1Q.) Find Input array is sorted or not, using recursion

Solution:-

example array arr[] = {2, 6, 7, 9, 3, 10}

→ The way to solve this is simply we can check if element at index [i] should be less than element at [i+1] index.

→ We only need to check till (n-1) elements be at n^{th} element their is no [i+1] element to check. Hence this becomes our base condition. i.e., if we reach till i == (n-1) & there is no false case encountered till now, tells us that this array is sorted.

dry run

2	6	7	9	3	10
i = 0	1	2	3	4	5

1) f(0) → [i] > [i+1]
2 > 6 false

2) f(1) → 6 > 7 false

3) f(2) → 7 > 9 false

4) f(3) → 9 > 3 True

Recurssion stops & returns array is not sorted.

code:-

```
#include <iostream>
using namespace std;
bool IsSortedArray(int *A, int size, int i)
{
    if(i == (size-1)) // base condition
    {
        return true;
    }
    if(A[i] > A[i+1])
    {
        return false;
    }
    return IsSortedArray(A, size, i+1) //Recursive Call
}
int main() {
    int arr[] = {10, 20, -15, 30, 40};
    cout << ((IsSortedArray(arr, sizeof(arr)/sizeof(arr[0]), 0) == 0) ?
            "Array is not sorted" : "Array is sorted");
}
```

Processing

Scanned with OKEN Scanner

2Q) Binary Search using recursion

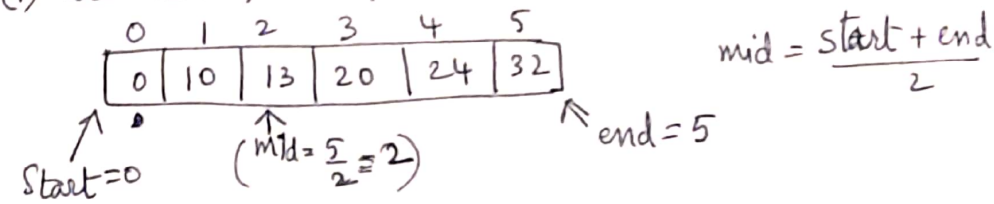
→ To perform Binary Search Array should be sorted.

→ In Binary search we will ~~check~~ calculate a mid based on the start & end index of the array hence these become the parameters for our functions will be calculated for every recursive call.

→ In simple binary search is explained in below pseudocode;

(i) if you are searching for 24 in arr = [0, 10, 13, 20, 24, 32];

(i) set start, end & mid.



(ii) check if mid == 24 else; ~~mid~~

(a) if mid > 24 (or) mid < 24

→ if greater search left side of mid

→ if smaller search right side of mid.

Code:-

```
#include <iostream>
using namespace std;
int RecursiveBinarySearch (int *A, int Key, int start, int end)
{
   int mid = start + ((end - start)/2);
   if(start > end) return -1;
   else
   {  if(A[mid] == Key) return mid;
      else if (A[mid] > Key) // left search
        return RecursiveBinarySearch(A, Key, start, mid -1);
      else // right search
        return RecursiveBinarySearch(A, Key, mid+1, end);
   }
}
int main(){
  int arr[] = {10,20,30,40,50}; // Sorted array
  cout << RecursiveBinarySearch(arr, 50, 0, (sizeof(arr)/sizeof(arr[0]))-1);
}
```

Scanned with OKEN Scanner

Subsequences of a String - using Recursion

→ Firstly its sequences not combinations.

so, for a string abc / cba is a combinations

Eg for a string ab / ba is combination.

combinations should not be taken, the sequence (order) must be only once.

→ example:-

Sequence of string:

1) ab → a, b, ab, "" → empty string.

2) abc → a, b, c, ab, bc, ac, abc, "" → empty string.

so, for 'K' characters in a string the possible sequences are 2^K i.e.,

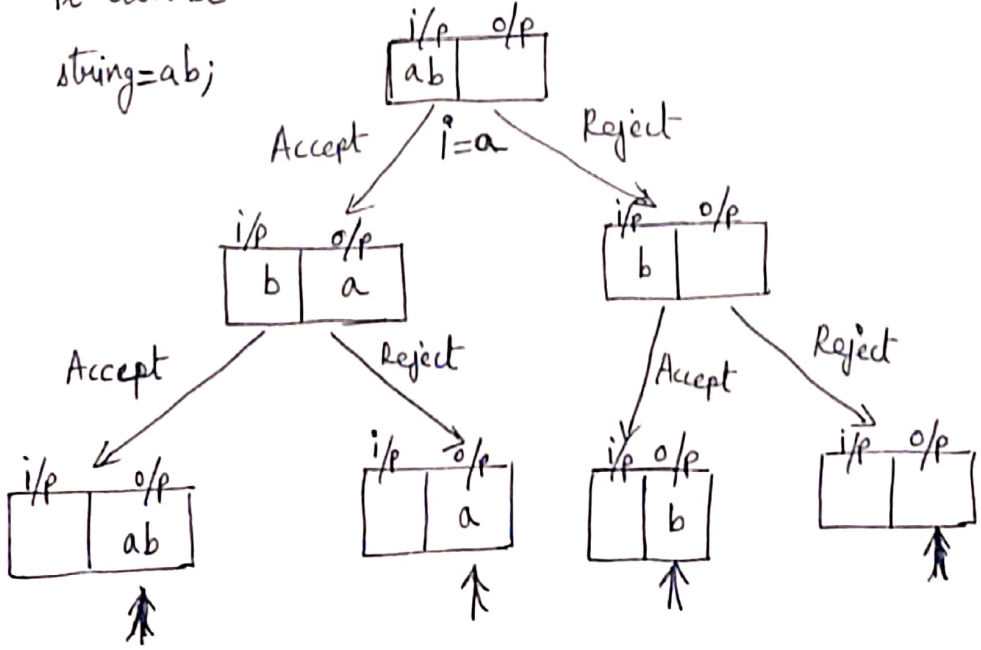
for "ab", K=2 i.e., $2^K \Rightarrow 2^2 = 2 \times 2 = 4$.

"abc", K=3 i.e., $2^K \Rightarrow 2^3 = 2 \times 2 \times 2 = 8$

and ~~so on~~.

so on.

→ so, to achieve these combinations through recursive program, it can be done like below, consider a function.



Scanned with OKEN Scanner

⇒ we will write a func which takes input string & output string bothas ~~params~~ formal parameters.

func (string input, string output).

⇒ In the input string we will pass the string ab & output will be empty. Each time we will pick an 0 i.e; the starting character & do two Operations either accept it or reject it.

note:- once the character is picked its removed from the input as shown in diagram.

→ using this we can observe from the diagram how the flow works.

→ // base condition → if you observe the diagram at the end level all the inputs becomes empty i.e if the input becomes empty we need to print output..

⇒ The above described process happens recursively!

code:-

```
#include <iostream>
using namespace std;
void PrintSequence(string input, string output)
{
  if (input == "") // base condition.
   {
     cout << output << " ";
     return;
   }
  PrintSequence (input.substr(1), output + input[0]); //accept
  PrintSequence (input.substr (1), output);
}
int main()
{ string s= "ab";
  PrintSubSequence (s," ");
}
```

Scanned with OKEN Scanner