Recursion

(class 3)

Date 12-03-2023

Ques Given an array, return True if array is sorted, & false if unsorted using recursion.

we linearly iterate over the array, at every ith element, check if i+1th element < ith element, the return false, else again call the function recursively for next ith element, and if we are at n-1th element means all the array is sorted. So, the base case is if (i == n-1), return true.

code:-

```
#include<iostream>
using namespace std;
bool checkSorted(int arr[], int &n,
                                int i)
{
    //base case
    if(i == n-1){
        return true;
    }
    if(arr[i+1] < arr[i]){
        return false;
    }
    return checkSorted(arr, n, i+1);
}
```

```
int main()
{
    int arr[] = {4, 3, 1, 2, 4, 4};
    int n = 6;
    int i = 0;
    bool isSorted = checkSorted(arr,
                                n, i);
    if(isSorted){
        cout << "Sorted";
    }
    else{
        cout << "unsorted";
    }
    return 0;
}
```

* Binary search using Recursion :-

we use same technique like we do in iteration method. we return only in two cases :- if (start > end), then return -1, means target not found, and another case is when the key is found.
So these cases become our base case in recursion.

code :-

```
#include<iostream>
using namespace std;
int binarysearch (vector<int> arr,
          int key, int start, int end)
{
    //base case 1
    if (start > end){
        return -1;
    }
    //base case 2
    int mid = (start + end)/2;
    if (key == arr[mid]) {
        return mid;
    }
    if(key < arr[mid]) {
        return binarySearch (arr,
                key, start, mid - 1);
    }
    else{
        return binarySearch(arr,
                     key, mid +1, end);
    }
}
int main()
{
    vector<int> arr{10, 15, 20, 26, 39,
                 44, 57, 69, 74, 88 };
    int key;
    cout << "enter key:";
    cin >> key;
    int n = arr.size();
    int start = 0, end = n - 1;
    int ans = binarySearch(arr, key,
                       start, end);
    if(ans == -1){
        cout << "value not found";
    }
    else{
        cout << "value found at" <<
             ans << " index.";
    }
    return 0;
}
```

using Ternary Operator:-

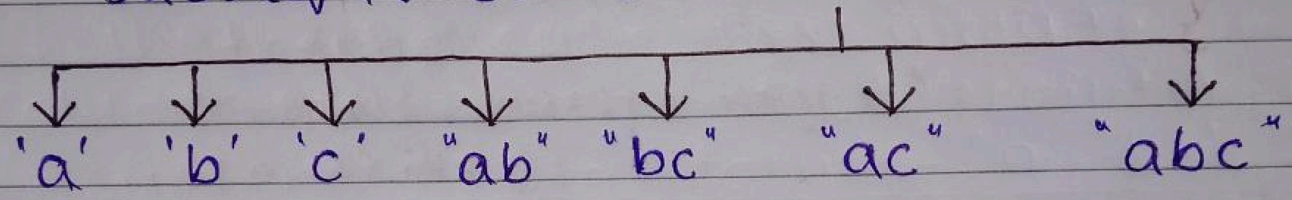
```
if(Key == arr[mid]){
    return mid;
}
else{
    return (Key < arr[mid]) ?
     binarySearch(arr, Key, start,
     mid-1) : binarySearch(arr,
     Key, mid+1, end);
}
```

Ques. Subsequences of a string using recursion. (V. Imp).

A subsequence of a string is a sequence that can be derived from the given string by deleting zero or more elements without changing the order of the remaining elements.

For example:-

Subsequences of "abc" are:-



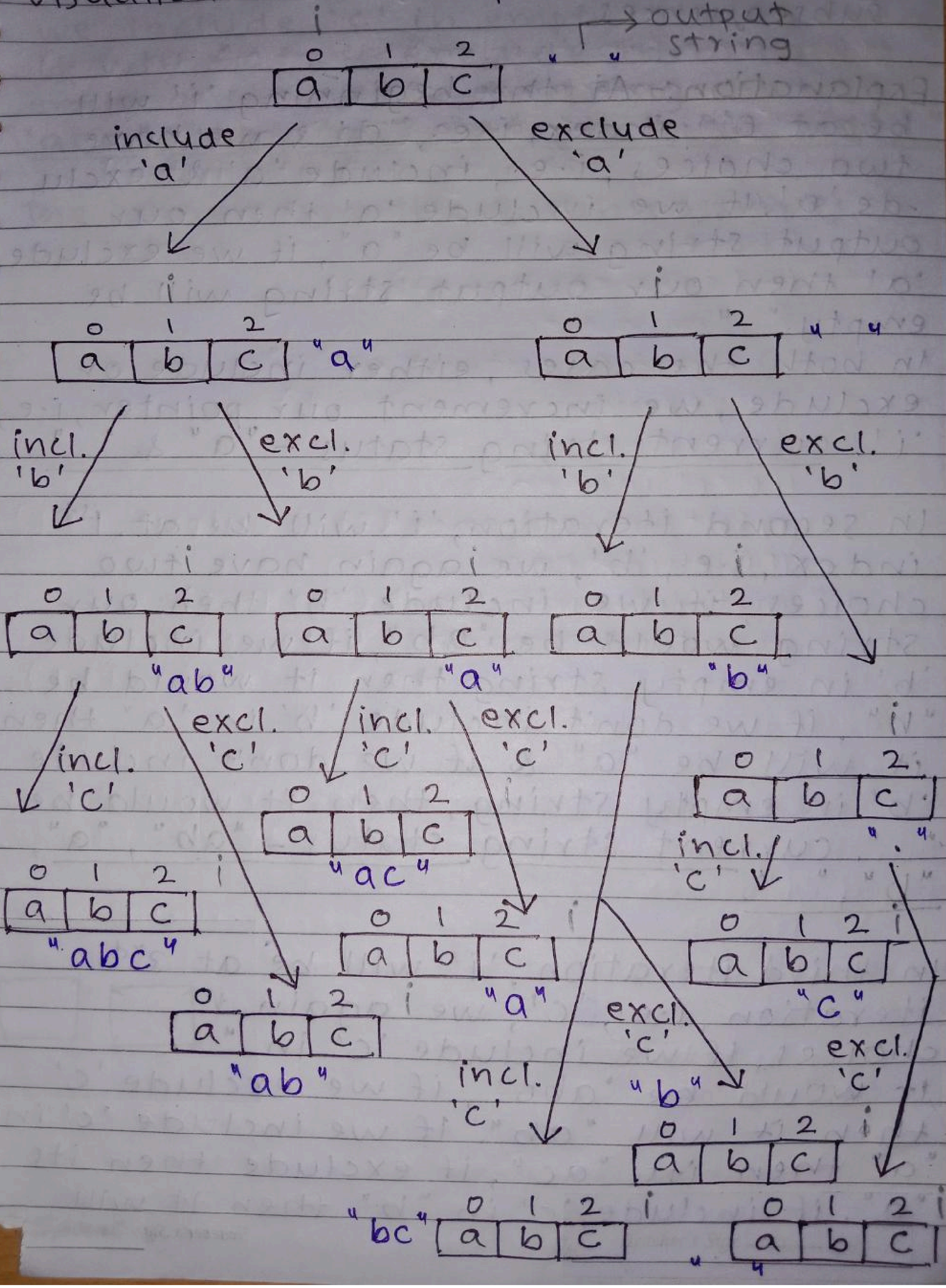
a	b	c		
✓	✓	✓	→	"abc"
✓	✓	×	→	"ab"
✓	×	✓	→	"ac"
✓	×	×	→	'a'
×	✓	✓	→	"bc"
×	✓	×	→	'b'
×	×	✓	→	'c'
×	×	×	→	' '

these are also called Power set.

Here, we observe a pattern, i.e., "Include exclude" pattern.

Note:- For 'n' characters in a string, the possible subsequences are "2^n".

visualization of this pattern :-



At the end nodes we get all our subsequences.

Explanation:- At the beginning 'i' will be at 0th index, i.e., 'a' & we have two choices, i.e. include 'a' & exclude 'a'. If we include 'a' then our output string will be "a", if we exclude 'a' then our output string will be empty " ".
In both the cases, either include or exclude, we increment our pointer, i.e, 'i'. current string status - "a" & " ".

In second iteration, 'i' will be at 1st index, i.e., 'b', we again have two choices, if we include 'b' then our string would be "ab", if we include 'b' in empty string then it would be "b", if we don't include 'b' in 'a' then it will be "a" & if we don't include 'b' in empty string, then it would be " ". current string status - "ab", "a", "b", " ".

In third iteration, 'i' will be at 3rd iteration, i.e., 'c', we again have two choices, If we include 'c' in "ab" then it would be "abc", if we exclude 'c' then it will "ab". If we include 'c' in "a" then its "ac", if exclude then its "a", if include 'c' in "b" then it will

"bc", if exclude then it will "b", if we include 'c' in empty string then it will "c", if exclude then its empty only, current string status -
"abc", "ab", "ac", "a", "bc", "b", "c", ""

The above are the 8 subsequences in "abc".

code :-

```
#include<iostream>
using namespace std;
void printSequence(string str,
                   string output, int i)
{
    //base case
    if(i >= str.length())
    {
        cout<< output << " ";
        return;
    }
    //exclude case
    printsubsequence(str, output, i+1);
    //include case
    output.push_back(str[i]);
    printsubsequence(str, output,
                               i+1);
}
```

our base case is, if 'i' index is >= length of string, then we print output string & return.

if we include a character, the first we add the 'i'th character in output string & again recursively call a function with increment of i.

if we exclude a character then we do nothing, but index 'i' will increment & recursive call.

```
int main()
{
    string str = "abc";
    string output = " ";
    int i = 0;
    cout << "all subsequences are:";
    printsubsequences(str, output, i);
    return 0;
}
```