this will become 10 lakhs. So the higher degree term in the polynomial In any equation The most impactful term It is taken ok. So I picked this because in comparison with n to the power 0 it is big. And I want to see things in a simple way.

Big O of n square. Big O is a log that scales according to the time required to run your algorithm. Linearly if your time scales with the input size. If it runs in linear time Big O. If your time runs in constant time, Big O is 1 ok. O in the industry means the order of And its mathematical definition that I will tell you. Industry definition is a minimum of this. But when you are answering in industry Then industry definition is used. When I use its mathematical definition then I say Big O But when I give industry definition Or I am answering any interview. Then I will say an order of because big O has a different definition. But they are used interchangeably. The graph of Big O of 1 is plotted like this. It does not mean it is 1 's graph. Do n't confuse it with the y=1 graph. This is the graph of x=k. Constant , whatever constant was there in constant time it was running. And this y=mx+c It can be somewhat distorted.

Operations on Arrays in Data Structures: Traversal, Insertion, Deletion and Searching

CodeWithHarry

Today we 'll talk about some operations in array. The first primary operation is called traversal. It 's the easiest thing in array, it 's like 2x2=4. In traversal, every array element has to be visited once. If you want to print all the array elements or you may want to set this array. If you 're confused why 0, 1, 2, 3 etc. or similar doubts, then your situation is such that you need to write C language. At home, do C language, 15 hours sleep a little less one day and complete it. At about 11 at night, you 'll be done with it. I 'm kidding, watch it in breaks whenever possible. I've included practice in it. So here , we 've seen traversal. I'll tell you what I was going to say. In traversal, we can use as many elements present. So, you can make two variables, one will be capacity. The capacity here will be 100. You can make size , it will be 5. Size means how many elements you are using. Capacity shows that this is the potential. If a pot can hold 10 liters and you store only a liter in it, but you bought it to store more if needed in future. That's why you take a higher capacity in array and the size is 5.

In school assembly, children used to stand in a line. The children are standing here and suppose something is being given out like ice cream. If I want 5 to be at index 2 , I 'll have to shift them first. Though in school you join the line first and shift later. You 'll make space first, add 15 blocks. This goes up to 99, 0, 1, 2, 3, 4. After that, what did I do for insertion ? I shifted the elements. Now, if you 're a very lucky person , then you 'll get index number 5 for insertion. All this is in the notes , nothing to worry about. I have provided everything , just download the notes. Zipping is for your convenience, use a computer to unzip. I zip it so that all codes, source codes, and pdfs are in one place. Comment below what the best case deletion run time is and what the worst case is. It 'll be O (1) in best case, you had to delete the last one. In worst case, the element order does n't matter. In both best case and worst case cases, the size does n' matter.

Doubly Linked Lists Explained With Code in C Language

CodeWithHarry

We have a pointer pointing to the first node. We have seen this here in the linked list. In that too there are operations like insertion , deletion , search too. In doing that , you will have to use the same concepts that you people have in singly linked lists. There are many uses of doubly linked list here and one of the uses of it you guys can reverse it easily by swapping the pointers. At the same time, if you have a pointer to the last node then you can walk in both this direction and in this direction. There are other uses too I will also make you guys get the practice sets of linked list etc. in it : you have extra information in this , what is the node with your previous : you can also know this. A lot of people have access but still the number of people who picked up the course : not everyone has access from here I want all to access , to bookmark I hope you guys must be easy to understand because you guys have accessed : this playlist If you have n't done it then definitely you will have a little problem if you're a beginner. bookmark this playlist by clicking here Click here to save

I prefer to use singly linked list if I feel like it will be more feasible. If I have memory constraints or memory constraints i will be happy with my singly link list. You come in a doubly linked linked list when you may have to go back and forth in both directions. The best thing is that a pointer who is traveling here : That can change his mood and start traveling here. I want to give a challenge to you guys here. I want you guys to write a function that first traverses it in this direction then once your pointer reaches here it comes back and traverses again that is , once a straight linked list is being printed here. Then you see people getting the reverse linked list printed. The person who is in realistic time can finish this course can finish it. We will wrap the topic of linked list and come later we will return to linked list And after that we will see the people who : this , but you have to code the doubly linked list. I am putting it in all the codes nowadays I have taken the feedback very well : Thank you guys So let 's what we will do now.

C Code For Implementing Stack Using Array in Data Structures

CodeWithHarry

In this course in our Data Structures course we took a closer look at Stacks and we are now here to implement Stacks : In C language. Now we will run in code , i. e. in Visual Studio Code and we will write here how the stack is made. There can be a custom datatype but for learning purpose we read only Stack of Integers. I am going to do a little variation here and you guys see what variation is going to happen I made a very simple stack here. Tell me in the comment below by putting a timestamp will it be right and if it is not right then why will it not be You guys tell me by commenting below. The most talked about ones are push operations : one , and pop operations : to insert an element inside the stack. But before that, before we push or pop , we ask you a question and my question is , you guys tell me one thing. what is push and pop? Is there any condition for them what i mean to say here is can i push if stack is full : Or can I pop if the stack is empty or not?

IsEmpty :sEmpty checks if this stack is not empty or not. If the top value is (-1) then the stack will be empty. If my stack is empty, then I will write if (ptr - > top == -1) ok, if i did like this I will return (1) What does (1) mean, return (1) means true and i will do (else) return (0) Now I always prefer to write my function in such a way that I am applying (if) also (else) because readability increases. If I write that the top of the ptr is equal to that of the pointer (size-1) then the stack full : What is Stack , Stack Full : So here we will validate these two functions , once once. Once. We have created our functions with the help of which we will push into the stack. I want Interact, comment below and I want the comment to be filled with these answers. I would love to hear that you guys actually doing the implementation that I am doing. I will be very happy you guys will do this and so many people have accessed this playlist. At least once you open the playlist and see the topic. What are the topics here. I have written down all the topics : and after that making notes for all : , given the links : in all the descriptions.

Everything is lying on the site, all the source code , all your notes , I have put all the notes. So with that said , So I would say that you guys please lock this video if you guys like this video If you guys want to make videos of Data Structure very quickly , many people are asking : Make videos of data structure quickly.

Peek Operation in Stack Using Arrays (With C Code & Explanation)

CodeWithHarry

In today's lecture we will talk about Peek Operation that is , an operation that will tell you about what value : at which position inside a stack. I want to give an answer to What happens when we pop things out of the stack then we keep going down the top value that is from 2 to 1 , becoming 1 to 0. What is this peek? What are these peek operations ? This operation peek , what is this thing I tell about it So what will it take It will take my position (i) When we are using stack, we can not start index from 0 We 'll start with 1 because it 's easy to understand for them 1, 2, 3, 4 are difficult for them to understand. In terms of my position : , if I want to write index then I will write Top I wrote Top here , (-i +1) because this position : is not an index So here 's how I came from And look, if you put the value of (i) as 1 then this formula will come all. When it is 2 then the index of the array is 1, ok And this is 3 so this is 0 So now this is filled with : 3 elements this is our stack If it was out of 10 then what would be the value of my top Right now our value of top is 2

do (Top-1) You will do , as if the first position is taken by you , So you 'll come one step down (Top-2) if you do then you will come down 2 steps. But once you 're 2 steps down , you need 2 So (+1) for adjustments , okay So (Top -i +1) you all must have understood : where it came from So now i have (Top -i -1) So it is very easy for me to complete the peek function. The video was kept : 15 hours with notes , watch it and you must have seen yourself So far (C) language is well understood by everyone. So I would also like to re-use the old code a bit And along with that, I will also tell you how you can peek the stack here. This playlist is for you. No this one I am not talking about another playlist. I have made one of the practice programs I made. I pretty much did what I told you guys here. And if you do dry run then you will understand. If you feel that all this is not happening to me I do not understand.

have no excuses that from where this much data came this can also happen that your expression contains 10 crore opening parenthesis.

C is a C video of a C program which uses a function to match parenthesis matching using stack. The program does not tell about the validity of this expression. This will only match the parenthesis , i will show you the proof of this. i will provide all the source codes to you time to time and notes will be uploaded soon. If i do it like this , so this expression is valid this expression is not valid. If talk about the parenthesis of this expression so they are matching, so this program will tell me that if your parenthesis are matching or not. So you always keep in mind that this parenthesis matching program will never tell you about validity of the expression. Please share this playlist if possible on instagram.

Multiple Parenthesis Matching Using Stack with C Code

CodeWithHarry

If any expression has more than 1 parenthesis, suppose I write an expression A= { 7- (3-2) + [8+ (99-11)] something like this. If we get this type of parenthesis Then we will push in stack just like we used to solve our single parenthesis problem. So when I am popping it then I will matched with which one ? I will match it with closing parenthesis which was encountered. And with that there is one additional step which I will show you. When the full expression finishes if yes then it is balanced , Else not balanced you might be thinking why explain it so quickly. So , I did push the same way I was doing. But while popping I am adding one more step that whatever is popping is that matching the closing parenthesis or not like I pop this when I am here the topmost element I popped. So is this opening parenthesis is matching this if yes. If yes then pop successful. I 'll pop it. If it is this then I will push it. So, I pushed it. When I came here I pushed then when I popped is it matching ? Yes then yes pop is successful and again because this is We need to do a balanced one too else else else it will be wrong. (7-11+{ 22-8+2 } - [11+7]). I took a big expression but we will practice it. We just need to convert that code to logic. We will code according to the said as we made the logic.

When I am popping, listen carefully , then I will take this in a character. named popped_ch and name it here inside the function. So, I popped this character and after that, that my popped character which I popped, and the top element of my stack are matching or not. I will push exp [i] = '' \0 '' means till end of expression. Match ( ) function returns either 0 or 1 if it matches. I will pop whenever my exp [i] will be closing. or closing } or closing]. Else I am ignoring the character of my expression. I am doing this because my opening character will be stackTop one and closing will be popped. This program will tell you whether it is balanced or not. I hope if you understood this then you can do balance any parenthesis. There will be no problem. This program tells you whether the parenthesis are

Infix To Postfix Using Stack

CodeWithHarry

Today we are going to see how an infix is converted into postfix using stack. Let me tell you beforehand that this video is going to be very important. And watch the video till end else you wo n't be able to understand. The way we saw will be difficult to program so with the help of stack , How will I do it just watch. So see - has what precedence ? It has 1 precedence. So can I push it ? No. because it can not stay on top of king. - has 1. and this has 2 so it. so it can. stay. on top. So , what happened is when I reached / it had 2 precedence so I pushed. But as soon as it came on - then what happened - can not. stay on the top of /. because - is smaller than / so it is smaller so it cannot stay on. top of it. So - and - have same precedence then too we will pop. So, what we will add this - here. We pop this. We say infix to postfix expression manually. Let 's check that the procedure I have used with the help of stack is giving right answer. First of all, I will parenthesize it. I/j y/z will evaluate first so I will put parenthesis there. Then I will write it here ([xyz/-]) - [kd *] and now xyz/kd * - - X will come as it is. Then + will go in stack. This is operator this is precedence will write p only due to space. After that will write y here , then * can it be added above this. Then - can come so I pushed - and I

In last lecture we have converted some expression in postfix , do practice them. After practice you can see how conversions are done. So if I do it with stack Will do it quickly so look carefully and will take red color to make a stack. Will make stack longer so can cut and pop elements. If you know data structures then it will help you in placements. You will get help in comparative programming , logic building will be good. Programs will start creating and many things. So, you need to watch. If you like this video then do like. And with that if you think this course is helpful , You can take screenshot of it and share on Instagram. And I will reshare your story.