RECURSION 05

Alright guys hope you had learnt about the powerful technique of recursion, i hope you remember search algorithm that you had studied previously in your previous weeks namely the binary search, let me give you a very quick overview of what is does? It is basically how we search for in a dictionary or in a telephone directory something like that. Why did i take dictionary or telephone directory? Because the words or the contents in the directory are sorted or that is arranged in some ascending order they are arranged in the alphabetical order so that is why we can apply this technique so binary search basically calls for a sorted list, you want a list that is arranged in some specific order, either ascending or descending it calls for list arranged in their order and in that list you want to see whether a particular number is present in the list or not whether a particular item is present in the list or not, so what would you do? You would check the middle of the list, if that is your required item you are done else you check if it is less than your required number or greater than your required number, accordingly you will branch on to the left half of the list or the right half of the list. So this is how your binary search works you have seen a iterative version of binary search in your previous weeks i guess so that is easier but still an easier thing like how i had given you an intuitive version of binary search like you have to scan into the left half and you have to scan into the right half something like that i had given you an intuitive version so such a intuitive thing can be easily translated to code when you use this technique of recursion so let us see how we can use recursion through simplify the process of writing code for binary search. Let me say i will define binary search, binary search define binary search for this i need list or you can call it list l i need a list l, some element x which we want to find, you were processing it by sing the index values, you would have seen that till now you may be familiar but still i am telling you a very quick recap, indexes in computers starts from zero human start counting from one cut computers start counting from zero so the counting starts from zero so if your list has five elements the counting would be done as zero one two three four, zeroth element, first element, second element, third element, fourth element this is how your computer counts so you have to give the starting index and the ending index these are required for your binary search right, so initially your starting index is zero ending index is actual end of your list so that is entire list then as you scan the middle element you keep discarding one half of the list and just keep searching in the other half so that time it will change so these are the Para things you need to do binary search, you need a list you need the element x which you want to search you need a start and end index of the list where you want that is you won't search the entire list all the time, you will search parts of list by discarding the unnecessary half this is how your binary search works so we needed this four parameters ok let me define as i had said recursion requires base case, what is the base case? When we have only one element left in our list, if just one element is left in our list then we have to see that particular element if that is the required element then you say yes required element has been found else you say element is not found ok so for element not found i am going to return value minus one i am going to assume that my list has all positive values and minus is not the value in my list also i am going to return the position of the element, i am not going to return the element even if have negative element i don't mind let's find i am going to return the binary search basically here i am going to return the position in which the element is present at the list, so that is what i am going to return so position starts from zero right? so if the element is not found i will return minus one that is the element is found somewhere outside the list, that's what basically i mean so i will return minus one in case if it is not found so base case is the base case is one element just one element in the list that is the base case so if there is just one element in the list how would you start and end, they would be equal, start and end will be the same value because there is only one element that is the start as well that is the end so i will start using the start and end indices, if start is equal to end i would check if l of start or end you can use anything inter change of it because they both are equal in this case, if that are equal to your required element x you return the start value, that is return that position hence there is just one element say suppose assume you have just one element ten in your list and what you want to search is fifty, fifty is not present in the list and it is having only one element then you have to say element not found so for that i said the code i am going to use encoding i am going to use is minus one so i will return ,minus one, element fifty is present in the minus oneth position meaning that it is not present ok, so i will return minus one in that case ok so this is the base case if the base case is not true then else that is there are more elements in a list, in that case you have to find the middle element and you have to discard the unnecessary half and search through the required half, you have to split the array into halves and search through the required half that is what you have to do right? so divide the divide the array into halves this thing we have to do for that we need to find the mid position, mid position is nothing but start plus end here i have to put up bracket because i need to calculate the sum first then i have to divide by two this is the midpoint right this is intuitive start and so here will be your start and here will be your end the midpoint will be start plus end divide by two this will be your midpoint when you divide there is a possibility that you can get some value that is floating point may be seven by two three point five, what is the meaning for position three point five? Either you have to take it to three or to four so for that we are using a functionality int so we are type casting it to integer because position is a integer we want it in a integer that is first position second position third position this is what we want, we don't want three point five position so will use int so it will have its own conversion it will take it to three point five to three or four maybe we can run here and check int of three point five let me give three so it is taking to floor functionality, floor is nothing but the integer that is closest to it the greatest value of the integer that is closest to it if we take the number line three point five is somewhere here three is here and four is here it is going to the left side so int functionality is going to the left side and taking the greatest integer two one everything is present in the left but the greatest in the left the integer is three so three is returned as the answer. So that will be taken here let me start a new console better because for this program new console ok so this is the midpoint we have computed, we have to check mid element is the required element so if the if l of mid is equal to x that is the case we have done we have found the element return the mid element else i have to check if it is greater or less so i would say l of mid is greater than x means you have a sorted list and the middle element is greater than your required element so where should you search, definitely your element will not be in your right half you can discard the right half and you need to search the left half so your array would now shrink from the starting position to mid minus one so mid position didn’t have the element also there is no chance that it will be after mid so it would be before mid so we need to shrink the array up till mid minus one position that is one position before the mid till that we need to shrink the array, that is we will do as return the result of binary search on the same list for the same element x start position is not change see because we are we have to check the left half of the array so the start position is the same but the end position is not the precious end position, previously we have the bigger list we want to split it so the end is mid minus one why mid minus one? Because at mid position we had checked, at mid position didn’t contain the element x so an x is the smaller value that is x is l of mid is greater than x meaning that x is lesser than l of mid so x being less than the mid value will be present in the left half so from starting position till one position before mid whatever is the array left search in that array and return the answer if what we mean, see it is really intuitive see this is the left half, this is the left half than using of iterative method recursion is easy you can translate your intuition into code very easily if you use the recursive ideology ok so this is the recursive way to call the left half of the array, call the procedure on the left half of the array search the left half else you have to search the right half, so for the right half end value is the same but starting point is different right? we had searched till mid we didn’t find it at the position mid and the value x is greater than the mid value so it would be present from one anywhere from one position after mid up till the actual end point right? so your start value will change or i would say return binary search result on l, x mid plus one and end, the modify starting point is mid plus one till mid position we had checked there was no element that is why we are checking the path after the midpoint, here we are checking the path after the midpoint, here we are checking the path before the midpoint that is the left half and the right half whatever we had it intuitively in our mind we are translating it to code very easily so this is the power of the recursion so it would recursively keep computing and we will get the answer so you can take any example, i had given you an example of how recursion works in the factorial screen cast similar to that you take a list you try applying binary search on it, you try how recursion works, you would have understood how recursion works so this particular search the bigger list depends on the value of searching on the smaller list so whatever the result you get after searching on smaller list will be translated back and will be returned as the result for searching on a bigger list this is how your binary search in the recursive manner so we had defined binary search now we have to used it, right? so let me give a random list you can also change it with your version you ask the user to input how many numbers he want to input, you input that many number of numbers or till he presses some keys you keep getting input you can by now i guess you all are familiar with the various conditional construes using that you can modify the code as you wish, now my motto is to demonstrate binary search so i would just give a smaller list twenty forty five sixty seventy ninety this is my list i have given please note that i have given a list in the sorted order, if it is not sorted you have to apply sorting first then after you get the sorted list then you have pass the sorted list to the binary search, binary search expects the sorted list so that is very important please make a note of it, so this is my list and i will input the x value from the user ok, let me input the value of x from the user and i have to typecast the input in my Mac machine so i am type casting you please follow as per your machine dependency input enter search key search key i would say that is what number you want to search, i would input the search key from the user, that is my x so binary search will return the index and so let me store it as index is equal to binary search on the list l search key x starting position is always zero for the bigger list initially we start with the entire list zero and the end position is zero to length minus one this is the five element list so it will be four so let me generalise it so let me say length of l this will give the length of the list that is how many elements are present in a list will be return by this length of the list minus one this is the ending index so you initially start of your search with this particular index this is the suppose my search key is eighty this will be the entire key is passed so it would find the mid value sixty it is righty is greater than sixty so this half is not needed this is neglected and this particular thing will be executed and seventy ninety this will be my new list in this list i will search for my key seventy and ninety this will be taken as the mid value as we have seen that floor value is taken so this value is taken as mid value, eighty is still greater than seventy so it will skip to the right half so here there is only one element ninety so this case would be executed ninety is not equal to eighty so it will return minus five so i will come to know that this thing is not present so something like that it works you can trace through it i hope now you are clear with how recursion works and you are clear with the binary search concept too so you can understand it you all that you need is take a pen and paper and trace through how some example work, that is what is required with that you can easily understand the concept ok so it will return the index where the element is present in the list ok i have got the index so if i just say the index this is not enough for a normal user for us element sixty is present at the third position if i say two they will say what is this computers doesn’t know even this so we have to translate it into a human friendly format right, so basically what is that we have to do, we have to add one, one is represented as zero, the index in second position is index one, third position is index two so basically whatever is the index it returns add one to it and display it to the user and if it returns minus one, you should not say it is present in the zero you should say that element is not found, print appropriate message so first we will give a check that if index is equal to minus one in that case you should print x value not found ok x value is not found ok else you should print, that is minus one is the not the case that you should print that x value is found at found at position index plus one, computers counting system and the humans counting system differ by one that’s why we are adding one and display ok so this is how you have done, let me save the code i hope you are clear with it we have a list we input a search key this you can even modify it to getting the input for the list element is from the user, you have to sort the list please note that you have to send a sorted list for binary search to occur, so you have to sort that list input a search key then you apply binary search on it how it works? If there is just one element in the list it will check if that element is the required x element x, if it is the case it will return the index else it will return minus one so whenever the element is not found it will return minus one that is how we have encoded it. In case if the list have more than one element what it does is, it will find the mid element and based on the mid element since the array is sorted it will discard one of the halves if the element required is exactly the mid element we are done if it is less than the mid element you have to search the left half so it will discard the right half it will search only the left half if it is greater you have to search only the right half it will discard the left half so for this to understand this really well i would suggest that you take a twenty element list basically and you try tracing it on paper, basically to understand this clearly you should work out a lot on papers and less on computers, computers can do this in fraction of second but humans to understand this strategy it requires some practice so please take a pen and paper take some twenty element list randomly you sort it maybe for that you can use the computers as well because sorting a twenty element list may take some time, so you can use the computer as well you sort it or you take the sorted list of the twenty elements you randomly give some search element you give some element which is present in the list as well as some element which is not present in the list try to understand how the various runs of the programs are and you will really understand the process very easily after practice all that need is needed is practice please do practice practice practice that’s it now let us run this program let me run it ok it is asking me to enter a search key let me enter eighty, eighty not found perfect! Ok now let me enter ninety, let me enter ninety, ninety is found at position five perfect. We got it when i entered eighty, eighty not found at this list so it’s says eighty not found and when you enter ninety, ninety is found here it is found at position five so it says ninety is found at position five so it works, i would recommend that please you take some pen and paper and work through the various example, work through list with large elements as well here just that i wanted to demonstrate the recursion technique i had taken a smaller list and already sorted list, you try different things unsorted list you sort it then apply binary search at huge list, a list where you get somewhere it is if it is a twenty element list try to find the sixth seventh element try to find the eighteenth element try to find something exactly near middle or near the middle something like that, you try various possibilities you will understand actually what is logic of binary search how it works, all that is needed is practice with pen and paper. Keep practicing thanks for watching have a nice day.