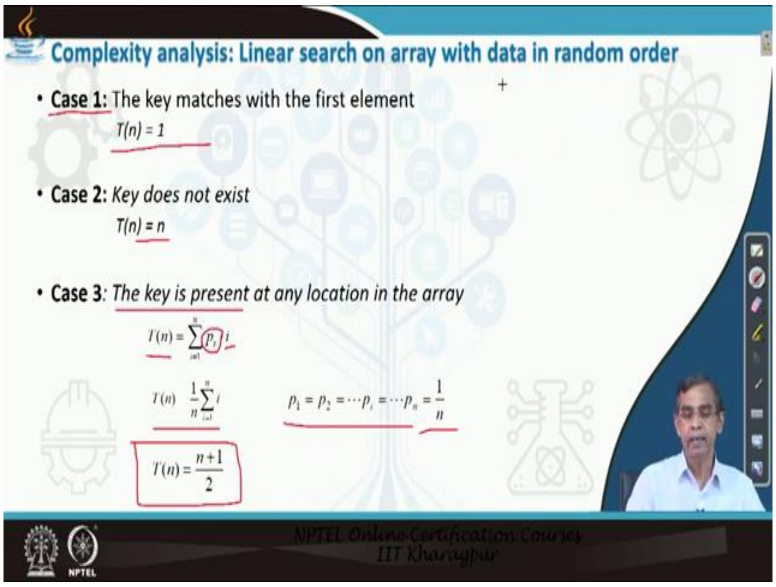
Now idea is more or less similar to the previous algorithm. But here actually, so here if you find the element or if you see that your element, which is the element that you want to search is already exceeds the limit. For example, you want to search a 5, for example 5. So, we can start we can start from here, we can reach here and then we can find that 5 is available here. Now, suppose we are searching for some numbers say 3.5.

So, we can start again 3, now whenever we come 4, we can say that moving this direction will not be fruitful, because the element that we have searching are not possible to be here because we have already crossed the limit. So, because 3.5 should be here in this side only, not in this side. So, we can stop the searching. That is the only difference from the previous one and this one.

Because in case of previous one, as data was in random order, we have to start searching whenever it is available or not available. So, this is the only difference that you have to take care in this case and accordingly, you can decide the algorithm. I have given an algorithm for you, this is a flowchart and then the algorithm is you can follow this algorithm. The same algorithm simple logic and it is easy to implement writing program will not be a big issue for any programmer.

So, this you can try to write a program, storing the elements in an array and in a sorted fashion and you can repeat the same thing starting with ascending order as well as descending order. Logic will work in same way, but you can check it, if the data is stored in ascending or descending order whatever it is there. So, you can just have the idea about simple linear search, it is really simple, because idea is so simple and logic is also not so complicated and you can do it.

(Refer Slide Time: 16:45)



Complexity analysis: Linear search on array with data in random order

- **Case 1:** The key matches with the first element

  $T(n) = 1$

- **Case 2:** *Key does not exist*

  $T(n) = n$

- **Case 3**: *The key is present at any location in the array*

  $T(n) = \sum_{i=1}^{n} p_i \cdot i$

  $T(n) \; \frac{1}{n}\sum_{i=1}^{n} i$ $\quad p_1 = p_2 = \cdots p_i = \cdots p_n = \frac{1}{n}$

  $\boxed{T(n) = \frac{n+1}{2}}$

NPTEL Online Certification Courses
IIT Kharagpur

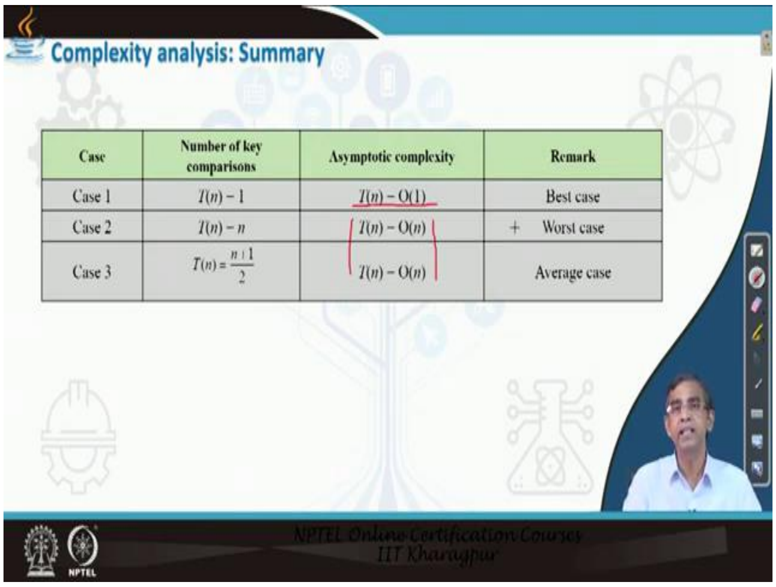
Now let us come to the analysis of the searching algorithm. So, what is the time complexity or the running time that is required? Now, we are assuming that data first of all stored in a random order in an array. So in the best case, so this is a case one which we can see the best case. In case of best case, that total time that is required you can say one unit, I am telling unit because it is basically only to search the first element and stop it there are assuming that the element that is your target element is available in the first location itself.

So, this is the best case or ideal case of course. Now, on the other hand, suppose the element is not present in the list then you have to traverse starting from 1 to n. So, total cost of traversal is basically you can say n unit or you can say the total time complexity n. And on the other if k is present, but if k is present, it can be located at any position. So, if it is available at ith position and pi is the probability that the element will be in ith position, then we can say that this is a total time that is required.

Now, so now, we can take the summation for i equals to 1 taking the average. So, it will give the average time that is required, assuming that P 1 to P 2 n are equal probabilities, that means element, probability that the element will be located at any position are equal probability. So, then we can say that all probability values are 1 by n. So, this calculation therefore, can be calculated in a simple, is that n plus 1 by 2. So, on the average half of the elements needs to be moved or traverse actually, this is the best the idea actually there.

So, these are the 3 different cases. This is the best case, this is basically the worst case and this is the case it is called the average case, on the average it will take like this. So, over all the complexity that actually time complexity what you can say is it is the order of n, actually that is the asymptotic complexity. Now, I have listed all the asymptotic complexity in this table for your summary, you can follow it.

(Refer Slide Time: 18:57)



Complexity analysis: Summary

Case	Number of key comparisons	Asymptotic complexity	Remark
Case 1	$T(n) = 1$	$T(n) = O(1)$	Best case
Case 2	$T(n) = n$	$T(n) = O(n)$	+ Worst case
Case 3	$T(n) = \frac{n+1}{2}$	$T(n) = O(n)$	Average case

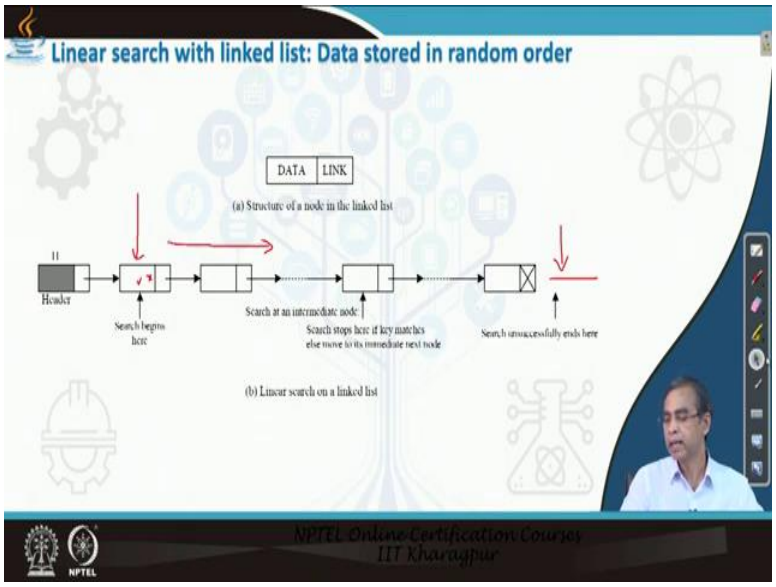
NPTEL Online Certification Courses
IIT Kharagpur

So, the case 1 that is the best case as I told you, T n equals to 1, then complexity it is an asymptotic complexity notation or big O notation it is called. If you are not aware about that big O notation, you should have this concept actually, any algorithm book can be consulted including this Classic Data Structure also you can follow how to express the complexity of an algorithm that is called synthetic complexity. There are many notations, big O notation, theta notation, big omega notation, small theta notation like this.

So, that actually I advise you to go to a special chapter 11 there and you can find the complexity notation for the different algorithms that you can have. Now, anyway so this is the T n in big O notation order of 1 because it is required constant time whatever be the size of the array, it is constant only the fixed time it is required. But other techniques whenever the other cases are that based, other than best case that means worst case and average case, the complexity is in the order of n.

This means that that time, running time of the algorithm will increase linearly as the number of elements stored in the array increases. So, this is basically linear time complexity we can say. So, these are the complexity analysis, details how these complexity analysis can be calculated that you can follow the book Classic Data Structure for detailed discussion. So, this is the idea about the simple linear search, rather we can see simple technique.

(Refer Slide Time: 20:32)



Linear search with linked list: Data stored in random order

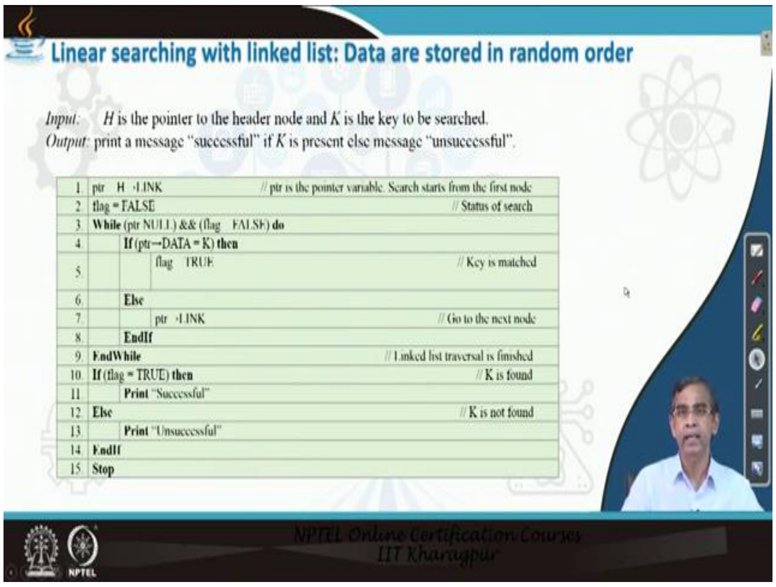
DATA
LINK
(a) Structure of a node in the linked list
Header
Search begins here
Search at an intermediate node
Search stops here if key matches else move to its immediate next node
Search unsuccessfully ends here
(b) Linear search on a linked list
NPTEL Online Certification Courses
IIT Kharagpur

Now, let us come to the discussion about again another linear search technique, but when the data is stored in a into a linked list data structure. We are already familiar to linked list data structure and possibly we have discussed about the search operation there, we defined one data structure called linked list where we defined many methods insertion, deletion, possibly searching also included there also.

Anyway the same concept is there, it is similar to the linear array because like this, but we have to traverse not consecutive memory location rather to traverse to the next node depending or following the link that is maintained by the previous node. So, this is the concept which is there. Now searching it will take place as usual starting from the first node you can first check whether your target element is present here or not if not present here, then we have to move in this direction and we have to repeat this until we find the element in a particular node or we reach to the end of the node, I mean end of the list rather we can say.

So, in this case the searching is unsuccessful, otherwise searching is successful. Again, we are assuming that data here stored not in an order, it is in random order. So, this is the basic technique and then an algorithm also you can follow.

(Refer Slide Time: 22:06)



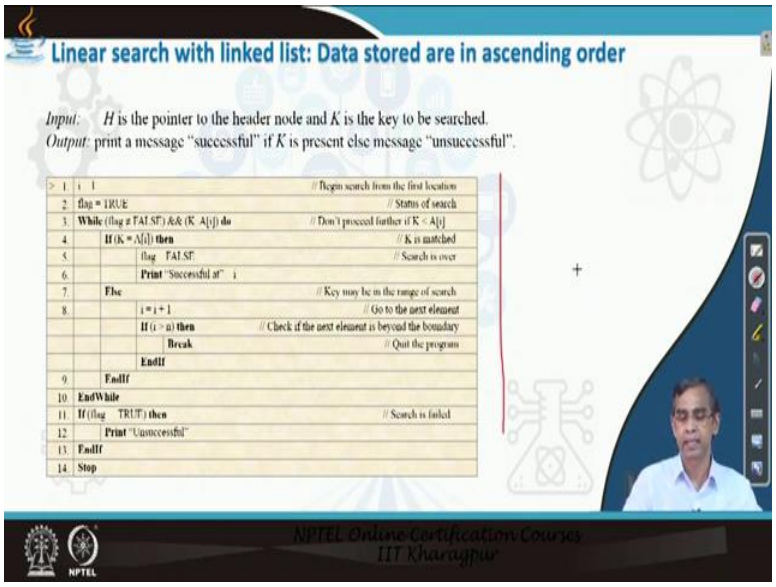
Linear searching with linked list: Data are stored in random order

Input: *H* is the pointer to the header node and *K* is the key to be searched.
Output: print a message "successful" if *K* is present else message "unsuccessful".

```
1   ptr   H →LINK                          // ptr is the pointer variable. Search starts from the first node
2   flag = FALSE                           // Status of search
3   While (ptr NULL) && (flag   FALSE) do
4       If (ptr→DATA = K) then
5           flag   TRUE                    // Key is matched
6       Else
7           ptr →LINK                      // Go to the next node
8       EndIf
9   EndWhile                               // Linked list traversal is finished
10  If (flag = TRUE) then                  // K is found
11      Print "Successful"
12  Else                                   // K is not found
13      Print "Unsuccessful"
14  EndIf
15  Stop
```

This algorithm is given for you, so that you can implement this algorithm the same algorithm we have possibly implemented while we are discussing about the linear data linked list structure. This algorithm you can again pretty simple that can be implemented easily.

(Refer Slide Time: 22:20)



Linear search with linked list: Data stored are in ascending order

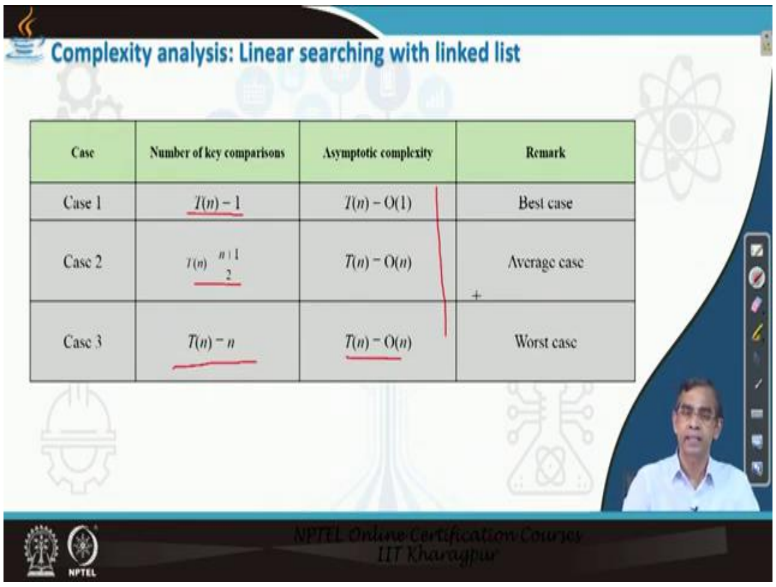
Input: *H* is the pointer to the header node and *K* is the key to be searched.
Output: print a message "successful" if *K* is present else message "unsuccessful".

```
1   i   1                                  // Begin search from the first location
2   flag = TRUE                            // Status of search
3   While (flag ≠ FALSE) && (K  A[i]) do   // Don't proceed further if K < A[i]
4       If (K = A[i]) then                 // K is matched
5           flag   FALSE                   // Search is over
6           Print "Successful at"   i
7       Else                               // Key may be in the range of search
8           i = i + 1                      // Go to the next element
            If (i > n) then                // Check if the next element is beyond the boundary
                Break                      // Quit the program
            EndIf
9       EndIf
10  EndWhile
11  If (flag   TRUE) then                  // Search is failed
12      Print "Unsuccessful"
13  EndIf
14  Stop
```

And when that data are stored in an order, assume that in ascending order how the algorithm works, it is mentioned in this algorithm, this algorithm you can follow. It is basically the same as the linear data structure with array that we have discussed, start from the first node and we can move to the next depending on 2 conditions. One condition is that element is not found or element is less than the target element that is we are going to find it. So, whatever it is there we have to continue.

And we stop under two situations when we found the element there or we reached to the end of the list. So, these are the condition and this algorithm is basically ported the same concept and you can follow it easily, this algorithm has English like sentence not exactly in Java programming syntax, but you will be able to easily convert this into Java programming syntax. So, this is again writing program for this is left as an exercise for you so that you can practice it.

(Refer Slide Time: 23:25)



Complexity analysis: Linear searching with linked list

Case	Number of key comparisons	Asymptotic complexity	Remark
Case 1	$T(n) - 1$	$T(n) - O(1)$	Best case
Case 2	$T(n) \quad \frac{n+1}{2}$	$T(n) - O(n)$	Average case
Case 3	$T(n) - n$	$T(n) - O(n)$	Worst case

Now, here the time complexity for this purpose as you can see is similar to the previous one. So, this is the best case is a fixed time. This is basically the average case, as a unit of n plus 1 by 2 half of the least to be traverse actually in general and worst case, this basically either element is available at the last node or is not available in the thing. So, that worst-case is this one. So, this is the time complexity. So, what we can say the time complexity for linear search and your linear search technique with array and with linked lists are almost same.

So, they are a similar complexity. That means we can say that the algorithms which we have considered for searching over an area or overlaying are same level of complexity or running time will be almost same. And in both cases it will be basically proportional to the number of elements stored in the data structure. Now, so we have discussed the different searching technique, linear searching technique rather when data stored into a linear data structure, linear data structure namely array or linked list.

(Refer Slide Time: 24:42)

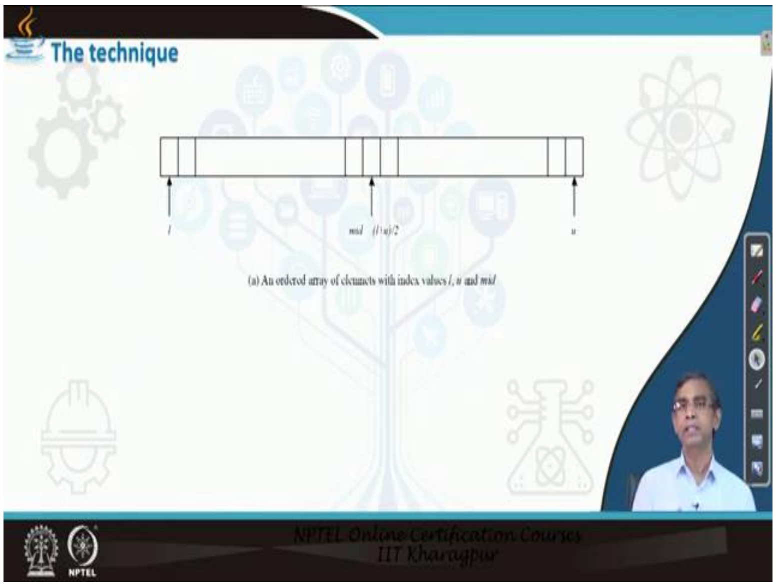
Binary Search Technique

NPTEL Online Certification Courses
IIT Kharagpur

Now we will consider another better searching technique. It is basically the same that element is stored in an array. We can apply the same thing on the linked list also, but popularly this technique is applied to data structure when data stored in an array data structure. And another condition we will follow for the searching technique is that elements are stored in an array and already in sorted order.

So, if the elements or data is stored in an array and in sorted order, then we can devise a better searching technique, which will work better than the linear searching technique that we have discussed. Now, let us follow the techniques that we are going to learn about how it works actually.

(Refer Slide Time: 25:36)



The technique

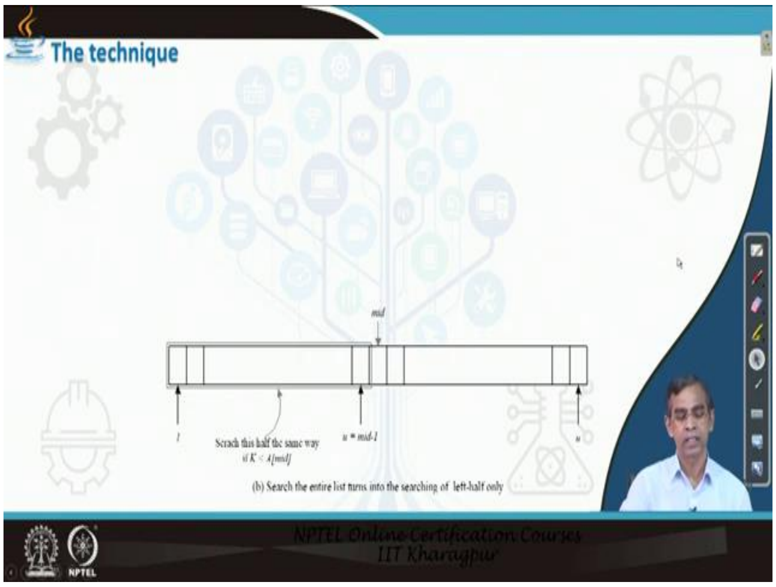
(a) An ordered array of elements with index values l, u and mid

NPTEL Online Certification Courses
IIT Kharagpur

So, the idea it is like this, it is already sorted. So, we know that first location of the list and the last location of the list. So, past what we should do is that we have to hit at that middle location. So, hit at the middle location means which is at the middle position of the array. Now, your target key can be compared with the element which stored in the middle position. Now, two situations will occur, one is that element is present, therefore matches, so stop the searching.

Another thing is that element does not match with them elements which is at the middle location. Then what we should do we can repeat the same procedure but not on the same array, rather on half of the array. Now, which half you have to decide that whether the element the target element which is less than the element which is at the middle or is greater than which is at the middle. If the target element which is less than the element at the middle, then we should start the next part of the location.

(Refer Slide Time: 26:50)



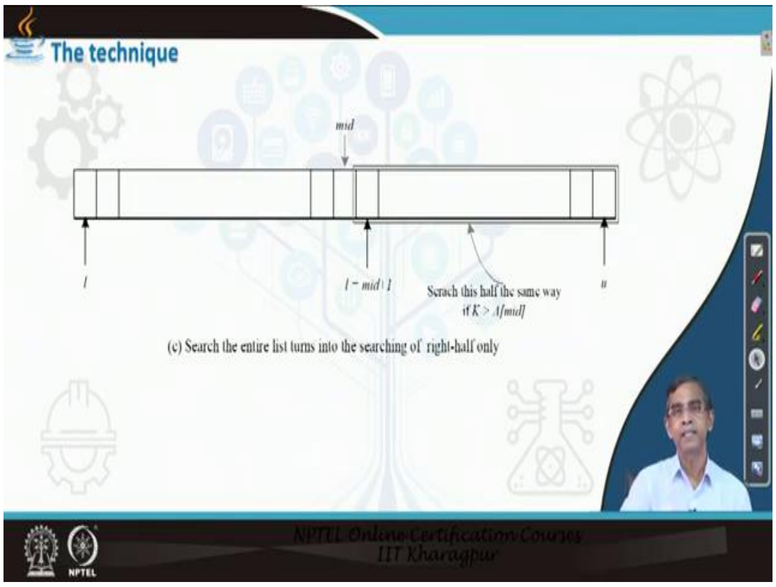
The technique

mid
l
Scrach this half the same way
if $K < A[mid]$
$u = mid-1$
u
NPTEL Online Certification Courses
IIT Kharagpur
NPTEL

(b) Search the entire list turns into the searching of left-half only

So, this part. So, in this part we will search it. On the other hand if the target element is greater than the element which is presented the middle location then we should search the element are the rightmost part of the list.

(Refer Slide Time: 27:06



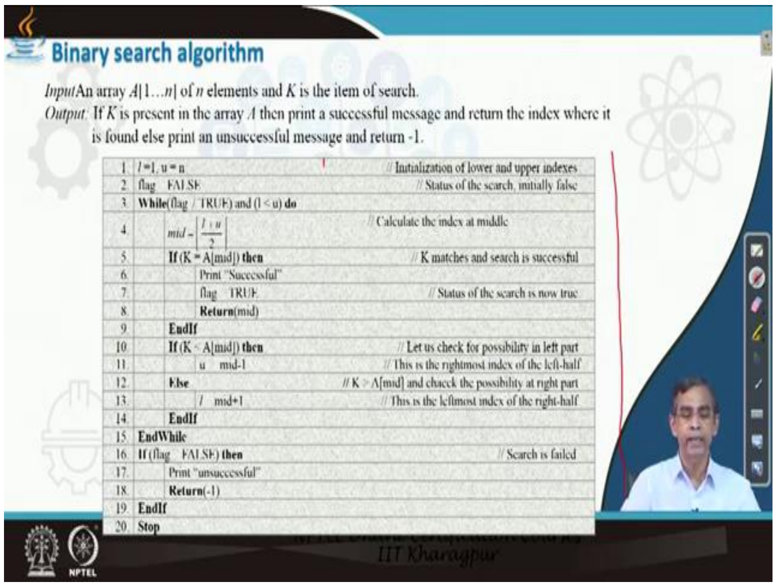
The technique

mid
l
$l = mid + 1$
Scrach this half the same way
if $K > A[mid]$
u
NPTEL Online Certification Courses
IIT Kharagpur

(c) Search the entire list turns into the searching of right-half only

So, it depends on that target element and this procedure will continue until we come to a list which is no more splitable, I mean we will not be able to divide into 2 parts, that means it contains only one element or you already find an element that means matches soccer. So this is a case that you can think about it. So, each time you split the array into 2 parts, at the middle compare, then either check the left part or right part depending on the number that you are searching. So, this basically the idea about the binary search technique.

(Refer Slide Time: 27:55)



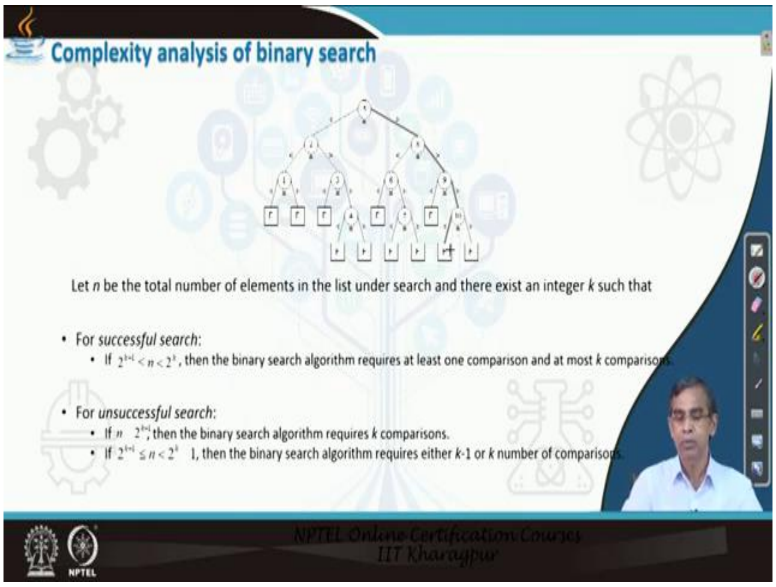
Binary search algorithm

Input: An array $A[1\ldots n]$ of n elements and K is the item of search.
Output: If K is present in the array A then print a successful message and return the index where it is found else print an unsuccessful message and return -1.

```
l = 1, u = n                                   // Initialization of lower and upper indexes
flag  FALSE                                    // Status of the search, initially false
While(flag ≠ TRUE) and (l < u) do
    mid = ⌊(l + u)/2⌋                          // Calculate the index at middle
    If (K = A[mid]) then                       // K matches and search is successful
        Print "Successful"
        flag  TRUE                             // Status of the search is now true
        Return(mid)
    EndIf
    If (K < A[mid]) then                       // Let us check for possibility in left part
        u  mid-1                               // This is the rightmost index of the left-half
    Else                                       // K > A[mid] and chaeck the possibility at right part
        l  mid+1                               // This is the leftmost index of the right-half
    EndIf
EndWhile
If (flag  FALSE) then                          // Search is failed
    Print "unsuccessful"
    Return(-1)
EndIf
Stop
```

It is very popular searching technique. The algorithm, the procedure that I told you is stated a more systematic manner in an algorithmic fashion. So, this algorithm is pretty simple. And if you follow will be able to understand how this algorithm actually work. And finally, writing program following this algorithm is also very simple, it is not a tedious job, it is also advised to you that you can write, try to write the program based on this algorithm. So that you can practice much better and learn the concept much better.

(Refer Slide Time: 28:33)



Complexity analysis of binary search

Let *n* be the total number of elements in the list under search and there exist an integer *k* such that

- For *successful search*:
    - If $2^{k-1} < n < 2^k$, then the binary search algorithm requires at least one comparison and at most *k* comparisons.
- For *unsuccessful search*:
    - If $n \quad 2^{k-1}$, then the binary search algorithm requires *k* comparisons.
    - If $2^{k-1} \le n < 2^k \quad 1$, then the binary search algorithm requires either *k*-1 or *k* number of comparisons.

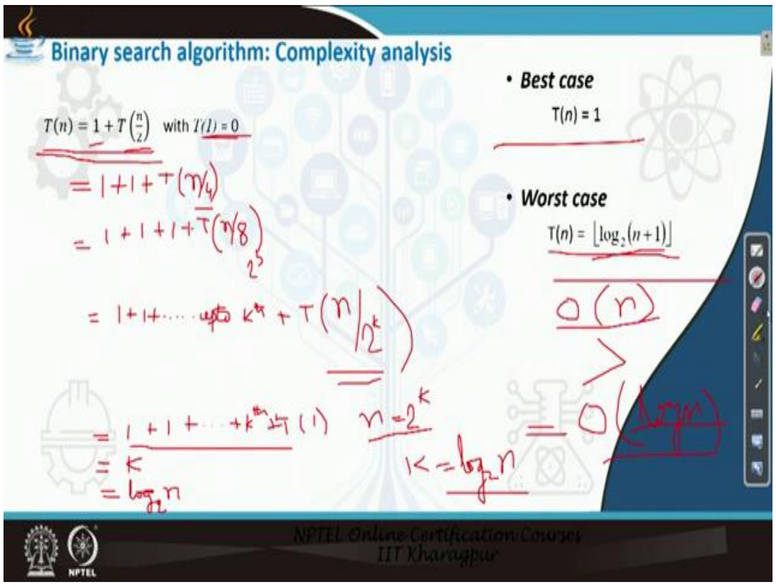
NPTEL Online Certification Courses
IIT Kharagpur

Now, the complexity analysis. The detailed complexity analysis is a very difficult job. I will just tell you that binary search basically follow a recursion tree like this one. So, this is the middle location then if you find it, if you find it stop here, if it is not find it and the target element is less than then you go there and so on so on. So, this basically the idea about we have given the idea about what are the elements of size 10, I mean array of size 10 and then how it will occur.

So 8, depending on the failure case, otherwise a few find it the equal cases is there. Now, so this is the idea that is the there. So, it basically is a tree and so whenever you search, the longest path that is in the tree is basically total complexity. And you know this is just like a binary tree and the longest path that is possible and it is almost height balanced tree you can say. It is basically height balance or we can say in this case for example, it is a complete binary tree.

So, complete binary tree is good for minimum height that you already know. So in this case, the height will be the maximum path length that is possible and which is basically log n, if the n number of nodes are stored in that tree like. So, this is basically log n and considering these things, the search complexity of this binary search tree is basically order of log n.

(Refer Slide Time: 30:04)



Binary search algorithm: Complexity analysis

$T(n) = 1 + T\left(\frac{n}{2}\right)$ with $T(1) = 0$

$= 1 + 1 + T(n/4)$

$= 1 + 1 + 1 + T(n/8)$

$= 1 + 1 + \cdots$ upto k^{th} $+ T(n/2^k)$

$= 1 + 1 + \cdots + k^{th} + T(1)$ $\quad n = 2^k$

$= k \quad k = \log_2 n$

$= \log_2 n$

- *Best case*

  $T(n) = 1$

- *Worst case*

  $T(n) = \lfloor \log_2 (n+1) \rfloor$

$O(n) > O(\log_2 n)$

NPTEL Online Certification Courses
IIT Kharagpur
NPTEL

Now, and in case a best case means if you find element which is present at a middle location. So, only one probe required, the best case is basically this complexity and the worst case is basically this complexity. And a recurrence relation that we can write, recurrence relation is basically the, this is the task that is required to find a location, that means middle position of the list. And then once you calculate the middle position, you can stop it or you can continue the same thing but with the half of the list.

So, recurrently this expression can be says that Tn is equals to 1 plus T n by 2. Now we can express this again this is basically 1 plus 1 if it is not found in the either left part or right part then you have to repeat it again. But in this case, your repeat will go on for the one fourth of the list. So, each time as you progress, so number of size of the part of the list is reduced. Now, if you suppose not available in the one fourth of the part, then we have to continue again.

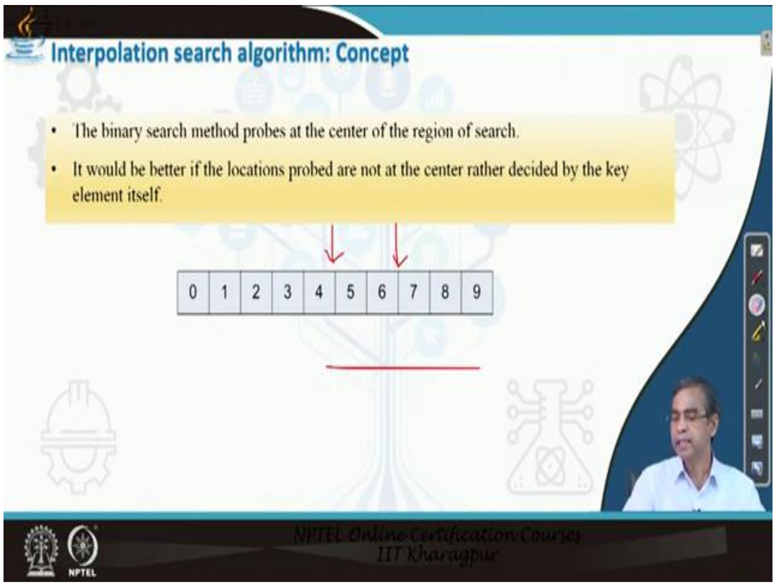
So now, next again so recurrently we are resolving then in that case it is basically, so T and it will go n by 8 we can say, so in by 8 means you need 2 to the power 3. So, it is 2 to the power 2, 2 to the power 3 and if we continue this way, so 1 plus 1. So, it is up k, k number of 1 up to k kth 1 and the next will be T n by 2 to the power k. Now, the boundary condition is T 1 that is possible when n equals to 2 to the power k.

So, suppose n equals to 2 to the party then what you can write is there 1 plus 1 plus up to plus kth 1 plus T 1. Now, here T 1 is equal to 0 so we can stop it here. So, we can say that this is equals to k because the k number of 1s are there. Now here, so k equals to, this is equals to log 2 n. So, this is basically we can say the complexity is log 2 base n. So, this is the time complexity which is written here basically, but it is taking the floor, so log 2 n plus 1.

Because here we assume that n is exactly divisible by 2 or is a power of 2, but it may not be like that. So, if it is not like then you have the complexity like this one. Now, here we assume that n is exactly a power to actually. Anyway so this basically gives an idea about the time complexity this one, now log 2 n. Now you compare the searching algorithm that we have discussed when data is stored in an array using simple linear searching technique, it takes the order of in.

So is basically, so this algorithm the complexity of this algorithm we can say order of log n and the previous time complexity that we have discussed is order of n. Now, actually this order of n, this is greater than this one, that means this algorithm is better than this algorithm. So, this is the idea about the sorting technique, searching technique and binary search technique that we have discussed.

(Refer Slide Time: 34:14)



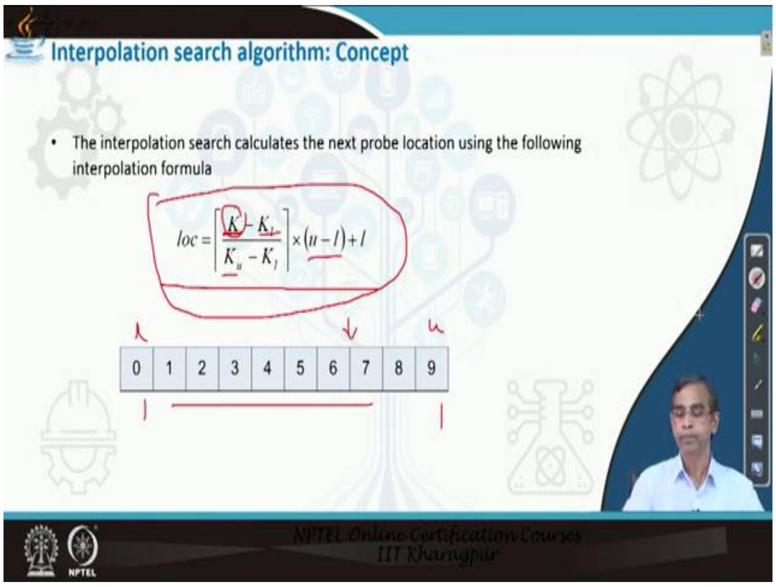
Interpolation search algorithm: Concept

- The binary search method probes at the center of the region of search.
- It would be better if the locations probed are not at the center rather decided by the key element itself.

Now, let us consider the another searching technique it is called the interpolation search interpolation search is basically an improved version of the binary search technique. In this search technique, the difference between the binary search an interpolation search is that almost a technique is same, but in case of binary search, we have to hit at the middle element, s elements at the middle position. But, but but in case of interpolation search, it is not exactly the middle basically we have to hit at the position which is very close to the target element if any.

So for example, you have to search for say 7.5. So, in case of binary search it will probe here but in case of interpolation search, it will probe here. So, this that the number of iterations or parts that is required only parlays in case of interpolation, then the binary search technique actually. So, this is an improvement. Now, let us see how we can prove which is very close to that target element actually. Now, so in this regard there is a interpolation formula calculation, interpolation formula is there, so we can follow this interpolation formula and can apply it.

(Refer Slide Time: 35:32)



Interpolation search algorithm: Concept

- The interpolation search calculates the next probe location using the following interpolation formula

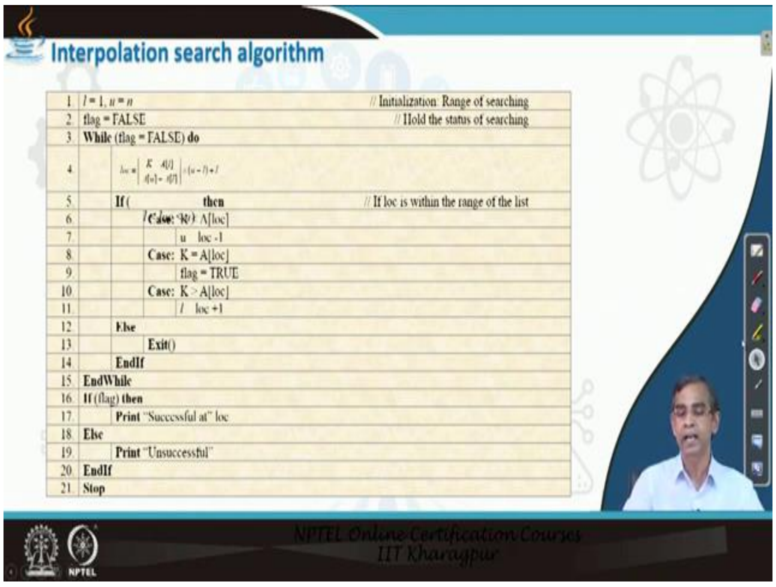
$$loc = \left[\frac{K - K_l}{K_u - K_l}\right] \times (u - l) + l$$

0	1	2	3	4	5	6	7	8	9

This interpolation formula to check that which is the possible location which is very close to the target key. So, k is the target key and Kl is the basically the bound in the lowest one and U is the upper upper one. So, in this series this is the K l and this is the K u and any elements basically your target element that you want to find, whether this is there in the list or not and u minus l is basically location.

So, this is l and this is u. So, this formula can give you to find for a given key which is the nearest location that we have to probe or hit. So, this formula is basically important and based on this concept, otherwise, everything is same as binary search algorithm. Only that problem is that here we have to calculate the position where we should hit instead of pretty simple location the middle location that is followed in binary search technique.

(Refer Slide Time: 36:31)



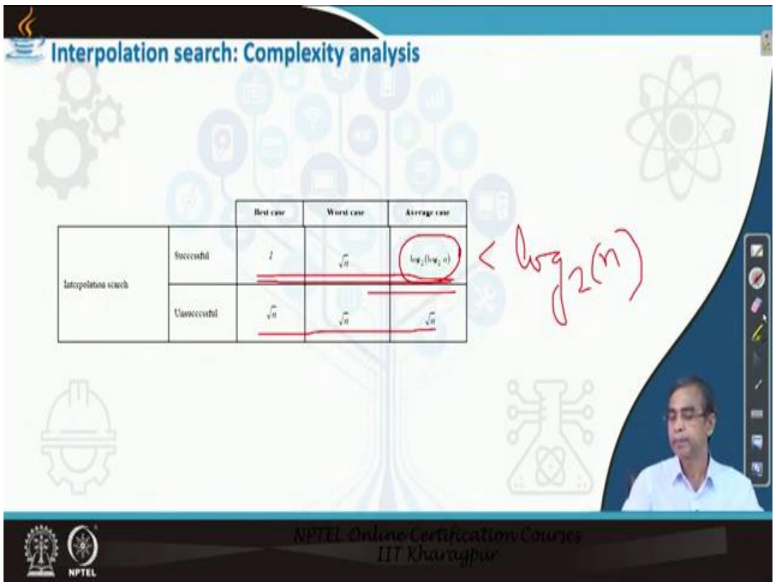
Interpolation search algorithm

```
l = 1, u = n                                         // Initialization: Range of searching
flag = FALSE                                         // Hold the status of searching
While (flag = FALSE) do
    loc = ⌊ (K − A[l]) / (A[u] − A[l]) × (u − l) ⌋ + l
    If (              then                           // If loc is within the range of the list
        Case: K [illegible] A[loc]
            u   loc -1
        Case: K = A[loc]
            flag = TRUE
        Case: K > A[loc]
            l   loc +1
    Else
        Exit()
    EndIf
EndWhile
If (flag) then
    Print "Successful at" loc
Else
    Print "Unsuccessful"
EndIf
Stop
```

NPTEL Online Certification Courses
IIT Kharagpur

The algorithm that you can follow it is there. The algorithm is written here, you can follow this algorithm same algorithm only this party is different, the calculation, here not middle hit actually the interpolation location calculation other other parts are the same.

(Refer Slide Time: 36:44)



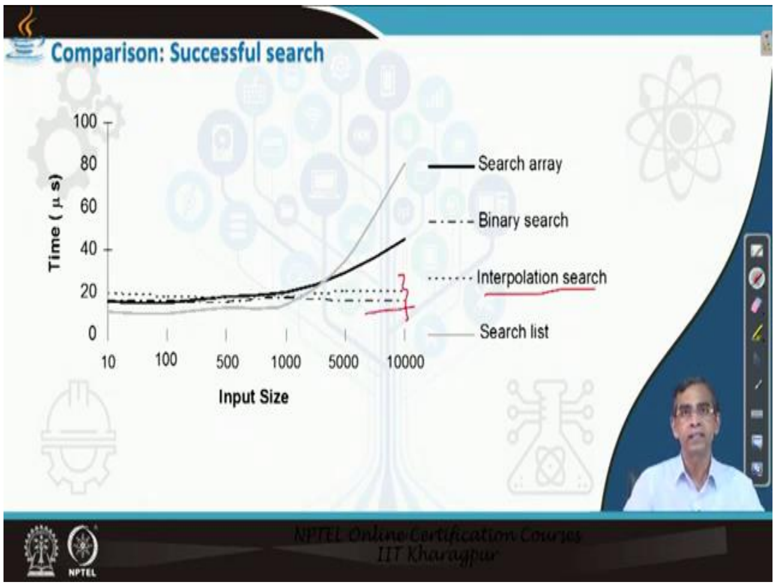
Interpolation search: Complexity analysis

		Best case	Worst case	Average case
Interpolation search	Successful	1	$\sqrt{n}$	$\boxed{\log_2(\log_2 n)}$ $< \log_2(n)$
	Unsuccessful	$\sqrt{n}$	$\sqrt{n}$	$\sqrt{n}$

NPTEL Online Certification Courses
IIT Kharagpur

Now, so far the complexity analysis is concerned, detailed complexity analysis you can follow any book, Classic Data Structure also you can consider you can find a detailed discussion. If the element present in the list, then depending on the best case worst case and average case time complexity will be like this. And if it is unsuccessful, then this is one. Now we can say that these complicity is on the average is log 2 n, now which is very very less than log 2 n actually. So, this means that this algorithm is better than the binary search algorithm that we have discussed. So, this is the better algorithm and a program you can write it.

(Refer Slide Time: 37:25)


Comparison: Successful search

`While programming we will discuss while we will discuss these technique details there, programming list will be discussed in another class. Now, I have given a big comparison about the different searching that we have discussed about. Now, as we see, the searching over a list, either linked list or it is in maybe all elements are stored in an array, we can assume that it is in a sorted array because the interpolation require that elements should be in sorted order, binary search also require that elements should be in sorted order.

Now, here we can see if we follow the linked list that as the time increases, I mean the number of elements increases the complexity increases x increases actually. If it is not exponentially, but it is increases with time. On the other hand, this algorithm is basically the algorithm for, so this algorithm is basically the interpolation search and this is the time complexity that is required for the binary search. As we see the interpolation search takes little bit more time than the binary search.

And in this binary search is very fast actually, although complexity is to log 2 n this is because in case of interpolation search, address calculation is basically is heavy that is how it is taking time. But anyway both the algorithms are comparable and it takes more or less same time and you can see that we as the number of elements increases, it does not increases exponentially as it increases for the list over the search it is there. So, this way we can just we can take we can see that which search algorithm is better and that is why interpolation search is not so better because of this address calculation.

However, binary search is the best searching algorithm so far the linear data structure is concerned. That means if you store the data in the form of a array or in a linked list, you can follow the linear searching techniques. Now I have given some basic concept about the linear search techniques when that data is stored in an array or linked list. There are a few more techniques also available in this book. I suggest you to follow the book for further studies. Thank you very much.