for yourself. If you know even a bit of maths, You 'll know that it is in an ascending order. Now what I say is that I 'll give a number : 'A' And I 'd like you to tell me If this number exists within the array , or not. Suppose the value of A is 8. So what will be your answer ? Yes. Meaning 1. If A is. . . Sorry, your answer will be no , because it is n't there. If the value is 9, What will the answer be ? Your answer will Algo 1 is a simple person. It does n't have much of a brain. It is comparing it with all the numbers. Is this the best way to do this work ? Obviously not. Because Algo. 1 is lucky , He will get A=1. It will tell us in the first comparison itself. In one comparison only.

If Algo 1 is in luck, The time needed is ' k ' - T=k. This means that it does n't depend on 'n '. Take a 10-element array, take a single element array or take a 10,000 element array. It only has to make one comparison because it is only searching for the first element in the array. Now, AlGo 1 's luck is bad. Till now, he was fortunate ; But now he 's not so lucky anymore. Average Case complexity is equal to. . . the sum of the run time for the total number of possibilities. The O (Sum of all possible run times divided by the number of possibility) is O (n) The average case complexity is the sum. Average Case is equal. to. . . The sum of all. possible run. times divided. by the total. number of possible run time. So for an array size of 5, We saw six cases at ' ; I 'll calculate ' O ' later. n+1 If 'n ' is the size of the array , Then there is 'n+1 ' number of possibilities. 'n' possibilities is when there is 1st element, 2nd element, 3rd element, 4th element, 5th element and 6th element. If the element is here, How many comparisons will it have to make ? It will have to do. . . 1. . . 2. . . 3 comparisons I've taken ' k ' as common out of everything. I removed this because this is different. And this I have added separately.

AP is Arithmetic Progression and GP is a geometric progression. AP is used in 'O ' a lot. APs are made as well as GPs sometimes. When there are questions on 'O' APs and GPs are used in the answers to questions on O. The formula of Average Case Complexity is All possible run times divided by the total number of possibilities. The Average Case Time is not generally asked for a unique algorithm. It can be asked for this type of algorithm , A simple algorithm that compares all of these. So what is the Average Case. Complexity ? O (n). Algo 1 was making 'n ' comparisons. Now we 'll see how many comparisons this will make. Algo 2 is a cunning person. It will match the first and last element and match it first. It made one comparison For a size of 10 array , As well as for an array of size 100. It is making the same comparison for all sizes of array.

The midpoint between 1 and 100 will be 50. Then you will discard this entire array that has elements greater than 50. If there are an even number of elements here, there are 2-2-4,2-6 elements. So the midpoint can be both 7 or 9. So I can take either of them as midpoint. You will keep halving an array until it gets over; In the worst case, you will find the element. You can halve 8 once, and it will become 4. 8 has become 3. How many times can you divide 16 ? You

MRF. Not MRF tyres, nor the bat one ; I 'll give you minimal. . . minimal Required functionalities. And some operations , that you can define later on your own. Let us understand this with the example of an array. Arrays can be found in a lot of programming languages. Arrays are found in C, C++, Java and Java. In Python, they are called 'lists' and in Java, they're called lists. There are some minimal required functionalities in arrays. For example, Get is done , Set is done. Along with that there can be a resize functionality.

An array is a collection of elements. Accessible by an index. The memory layout of the C programme is a code section. Inside the stack, there are local variations of a function. Inside a function is a stack. Inside an array is the heap. The kernel is the kernel of the kernels. In most programming languages, the index of an array starts at 0 It has a reason ; when we start from 0 , We get a certain advantage but I wo n't go there. But, you can search on the internet There 's a paper by jee-stars. In that they 've explained what will happen if it starts from 1 or 0. How calculations will become easy. If you say that you do n't want 38 , you want a greater one Then you wo n't get it. When you requested a memory till 38, then it is possible that some other application was given some memory by the compiler. It will be doing its job with this memory. Otherwise in the same programme It must have allocated a variable in correspondence with that memory. So you 're saying that you want to increase it.So you can not make it larger in this array. You can increase it in the linked list, we 'll talk about that ahead. If you skip the practice set it 'll be over in 6 hours.

An array can not be resized. Directly I can not resize it , I ca n't increase it. It wo n't get this block ; no. It can be. resized this way. Why should we use an array ? Why did we make an array? We made the array because Because at one point I have these 10 addresses. Get and Set is very easy in the C language. The programming language gives them to you by default. But we 'll implement this array ADT with the help of structure structure. This is our abstract data type. We 'll build it however we want to. And we'll add many more operations. In my ADT I will provide the user with the ability to use my Add method. So that he/she will be able to add lots of arrays directly. That why we use arrays. Faster retrieval , faster updation. Where does the scam happen in the array ? At the time of deletion and insertion. If I want to insert an element , it becomes costly. Because I 'll have to move it. There are some disadvantages too , but it varies from use case to use case. And you 'll get that by practising. And I will ensure that you get it.

In an abstract data type, abstraction means hiding the implementation details. And you are being given an assurance ; a provision To carry out these operations. Array ADT holds a collection of elements It can float , it can be int. It can be classes in C++. And it can also be structure in C language. The array can be accessed and updated in O (1) time. Because only one calculation has to be done And it will take constant time. To implement all of these. We 'll

will define my structure here. How can we make a function of linked list traversal ? What do I do to make function of traversal of linked lists? What will happen that with the help of this pointer , I will print the elements inside the whole list one by one. How will it be possible ? It should be possible that first my pointer will point to the head node then after that I will find the element next to the. head node. So what will it take , this is just what I want (struct Node * ptr) i need pointer I want a pointer that is of type Node * And it 's pointing , where at the head.

So to linked list traversal , I have given a pointer which is of type struct Node * : I get one. I will print the data and find that next node and run that data. And I 'll keep doing this until it becomes null until it points to null. I did n't even call this function. Head is a pointer, of struct Node * type So over here this is , struct Node* ptr Okay So what will I do here , I save it and run it. And you guys see , Element 7, Element 11, Element 66 and Element 66. If it also had a fourth node if it also has a fourth, i. e. I would put some data in the third node. And I have n't made the fourth , so I have to make the fourth. So look , made the head made the second, made the third and made the. fourth. This is the last element of the list because the next one is pointing to NULL So here is the chain , it 's closed Sometimes I had to add an element , can do it at any location. So I would just like to say that you guys here of four element , five element, five element , 10 element here are able to make linked list of it. Relatively , very few people have access to this playlist I would say by bookmark it and save it.

Only 1 or 2 people are coming forward to share the course in Data Structures. So far maximum 2-3 people must have done so if you do then thank you so much. Do n't forget to tag me Below I have given the link of my Instagram in the pin comment comment.

The Insertion and Deletion all this is going to come in linked list , it 's all very easy. It is Insert after and we will also talk about their complexity. What we are going to do is simple in a way to understand the problems of C programming. Head node is pointing to the first element of my linked list. I get access to linked list , with the help of head node. I have to fit it in the starting. How do i have to do to do this , you think for yourself First of all, what will I do to link it , something like this. There are 5 lakh elements, so how much time is it taking , is it increasing , if it is increasing , then because of what fear is increasing, that is time complexity. No matter how many elements you are ahead of here , it has not even seen the element. It is not depending on (n) at all And but when it does n't depend on (n), we had already seen in the lesson.

We can insert node at any one index. If I want to have an insert here, here is an insert , then I am calling it index 1. I 'll write a function named Insert at index and what do i do inside it first i will create a node. After that, point it to the next : from here. Complexity of Insert in between will have complexity O (n) O (n) : of our worst case complexity of insert in between. So this is very